
Sunshine

ReenigneArcher

Mar 14, 2024

ABOUT

1	Overview	1
1.1	About	1
1.2	System Requirements	1
1.3	Integrations	2
1.4	Support	2
1.5	Downloads	2
1.6	Stats	2
2	Setup	3
2.1	Binaries	3
2.2	Install	3
2.3	Usage	9
2.4	Network	11
2.5	Arguments	11
2.6	Shortcuts	11
2.7	Application List	11
2.8	Considerations	13
2.9	HDR Support	13
2.10	Tutorials and Guides	14
3	Docker	15
3.1	Important note	15
3.2	Build your own containers	15
3.3	Where used	16
3.4	Port and Volume mappings	16
3.5	Supported Architectures	18
4	Third Party Packages	19
4.1	AUR	19
4.2	Chocolatey	19
4.3	nixpkgs	19
4.4	Scoop	19
4.5	Solus	19
5	Guides	21
5.1	App Examples	21
5.2	Linux	26
6	Advanced Usage	37
6.1	Performance Tips	37
6.2	Configuration	37

6.3	General	38
6.4	Input	39
6.5	Audio/Video	44
6.6	Network	49
6.7	Config Files	52
6.8	Advanced	54
6.9	NVIDIA NVENC Encoder	57
6.10	Intel QuickSync Encoder	61
6.11	AMD AMF Encoder	63
6.12	VideoToolbox Encoder	65
6.13	Software Encoder	66
7	GameStream	69
7.1	Migration	69
7.2	Internet Streaming	69
7.3	Limitations	69
8	General	71
8.1	Forgotten Credentials	71
8.2	Web UI Access	71
8.3	Nvidia issues	71
9	Linux	73
9.1	Hardware Encoding fails	73
9.2	KMS Streaming fails	73
9.3	Gamescope compatibility	73
10	macOS	75
10.1	Dynamic session lookup failed	75
11	Windows	77
11.1	No gamepad detected	77
11.2	Permission denied	77
12	Build	79
12.1	Building Locally	79
12.2	Remote Build	79
13	Linux	81
13.1	Requirements	81
13.2	CUDA	84
13.3	Build	85
14	macOS	87
14.1	Requirements	87
14.2	Build	88
15	Windows	89
15.1	Requirements	89
15.2	Build	89
16	Contributing	91
16.1	Web UI	91
17	Localization	93
17.1	CrowdIn	93

17.2	Extraction	93
18	Testing	95
18.1	Clang Format	95
18.2	Sphinx	95
18.3	Unit Testing	96
19	Legal	97
19.1	Commercial Use	97
20	Source Code	99
20.1	Example Documentation Blocks	99
20.2	Source	100
21	Changelog	177
21.1	0.22.1 - 2024-03-13	177
21.2	0.22.0 - 2024-03-03	178
21.3	0.21.0 - 2023-10-15	181
21.4	0.20.0 - 2023-05-28	184
21.5	0.19.1 - 2023-03-30	187
21.6	0.19.0 - 2023-03-29	187
21.7	0.18.4 - 2023-02-20	188
21.8	0.18.3 - 2023-02-13	189
21.9	0.18.2 - 2023-02-13	189
21.10	0.18.1 - 2023-01-31	189
21.11	0.18.0 - 2023-01-29	189
21.12	0.17.0 - 2023-01-08	190
21.13	0.16.0 - 2022-12-13	192
21.14	0.15.0 - 2022-10-30	192
21.15	0.14.1 - 2022-08-09	193
21.16	0.14.0 - 2022-06-15	193
21.17	0.13.0 - 2022-02-27	194
21.18	0.12.0 - 2022-02-13	194
21.19	0.11.1 - 2021-10-04	194
21.20	0.11.0 - 2021-10-04	194
21.21	0.10.1 - 2021-08-21	195
21.22	0.10.0 - 2021-08-20	195
21.23	0.9.0 - 2021-07-11	195
21.24	0.8.0 - 2021-06-30	196
21.25	0.7.7 - 2021-06-24	196
21.26	0.7.1 - 2021-06-18	196
21.27	0.7.0 - 2021-06-16	196
21.28	0.6.0 - 2021-05-26	196
21.29	0.5.0 - 2021-05-13	197
21.30	0.4.0 - 2020-05-03	197
21.31	0.3.1 - 2020-04-24	197
21.32	0.3.0 - 2020-04-23	197
21.33	0.2.0 - 2020-03-21	197
21.34	0.1.1 - 2020-01-30	198
21.35	0.1.0 - 2020-01-27	198
	Index	199

OVERVIEW

LizardByte has the full documentation hosted on [Read the Docs](#).

1.1 About

Sunshine is a self-hosted game stream host for Moonlight. Offering low latency, cloud gaming server capabilities with support for AMD, Intel, and Nvidia GPUs for hardware encoding. Software encoding is also available. You can connect to Sunshine from any Moonlight client on a variety of devices. A web UI is provided to allow configuration, and client pairing, from your favorite web browser. Pair from the local server or any mobile device.

1.2 System Requirements

Warning: This table is a work in progress. Do not purchase hardware based on this.

Minimum Requirements

GPU	AMD: VCE 1.0 or higher, see obs-amd hardware support Intel: VA-API-compatible, see: VA-API hardware support Nvidia: NVENC enabled cards, see nvenc support matrix
CPU	AMD: Ryzen 3 or higher Intel: Core i3 or higher
RAM	4GB or more
OS	Windows: 10+ (Windows Server not supported) macOS: 12+ Linux/Debian: 11 (bullseye) Linux/Fedora: 38+ Linux/Ubuntu: 20.04+ (focal)
Network	Host: 5GHz, 802.11ac Client: 5GHz, 802.11ac

4k Suggestions

GPU	AMD: Video Coding Engine 3.1 or higher
	Intel: HD Graphics 510 or higher
	Nvidia: GeForce GTX 1080 or higher
CPU	AMD: Ryzen 5 or higher
	Intel: Core i5 or higher
Network	Host: CAT5e ethernet or better
	Client: CAT5e ethernet or better

HDR Suggestions

GPU	AMD: Video Coding Engine 3.4 or higher
	Intel: UHD Graphics 730 or higher
	Nvidia: Pascal-based GPU (GTX 10-series) or higher
CPU	AMD: todo
	Intel: todo
Network	Host: CAT5e ethernet or better
	Client: CAT5e ethernet or better

1.3 Integrations

1.4 Support

Our support methods are listed in our [LizardByte Docs](#).

1.5 Downloads

1.6 Stats

The recommended method for running Sunshine is to use the [binaries](#) bundled with the [latest release](#).

2.1 Binaries

Binaries of Sunshine are created for each release. They are available for Linux, macOS, and Windows. Binaries can be found in the [latest release](#).

Tip: Some third party packages also exist. See [Third Party Packages](#). No support will be provided for third party packages!

2.2 Install

Warning: The Docker images are not recommended for most users. No support will be provided!

Docker images are available on [Dockerhub.io](#) and [ghcr.io](#).

See [Docker](#) for additional information.

CUDA Compatibility

CUDA is used for NVFBC capture.

Tip: See [CUDA GPUS](#) to cross reference Compute Capability to your GPU.

Package	CUDA Version	Min Driver	CUDA Compute Capabilities
sunshine.AppImage	11.8.0	450.80.02	35;50;52;60;61;62;70;75;80;86;90
sunshine.pkg.tar.zst	11.8.0	450.80.02	35;50;52;60;61;62;70;75;80;86;90
sunshine_{arch}.flatpak	12.0.0	525.60.13	50;52;60;61;62;70;75;80;86;90
sunshine-debian-bookworm-{arch}.deb	12.0.0	525.60.13	50;52;60;61;62;70;75;80;86;90
sunshine-debian-bullseye-{arch}.deb	11.8.0	450.80.02	35;50;52;60;61;62;70;75;80;86;90
sunshine-fedora-38-{arch}.rpm	12.4.0	525.60.13	50;52;60;61;62;70;75;80;86;90
sunshine-fedora-39-{arch}.rpm	12.4.0	525.60.13	50;52;60;61;62;70;75;80;86;90
sunshine-ubuntu-20.04-{arch}.deb	11.8.0	450.80.02	35;50;52;60;61;62;70;75;80;86;90
sunshine-ubuntu-22.04-{arch}.deb	11.8.0	450.80.02	35;50;52;60;61;62;70;75;80;86;90

Caution: Use distro-specific packages instead of the AppImage if they are available.

According to AppImageLint the supported distro matrix of the AppImage is below.

- ✓ Debian bullseye
- ✓ Debian bookworm
- ✓ Debian trixie
- ✗ Debian sid
- ✓ Ubuntu mantic
- ✓ Ubuntu lunar
- ✓ Ubuntu jammy
- ✓ Ubuntu focal
- ✗ Ubuntu bionic
- ✗ Ubuntu xenial
- ✗ Ubuntu trusty
- ✗ CentOS 7

1. Download sunshine.AppImage to your home directory.

```
cd ~
wget https://github.com/LizardByte/Sunshine/releases/latest/download/sunshine.
↪AppImage
```

2. Open terminal and run the following code.

```
./sunshine.AppImage --install
```

Start:

```
./sunshine.AppImage --install && ./sunshine.AppImage
```

Uninstall:

```
./sunshine.AppImage --remove
```

1. Open terminal and run the following code.

```
wget https://github.com/LizardByte/Sunshine/releases/latest/download/sunshine.pkg.
↪tar.zst
pacman -U --noconfirm sunshine.pkg.tar.zst
```

Uninstall:

```
pacman -R sunshine
```

1. Download `sunshine-{distro}-{distro-version}-{arch}.deb` and run the following code.

```
sudo apt install -f ./sunshine-{distro}-{distro-version}-{arch}.deb
```

Note: The `{distro-version}` is the version of the distro we built the package on. The `{arch}` is the architecture of your operating system.

Tip: You can double click the deb file to see details about the package and begin installation.

Uninstall:

```
sudo apt remove sunshine
```

Caution: Use distro-specific packages instead of the Flatpak if they are available.

Important: The instructions provided here are for the version supplied in the [latest release](#), which does not necessarily match the version in the Flathub repository!

1. Install [Flatpak](#) as required.
2. Download `sunshine_{arch}.flatpak` and run the following code.

Note: Be sure to replace `{arch}` with the architecture for your operating system.

System level (recommended)

```
flatpak install --system ./sunshine_{arch}.flatpak
```

User level

```
flatpak install --user ./sunshine_{arch}.flatpak
```

Additional installation (required)

```
flatpak run --command=additional-install.sh dev.lizardbyte.sunshine
```

Start:

X11 and NVFBC capture (X11 Only)

```
flatpak run dev.lizardbyte.sunshine
```

KMS capture (Wayland & X11)

```
sudo -i PULSE_SERVER=unix:${pactl info | awk '/Server String/{print$3}')} \
flatpak run dev.lizardbyte.sunshine
```

Uninstall:

```
flatpak run --command=remove-additional-install.sh dev.lizardbyte.sunshine
flatpak uninstall --delete-data dev.lizardbyte.sunshine
```

1. Add *rpmfusion* repositories by running the following code.

```
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
```

2. Download `sunshine-{distro}-{distro-version}-{arch}.rpm` and run the following code.

```
sudo dnf install ./sunshine-{distro}-{distro-version}-{arch}.rpm
```

Note: The `{distro-version}` is the version of the distro we built the package on. The `{arch}` is the architecture of your operating system.

Tip: You can double click the rpm file to see details about the package and begin installation.

Uninstall:

```
sudo dnf remove sunshine
```

The *deb*, *rpm*, *zst*, *Flatpak* and *AppImage* packages should handle these steps automatically. Third party packages may not.

Sunshine needs access to *uinput* to create mouse and gamepad events.

1. **Create and reload *udev* rules for *uinput*.**

```
echo 'KERNEL=="uinput", SUBSYSTEM=="misc", OPTIONS+="static_node=uinput", TAG+=
↪"uaccess"' | \
sudo tee /etc/udev/rules.d/60-sunshine.rules
sudo udevadm control --reload-rules
sudo udevadm trigger
sudo modprobe uinput
```

2. **Enable permissions for KMS capture.**

Warning: Capture of most Wayland-based desktop environments will fail unless this step is performed.

Note: `cap_sys_admin` may as well be `root`, except you don't need to be `root` to run it. It is necessary to allow Sunshine to use KMS capture.

Enable

```
sudo setcap cap_sys_admin+p $(readlink -f $(which sunshine))
```

Disable (for Xorg/X11 only)

```
sudo setcap -r $(readlink -f $(which sunshine))
```

3. Optionally, configure autostart service

- filename: `~/.config/systemd/user/sunshine.service`
- contents:

```
[Unit]
Description=Sunshine self-hosted game stream host for Moonlight.
StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
ExecStart=<see table>
Restart=on-failure
RestartSec=5s
#Flatpak Only
#ExecStop=flatpak kill dev.lizardbyte.sunshine

[Install]
WantedBy=graphical-session.target
```

package	ExecStart	Auto Configured
aur	/usr/bin/sunshine	✓
deb	/usr/bin/sunshine	✓
rpm	/usr/bin/sunshine	✓
AppImage	~/sunshine.AppImage	✓
Flatpak	flatpak run dev.lizardbyte.sunshine	✓

Start once

```
systemctl --user start sunshine
```

Start on boot

```
systemctl --user enable sunshine
```

4. Reboot

```
sudo reboot now
```

Important: Sunshine on macOS is experimental. Gamepads do not work.

1. Install [Homebrew](#)
2. Update the Homebrew sources and install Sunshine.

```
brew tap LizardByte/homebrew
brew install sunshine
```

1. Install [MacPorts](#)
2. Update the Macports sources.

```
sudo nano /opt/local/etc/macports/sources.conf
```

Add this line, replacing your username, below the line that starts with `rsync`.

`file:///Users/<username>/ports`

Ctrl+x, then Y to exit and save changes.

3. Download and install by running the following code.

```
mkdir -p ~/ports/multimedia/sunshine
cd ~/ports/multimedia/sunshine
curl -O https://github.com/LizardByte/Sunshine/releases/latest/download/Portfile
cd ~/ports
portindex
sudo port install sunshine
```

4. The first time you start Sunshine, you will be asked to grant access to screen recording and your microphone.
5. Optionally, install service

```
sudo port load Sunshine
```

Uninstall:

```
sudo port uninstall sunshine
```

Sunshine can only access microphones on macOS due to system limitations. To stream system audio use [Soundflower](#) or [BlackHole](#).

Note: Command Keys are not forwarded by Moonlight. Right Option-Key is mapped to CMD-Key.

Caution: Gamepads are not currently supported.

1. Download and install `sunshine-windows-installer.exe`

Attention: You should carefully select or unselect the options you want to install. Do not blindly install or enable features.

To uninstall, find Sunshine in the list [here](#) and select “Uninstall” from the overflow menu. Different versions of Windows may provide slightly different steps for uninstall.

Warning: By using this package instead of the installer, performance will be reduced. This package is not recommended for most users. No support will be provided!

1. Download and extract `sunshine-windows-portable.zip`
2. Open command prompt as administrator
3. Firewall rules

Install:

```
cd /d {path to extracted directory}
scripts/add-firewall-rule.bat
```

Uninstall:

```
cd /d {path to extracted directory}
scripts/delete-firewall-rule.bat
```

4. Virtual Gamepad Support

Install:

```
cd /d {path to extracted directory}
scripts/install-gamepad.bat
```

Uninstall:

```
cd /d {path to extracted directory}
scripts/uninstall-gamepad.bat
```

5. Windows service

Install:

```
cd /d {path to extracted directory}
scripts/install-service.bat
scripts/autostart-service.bat
```

Uninstall:

```
cd /d {path to extracted directory}
scripts/uninstall-service.bat
```

To uninstall, delete the extracted directory which contains the `sunshine.exe` file.

2.3 Usage

1. If Sunshine is not installed/running as a service, then start sunshine with the following command, unless a start command is listed in the specified package *install* instructions above.

Note: A service is a process that runs in the background. This is the default when installing Sunshine from the Windows installer. Running multiple instances of Sunshine is not advised.

Basic usage

```
sunshine
```

Specify config file

```
sunshine <directory of conf file>/sunshine.conf
```

Note: You do not need to specify a config file. If no config file is entered the default location will be used.

Attention: The configuration file specified will be created if it doesn't exist.

Start Sunshine over SSH (Linux/X11)

Assuming you are already logged into the host, you can use this command

```
ssh <user>@<ip_address> 'export DISPLAY=:0; sunshine'
```

If you are logged into the host with only a tty (teletypewriter), you can use `startx` to start the X server prior to executing `sunshine`. You may need to add `sleep` between `startx` and `sunshine` to allow more time for the display to be ready.

```
ssh <user>@<ip_address> 'startx &; export DISPLAY=:0; sunshine'
```

Tip: You could also utilize the `~/.bash_profile` or `~/.bashrc` files to setup the `DISPLAY` variable.

See also:

See [Remote SSH Headless Setup](#) on how to setup a headless streaming server without autologin and dummy plugs (X11 + NVidia GPUs)

2. Configure Sunshine in the web ui

The web ui is available on <https://localhost:47990> by default. You may replace *localhost* with your internal ip address.

Attention: Ignore any warning given by your browser about “insecure website”. This is due to the SSL certificate being self signed.

Caution: If running for the first time, make sure to note the username and password that you created.

1. Add games and applications.
2. Adjust any configuration settings as needed.

3. In Moonlight, you may need to add the PC manually.

4. When Moonlight requests for you insert the pin:

- Login to the web ui
- Go to “PIN” in the Navbar

- Type in your PIN and press Enter, you should get a Success Message
- In Moonlight, select one of the Applications listed

2.4 Network

The Sunshine user interface will be available on port 47990 by default.

Warning: Exposing ports to the internet can be dangerous. Do this at your own risk.

2.5 Arguments

To get a list of available arguments run the following:

```
sunshine --help
```

```
./sunshine.AppImage --help
```

```
flatpak run --command=sunshine dev.lizardbyte.Sunshine --help
```

2.6 Shortcuts

All shortcuts start with CTRL + ALT + SHIFT, just like Moonlight

- CTRL + ALT + SHIFT + N - Hide/Unhide the cursor (This may be useful for Remote Desktop Mode for Moonlight)
- CTRL + ALT + SHIFT + F1/F12 - Switch to different monitor for Streaming

2.7 Application List

- Applications should be configured via the web UI.
- A basic understanding of working directories and commands is required.
- You can use Environment variables in place of values
- \$(HOME) will be replaced by the value of \$HOME
- \$\$ will be replaced by \$, e.g. \$\$ (HOME) will become \$(HOME)
- env - Adds or overwrites Environment variables for the commands/applications run by Sunshine
- "Variable name": "Variable value"
- apps - The list of applications
- Advanced users may want to edit the application list manually. The format is json.
- **Example json application:**

```
{
  "cmd": "command to open app",
  "detached": [
    "some-command",
    "another-command"
  ],
  "image-path": "/full-path/to/png-image",
  "name": "An App",
  "output": "/full-path/to/command-log-file",
  "prep-cmd": [
    {
      "do": "some-command",
      "undo": "undo-that-command"
    }
  ],
  "working-dir": "/full-path/to/working-directory"
}
```

- cmd - The main application
 - detached - A list of commands to be run and forgotten about
 - * If not specified, a process is started that sleeps indefinitely
 - image-path - The full path to the cover art image to use.
 - name - The name of the application/game
 - output - The file where the output of the command is stored
 - auto-detach - Specifies whether the app should be treated as detached if it exits quickly
 - wait-all - Specifies whether to wait for all processes to terminate rather than just the initial process
 - exit-timeout - Specifies how long to wait in seconds for the process to gracefully exit (default: 5 seconds)
 - prep-cmd - A list of commands to be run before/after the application
 - * If any of the prep-commands fail, starting the application is aborted
 - * do - Run before the application
 - If it fails, all undo commands of the previously succeeded do commands are run
 - * undo - Run after the application has terminated
 - Failures of undo commands are ignored
 - working-dir - The working directory to use. If not specified, Sunshine will use the application directory.
- For more examples see [app examples](#).

2.8 Considerations

- On Windows, Sunshine uses the Desktop Duplication API which only supports capturing from the GPU used for display. If you want to capture and encode on the eGPU, connect a display or HDMI dummy display dongle to it and run the games on that display.
- When an application is started, if there is an application already running, it will be terminated.
- When the application has been shutdown, the stream shuts down as well.
 - For example, if you attempt to run `steam` as a `cmd` instead of `detached` the stream will immediately fail. This is due to the method in which the steam process is executed. Other applications may behave similarly.
 - This does not apply to detached applications.
- The “Desktop” app works the same as any other application except it has no commands. It does not start an application, instead it simply starts a stream. If you removed it and would like to get it back, just add a new application with the name “Desktop” and “desktop.png” as the image path.
- For the Linux flatpak you must prepend commands with `flatpak-spawn --host`.

2.9 HDR Support

Streaming HDR content is officially supported on Windows hosts and experimentally supported for Linux hosts.

- General HDR support information and requirements:
 - HDR must be activated in the host OS, which may require an HDR-capable display or EDID emulator dongle connected to your host PC.
 - You must also enable the HDR option in your Moonlight client settings, otherwise the stream will be SDR (and probably overexposed if your host is HDR).
 - A good HDR experience relies on proper HDR display calibration both in the OS and in game. HDR calibration can differ significantly between client and host displays.
 - You may also need to tune the brightness slider or HDR calibration options in game to the different HDR brightness capabilities of your client’s display.
 - Some GPUs video encoders can produce lower image quality or encoding performance when streaming in HDR compared to SDR.
- Additional information:
- HDR streaming is supported for Intel, AMD, and NVIDIA GPUs that support encoding HEVC Main 10 or AV1 10-bit profiles.
- We recommend calibrating the display by streaming the Windows HDR Calibration app to your client device and saving an HDR calibration profile to use while streaming.
- Older games that use NVIDIA-specific NVAPI HDR rather than native Windows HDR support may not display properly in HDR.
- HDR streaming is supported for Intel and AMD GPUs that support encoding HEVC Main 10 or AV1 10-bit profiles using VA-API.
- The KMS capture backend is required for HDR capture. Other capture methods, like NvFBC or X11, do not support HDR.
- You will need a desktop environment with a compositor that supports HDR rendering, such as Gamescope or KDE Plasma 6.

See also:

Arch wiki on HDR Support for Linux and Reddit Guide for HDR Support for AMD GPUs

2.10 Tutorials and Guides

Tutorial videos are available [here](#).

Guides are available [here](#).

Community!

Tutorials and Guides are community generated. Want to contribute? Reach out to us on our discord server.

3.1 Important note

Starting with v0.18.0, tag names have changed. You may no longer use `latest`, `master`, `vX.X.X`.

3.2 Build your own containers

This image provides a method for you to easily use the latest Sunshine release in your own docker projects. It is not intended to use as a standalone container at this point, and should be considered experimental.

```
ARG SUNSHINE_VERSION=latest
ARG SUNSHINE_OS=ubuntu-22.04
FROM lizardbyte/sunshine:${SUNSHINE_VERSION}-${SUNSHINE_OS}

# install Steam, Wayland, etc.

ENTRYPOINT steam && sunshine
```

3.2.1 SUNSHINE_VERSION

- `latest`, `master`, `vX.X.X`
- `nightly`
- commit hash

3.2.2 SUNSHINE_OS

Sunshine images are available with the following tag suffixes, based on their respective base images.

- `archlinux`
- `debian-bullseye`
- `fedora-36`
- `fedora-37`
- `ubuntu-20.04`
- `ubuntu-22.04`

3.2.3 Tags

You must combine the `SUNSHINE_VERSION` and `SUNSHINE_OS` to determine the tag to pull. The format should be `<SUNSHINE_VERSION>-<SUNSHINE_OS>`. For example, `latest-ubuntu-22.04`.

See all our available tags on [docker hub](#) or [ghcr](#) for more info.

3.3 Where used

This is a list of docker projects using Sunshine. Something missing? Let us know about it!

- [Games on Whales](#)

3.4 Port and Volume mappings

Examples are below of the required mappings. The configuration file will be saved to `/config` in the container.

3.4.1 Using docker run

Create and run the container (substitute your `<values>`):

```
docker run -d \
  --device /dev/dri/ \
  --name=<image_name> \
  --restart=unless-stopped \
  -e PUID=<uid> \
  -e PGID=<gid> \
  -e TZ=<timezone> \
  -v <path to data>:/config \
  -p 47984-47990:47984-47990/tcp \
  -p 48010:48010 \
  -p 47998-48000:47998-48000/udp \
  <image>
```

3.4.2 Using docker-compose

Create a `docker-compose.yml` file with the following contents (substitute your `<values>`):

```
version: '3'
services:
  <image_name>:
    image: <image>
    container_name: sunshine
    restart: unless-stopped
    volumes:
      - <path to data>:/config
    environment:
      - PUID=<uid>
      - PGID=<gid>
```

(continues on next page)

(continued from previous page)

```

- TZ=<timezone>
ports:
- "47984-47990:47984-47990/tcp"
- "48010:48010"
- "47998-48000:47998-48000/udp"

```

3.4.3 Using podman run

Create and run the container (substitute your <values>):

```

podman run -d \
--device /dev/dri/ \
--name=<image_name> \
--restart=unless-stopped \
--userns=keep-id \
-e PUID=<uid> \
-e PGID=<gid> \
-e TZ=<timezone> \
-v <path to data>:/config \
-p 47984-47990:47984-47990/tcp \
-p 48010:48010 \
-p 47998-48000:47998-48000/udp \
<image>

```

3.4.4 Parameters

You must substitute the <values> with your own settings.

Parameters are split into two halves separated by a colon. The left side represents the host and the right side the container.

Example: `-p external:internal` - This shows the port mapping from internal to external of the container. Therefore `-p 47990:47990` would expose port 47990 from inside the container to be accessible from the host's IP on port 47990 (e.g. `http://<host_ip>:47990`). The internal port must be 47990, but the external port may be changed (e.g. `-p 8080:47990`). All the ports listed in the `docker run` and `docker-compose` examples are required.

Parameter	Function	Example Value	Required
<code>-p <port>:47990</code>	Web UI Port	47990	True
<code>-v <path to data>:/config</code>	Volume mapping	/home/sunshine	True
<code>-e PUID=<uid></code>	User ID	1001	False
<code>-e PGID=<gid></code>	Group ID	1001	False
<code>-e TZ=<timezone></code>	Lookup TZ value	America/New_York	False

User / Group Identifiers:

When using data volumes (-v flags) permissions issues can arise between the host OS and the container. To avoid this issue you can specify the user PUID and group PGID. Ensure the data volume directory on the host is owned by the same user you specify.

In this instance PUID=1001 and PGID=1001. To find yours use id user as below:

```
$ id dockeruser
uid=1001(dockeruser) gid=1001(dockergroup) groups=1001(dockergroup)
```

If you want to change the PUID or PGID after the image has been built, it will require rebuilding the image.

3.5 Supported Architectures

Specifying `lizardbyte/sunshine:latest-<SUNSHINE_OS>` or `ghcr.io/lizardbyte/sunshine:latest-<SUNSHINE_OS>` should retrieve the correct image for your architecture.

The architectures supported by these images are shown in the table below.

tag suffix	amd64/x86_64	arm64/aarch64
archlinux		
debian-bookworm		
debian-bullseye		
fedora-38		
fedora-39		
ubuntu-20.04		
ubuntu-22.04		

THIRD PARTY PACKAGES

Danger: These packages are not maintained by LizardByte. Use at your own risk.

4.1 AUR

4.2 Chocolatey

4.3 nixpkgs

4.4 Scoop

4.5 Solus

GUIDES

Collection of guides written by the community!

5.1 App Examples

Since not all applications behave the same, we decided to create some examples to help you get started adding games and applications to Sunshine.

Attention: Throughout these examples, any fields not shown are left blank. You can enhance your experience by adding an image or a log file (via the `Output` field).

Note: When a working directory is not specified, it defaults to the folder where the target application resides.

5.1.1 Common Examples

Desktop

Field	Value
Application Name	Desktop
Image	desktop.png

Steam Big Picture

Note: Steam is launched as a detached command because Steam starts with a process that self updates itself and the original process is killed.

Application Name	Steam Big Picture
Detached Commands	setsid steam steam://open/bigpicture
Image	steam.png

Application Name	Steam Big Picture
Detached Commands	open steam://open/bigpicture
Image	steam.png

Application Name	Steam Big Picture
Command	steam://open/bigpicture
Image	steam.png

Epic Game Store game

Note: Using URI method will be the most consistent between various games.

URI (Epic)

Applica- tion Name	Surviving Mars
Command	com.epicgames.launcher://apps/d759128018124dcabb1fbee9bb28e178%3A20729b9176c241f0b617c5723e7 action=launch&silent=true

Binary (Epic w/ working directory)

Application Name	Surviving Mars
Command	MarsEpic.exe
Working Directory	C:\Program Files\Epic Games\SurvivingMars

Binary (Epic w/o working directory)

Application Name	Surviving Mars
Command	"C:\Program Files\Epic Games\SurvivingMars\MarsEpic.exe"

Steam game

Note: Using URI method will be the most consistent between various games.

URI (Steam)

Application Name	Surviving Mars
Detached Commands	setsid steam steam://rungameid/464920

Application Name	Surviving Mars
Detached Commands	open steam://rungameid/464920

Application Name	Surviving Mars
Command	steam://rungameid/464920

Binary (Steam w/ working directory)

Application Name	Surviving Mars
Command	MarsSteam
Working Directory	~/.steam/steam/SteamApps/common/Survivng Mars

Application Name	Surviving Mars
Command	MarsSteam
Working Directory	~/.steam/steam/SteamApps/common/Survivng Mars

Application Name	Surviving Mars
Command	MarsSteam.exe
Working Directory	C:\Program Files (x86)\Steam\steamapps\common\Surviving Mars

Binary (Steam w/o working directory)

Application Name	Surviving Mars
Command	~/.steam/steam/SteamApps/common/Survivng Mars/MarsSteam

Application Name	Surviving Mars
Command	~/.steam/steam/SteamApps/common/Survivng Mars/MarsSteam

Application Name	Surviving Mars
Command	"C:\Program Files (x86)\Steam\steamapps\common\Surviving Mars\MarsSteam.exe"

5.1.2 Prep Commands

Changing Resolution and Refresh Rate

Command	Do: sh -c "xrandr --output HDMI-1 --mode \"\${SUNSHINE_CLIENT_WIDTH}x\${SUNSHINE_CLIENT_HEIGHT} --rate \${SUNSHINE_CLIENT_FPS}\"
Preparations	Undo: xrandr --output HDMI-1 --mode 3840x2160 --rate 120

Hint: The above only works if the xrandr mode already exists. You will need to create new modes to stream to macOS and iOS devices, since they use non standard resolutions.

You can update the Do command to this:

```
bash -c "${HOME}/scripts/set-custom-res.sh \"${SUNSHINE_CLIENT_WIDTH}\" \"${SUNSHINE_CLIENT_HEIGHT}\" \"${SUNSHINE_CLIENT_FPS}\""
```

The set-custom-res.sh will have this content:

```
#!/bin/bash

# Get params and set any defaults
width=${1:-1920}
height=${2:-1080}
refresh_rate=${3:-60}

# You may need to adjust the scaling differently so the UI/text isn't too small / big
scale=${4:-0.55}

# Get the name of the active display
display_output=$(xrandr | grep " connected" | awk '{ print $1 }')

# Get the modeline info from the 2nd row in the cvt output
modeline=$(cvt $width $height $refresh_rate | awk 'FNR == 2')
xrandr_mode_str=${modeline//Modeline "\"\ "/}
mode_alias="${width}x${height}"

echo "xrandr setting new mode ${mode_alias} ${xrandr_mode_str}"
xrandr --newmode ${mode_alias} ${xrandr_mode_str}
xrandr --addmode ${display_output} ${mode_alias}

# Reset scaling
xrandr --output ${display_output} --scale 1

# Apply new xrandr mode
xrandr --output ${display_output} --primary --mode ${mode_alias} --pos 0x0 --rotate_
↪normal --scale ${scale}

# Optional reset your wallpaper to fit to new resolution
# xwallpaper --zoom /path/to/wallpaper.png
```

Command Preparations	Do: sh -c "wlr-xrandr --output HDMI-1 --mode \"\${SUNSHINE_CLIENT_WIDTH}x\${SUNSHINE_CLIENT_HEIGHT}\" Undo: wlr-xrandr --output HDMI-1 --mode 3840x2160@120Hz
----------------------	--

Command Preparations	Do: sh -c "kscreen-doctor output.HDMI-A-1.mode.\${SUNSHINE_CLIENT_WIDTH}x\${SUNSHINE_CLIENT_HEIGHT} Undo: kscreen-doctor output.HDMI-A-1.mode.3840x2160@120
----------------------	--

Command Preparations	Do: sh -c "\${HOME}/scripts/set-custom-res.sh \${SUNSHINE_CLIENT_WIDTH} \${SUNSHINE_CLIENT_HEIGHT}" Undo: sh -c "\${HOME}/scripts/set-custom-res.sh 3840 2160"
----------------------	---

The `set-custom-res.sh` will have this content:

```
#!/bin/bash

# Get params and set any defaults
width=${1:-1920}
height=${2:-1080}
output=${3:-HDMI-1}
nvidia-settings -a CurrentMetaMode="${output}: nvidia-auto-select { ViewPortIn=$
↪ ${width}x${height}, ViewPortOut=${width}x${height}+0+0 }"
```

Note: This example uses the *displayplacer* tool to change the resolution. This tool can be installed following instructions in their [GitHub repository](#).

Command Preparations	Do: displayplacer "id:<screenId> res:1920x1080 hz:60 scaling:on origin:(0,0) degree:0" Undo: displayplacer "id:<screenId> res:3840x2160 hz:120 scaling:on origin:(0,0) degree:0"
----------------------	---

Note: This example uses the *QRes* tool to change the resolution and refresh rate. This tool can be downloaded from their [SourceForge repository](#).

Command Preparations	Do: cmd /C FullPath\qres.exe /x:%SUNSHINE_CLIENT_WIDTH% /y:%SUNSHINE_CLIENT_HEIGHT% /r:%SUNSHINE_CLIENT_FPS% Undo: cmd /C FullPath\qres.exe /x:3840 /y:2160 /r:120
----------------------	---

5.1.3 Additional Considerations

Attention: Because Flatpak packages run in a sandboxed environment and do not normally have access to the host, the Flatpak of Sunshine requires commands to be prefixed with `flatpak-spawn --host`.

Elevating Commands (Windows)

If you've installed Sunshine as a service (default), you can specify if a command should be elevated with administrative privileges. Simply enable the elevated option in the WEB UI, or add it to the JSON configuration. This is an option for both `prep-cmd` and regular commands and will launch the process with the current user without a UAC prompt.

Note: It is important to write the values “true” and “false” as string values, not as the typical true/false values in most JSON.

Example

```
{
  "name": "Game With AntiCheat that Requires Admin",
  "output": "",
  "cmd": "ping 127.0.0.1",
  "exclude-global-prep-cmd": "false",
  "elevated": "true",
  "prep-cmd": [
    {
      "do": "powershell.exe -command \"Start-Streaming\"",
      "undo": "powershell.exe -command \"Stop-Streaming\"",
      "elevated": "false"
    }
  ],
  "image-path": ""
}
```

5.2 Linux

Collection of Sunshine Linux host guides.

5.2.1 Remote SSH Headless Setup

Table 1: Remote SSH Headless Setup

Author	Eric Dong
Difficulty	Intermediate

This is a guide to setup remote SSH into host to startup X server and sunshine without physical login and dummy plug. The virtual display is accelerated by the NVidia GPU using the TwinView configuration.

Attention: This guide is specific for Xorg and NVidia GPUs. I start the X server using the `startx` command. I also only tested this on an Artix runit init system on LAN. I didn't have to do anything special with pulseaudio (pipewire untested).

Keep your monitors plugged in until the *Checkpoint* step

Tip: Prior to editing any system configurations, you should make a copy of the original file. This will allow you to use it for reference or revert your changes easily.

The Big Picture

Once you are done, you will need to perform these 3 steps:

1. Turn on the host machine
2. Start sunshine on remote host with a script that:
 - Edits permissions of `/dev/uinput` (added sudo config to execute script with no password prompt)
 - Starts X server with `startx` on virtual display
 - Starts Sunshine
3. Startup Moonlight on the client of interest and connect to host

Hint: As an alternative to SSH...

Step 2 can be replaced with autologin and starting sunshine as a service or putting `sunshine &` in your `.xinitrc` file if you start your X server with `startx`. In this case, the workaround for `/dev/uinput` permissions is not needed because the udev rule would be triggered for “physical” login. See *Linux Setup*. I personally think autologin compromises the security of the PC, so I went with the remote SSH route. I use the PC more than for gaming, so I don't need a virtual display everytime I turn on the PC (E.g running updates, config changes, file/media server).

First we will setup the host and then the SSH Client (Which may not be the same as the machine running the moonlight client)

Host Setup

We will be setting up:

1. *Static IP Setup*
2. *SSH Server Setup*
3. *Virtual Display Setup*
4. *Uinput Permissions Workaround*
5. *Stream Launcher Script*

Static IP Setup

Setup static IP Address for host. For LAN connections you can use DHCP reservation within your assigned range. e.g. 192.168.x.x. This will allow you to ssh to the host consistently, so the assigned IP address does not change. It is preferred to set this through your router config.

SSH Server Setup

Note: Most distros have OpenSSH already installed. If it is not present, install OpenSSH using your package manager.

```
sudo apt update
sudo apt install openssh-server
```

```
sudo pacman -S openssh
# Install openssh-<other_init> if you are not using SystemD
# e.g. sudo pacman -S openssh-runit
```

```
sudo apk update
sudo apk add openssh
```

CentOS/RHEL 7

```
sudo yum install openssh-server
```

CentOS/Fedora/RHEL 8

```
sudo dnf install openssh-server
```

Next make sure the OpenSSH daemon is enabled to run when the system starts.

```
sudo systemctl enable sshd.service
sudo systemctl start sshd.service # Starts the service now
sudo systemctl status sshd.service # See if the service is running
```

```
sudo ln -s /etc/runit/sv/sshd /run/runit/service # Enables the OpenSSH daemon to run
↳ when system starts
sudo sv start sshd # Starts the service now
sudo sv status sshd # See if the service is running
```

```
rc-update add sshd # Enables service
rc-status # List services to verify sshd is enabled
rc-service sshd start # Starts the service now
```

Disabling PAM in sshd

I noticed when the ssh session is disconnected for any reason, pulseaudio would disconnect. This is due to PAM handling sessions. When running `dmesg`, I noticed `elogind` would say removed user session. In this [Gentoo Forums post](#), someone had a similar issue. Starting the X server in the background and exiting out of the console would cause your session to be removed.

Caution: According to this [article](#) disabling PAM increases security, but reduces certain functionality in terms of session handling. *Do so at your own risk!*

Edit the `sshd_config` file with the following to disable PAM.

```
usePAM no
```

After making changes to the `sshd_config`, restart the `sshd` service for changes to take effect.

Tip: Run the command to check the `ssh` configuration prior to restarting the `sshd` service.

```
sudo sshd -t -f /etc/ssh/sshd_config
```

An incorrect configuration will prevent the `sshd` service from starting, which might mean losing SSH access to the server.

```
sudo systemctl restart sshd.service
```

```
sudo sv restart sshd
```

```
sudo rc-service sshd restart
```

Virtual Display Setup

As an alternative to a dummy dongle, you can use this config to create a virtual display.

Important: This is only available for NVidia GPUs using Xorg.

Hint: Use `xrandr` to see name of your active display output. Usually it starts with DP or HDMI. For me, it is DP-0. Put this name for the `ConnectedMonitor` option under the `Device` section.

```
xrandr | grep " connected" | awk '{ print $1 }'
```

```
Section "ServerLayout"
    Identifier "TwinLayout"
    Screen 0 "metaScreen" 0 0
EndSection

Section "Monitor"
    Identifier "Monitor0"
    Option "Enable" "true"
EndSection

Section "Device"
    Identifier "Card0"
    Driver "nvidia"
```

(continues on next page)

(continued from previous page)

```
VendorName "NVIDIA Corporation"
Option "MetaModes" "1920x1080"
Option "ConnectedMonitor" "DP-0"
Option "ModeValidation" "NoDFPNativeResolutionCheck,NoVirtualSizeCheck,
↪NoMaxPclkCheck,NoHorizSyncCheck,NoVertRefreshCheck,NoWidthAlignmentCheck"
EndSection

Section "Screen"
    Identifier "metaScreen"
    Device "Card0"
    Monitor "Monitor0"
    DefaultDepth 24
    Option "TwinView" "True"
    SubSection "Display"
        Modes "1920x1080"
    EndSubSection
EndSection
```

Note: The `ConnectedMonitor` tricks the GPU into thinking a monitor is connected, even if there is none actually connected! This allows a virtual display to be created that is accelerated with your GPU! The `ModeValidation` option disables valid resolution checks, so you can choose any resolution on the host!

References

- [issue comment on virtual-display-linux](#)
 - [Nvidia Documentation on Configuring TwinView](#)
 - [Arch Wiki Nvidia#TwinView](#)
 - [Unix Stack Exchange - How to add virtual display monitor with Nvidia proprietary driver](#)
-

Uinput Permissions Workaround

Steps

We can use `chown` to change the permissions from a script. Since this requires `sudo`, we will need to update the `sudo` configuration to execute this without being prompted for a password.

1. Create a `sunshine-setup.sh` script to update permissions on `/dev/uinput`. Since we aren't logged into the host, the `udev` rule doesn't apply.
2. Update user `sudo` configuration `/etc/sudoers.d/<user>` to allow the `sunshine-setup.sh` script to be executed with `sudo`.

Note: After I setup the *udev rule* to get access to `/dev/uinput`, I noticed when I `ssh`d into the host without physical login, the `ACL` permissions on `/dev/uinput` were not changed. So I asked [reddit](#). I discovered that `SSH` sessions are not the same as a physical login. I suppose it's not possible for `SSH` to trigger a `udev` rule or create a physical login session.

Setup Script

This script will take care of any preconditions prior to starting up `sunshine`.

Run the following to create a script named something like `sunshine-setup.sh`:

```
echo "chown $(id -un):$(id -gn) /dev/uinput" > sunshine-setup.sh &&\
chmod +x sunshine-setup.sh
```

(Optional) To Ensure ethernet is being used for streaming, you can block WiFi with `rkill`.

Run this command to append the `rkill` block command to the script:

```
echo "rkill block $(rkill list | grep "Wireless LAN" \
| sed 's/^\([[[:digit:]]\)\.*/\1/')" >> sunshine-setup.sh
```

Sudo Configuration

We will manually change the permissions of `/dev/uinput` using `chown`. You need to use `sudo` to make this change, so add/update the entry in `/etc/sudoers.d/${USER}`

Danger: Do so at your own risk! It is more secure to give `sudo` and no password prompt to a single script, than a generic executable like `chown`.

Warning: Be very careful of messing this config up. If you make a typo, *YOU LOSE THE ABILITY TO USE SUDO*. Fortunately, your system is not borked, you will need to login as root to fix the config. You may want to setup a backup user / SSH into the host as root to fix the config if this happens. Otherwise you will need to plug your machine back into a monitor and login as root to fix this. To enable root login over SSH edit your SSHD config, and add `PermitRootLogin yes`, and restart the SSH server.

1. First make a backup of your `/etc/sudoers.d/${USER}` file.

```
sudo cp /etc/sudoers.d/${USER} /etc/sudoers.d/${USER}.backup
```

2. `cd` to the parent dir of the `sunshine-setup.sh` script.
3. Execute the following to update your sudoer config file.

```
echo "${USER} ALL=(ALL:ALL) ALL, NOPASSWD: $(pwd)/sunshine-setup.sh" \
| sudo tee /etc/sudoers.d/${USER}
```

These changes allow the script to use `sudo` without being prompted with a password.

e.g. `sudo $(pwd)/sunshine-setup.sh`

Stream Launcher Script

This is the main entrypoint script that will run the `sunshine-setup.sh` script, start up X server, and Sunshine. The client will call this script that runs on the host via `ssh`.

Sunshine Startup Script

This guide will refer to this script as `~/scripts/sunshine.sh`. The setup script will be referred as `~/scripts/sunshine-setup.sh`

```
#!/bin/bash
```

(continues on next page)

(continued from previous page)

```
export DISPLAY=:0

# Check existing X server
ps -e | grep X >/dev/null
[[ ${?} -ne 0 ]] && {
    echo "Starting X server"
    startx &>/dev/null &
    [[ ${?} -eq 0 ]] && {
        echo "X server started successfully"
    } || echo "X server failed to start"
} || echo "X server already running"

# Check if sunshine is already running
ps -e | grep -e .*sunshine$ >/dev/null
[[ ${?} -ne 0 ]] && {
    sudo ~/scripts/sunshine-setup.sh
    echo "Starting Sunshine!"
    sunshine > /dev/null &
    [[ ${?} -eq 0 ]] && {
        echo "Sunshine started successfully"
    } || echo "Sunshine failed to start"
} || echo "Sunshine is already running"

# Add any other Programs that you want to startup automatically
# e.g.
# steam &> /dev/null &
# firefox &> /dev/null &
# kdeconnect-app &> /dev/null &
```

SSH Client Setup

We will be setting up:

1. *SSH Key Authentication Setup*
2. *SSH Client Script (Optional)*

SSH Key Authentication Setup

1. Setup your SSH keys with `ssh-keygen` and use `ssh-copy-id` to authorize remote login to your host. Run `ssh <user>@<ip_address>` to login to your host. SSH keys automate login so you don't need to input your password!
2. Optionally setup a `~/.ssh/config` file to simplify the `ssh` command

```
Host <some_alias>
    Hostname <ip_address>
    User <username>
    IdentityFile ~/.ssh/<your_private_key>
```

Now you can use `ssh <some_alias>. ssh <some_alias> <commands/script>` will execute the command or script on the remote host.

Checkpoint

As a sanity check, let's make sure your setup is working so far!

Test Steps

With your monitor still plugged into your Sunshine host PC:

1. `ssh <alias>`
2. `~/scripts/sunshine.sh`
3. `nvidia-smi`

You should see the sunshine and Xorg processing running:

```
nvidia-smi
```

Output:

```
+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2
|
+-----+
| GPU  Name                Persistence-M | Bus-Id        Disp.A | Volatile Uncorr.
| GPU  Name                Persistence-M | Bus-Id        Disp.A | Volatile Uncorr.
| Fan  Temp   Perf          Pwr:Usage/Cap |      Memory-Usage | GPU-Util  |
| Compute M. |
|
| MIG M. |
+-----+
|  0  NVIDIA GeForce RTX 3070                Off | 00000000:01:00.0  On |
| N/A |
| 30%  46C    P2              45W / 220W |  549MiB /  8192MiB |      2%   |
| Default |
|
| N/A |
+-----+
+-----+
| Processes:
|
| GPU  GI    CI        PID   Type   Process name                      GPU
| Memory |
|      ID    ID
|
|
+-----+
|  0  N/A  N/A       1393    G   /usr/lib/Xorg                      86MiB
```

(continues on next page)

(continued from previous page)

```
|      0   N/A   N/A      1440    C+G    sunshine
↪293MiB |
+-----+
↪-----+
```

4. Check /dev/uinput permissions

```
ls -l /dev/uinput
```

Output:

```
crw----- 1 <user> <primary_group> 10, 223 Aug 29 17:31 /dev/uinput
```

5. Connect to Sunshine host from a moonlight client

Now kill X and sunshine by running `pkill X` on the host, unplug your monitors from your GPU, and repeat steps 1 - 5. You should get the same result. With this setup you don't need to modify the Xorg config regardless if monitors are plugged in or not.

```
pkill X
```

SSH Client Script (Optional)

At this point you have a working setup! For convenience I created this bash script to automate the startup of the X server and Sunshine on the host. This can be run on Unix systems, or on Windows using the `git-bash` or any bash shell.

For Android/iOS you can install Linux emulators, e.g. Userland for Android and ISH for iOS. The neat part is that you can execute one script to launch Sunshine from your phone or tablet!

```
#!/bin/bash

ssh_args="<user>@192.168.X.X" # Or use alias set in ~/.ssh/config

check_ssh(){
  result=1
  # Note this checks infinitely, you could update this to have a max # of retries
  while [[ $result -ne 0 ]]
  do
    echo "checking host..."
    ssh $ssh_args "exit 0" 2>/dev/null
    result=$?
    [[ $result -ne 0 ]] && {
      echo "Failed to ssh to $ssh_args, with exit code $result"
    }
    sleep 3
  done
  echo "Host is ready for streaming!"
}

start_stream(){
  echo "Starting sunshine server on host..."
}
```

(continues on next page)

(continued from previous page)

```
echo "Start moonlight on your client of choice"
# -f runs ssh in the background
ssh -f $ssh_args "~/scripts/sunshine.sh &"
}

check_ssh
start_stream
exit_code=${?}

sleep 3
exit ${exit_code}
```

Next Steps

Congrats you can now stream your desktop headless! When trying this the first time, keep your monitors close by incase something isn't working right.

If you have any feedback and any suggestions, feel free to make a post on Discord!

See also:

Now that you have a virtual display, you may want to automate changing the resolution and refresh rate prior to connecting to an app. See [Changing Resolution and Refresh Rate](#) for more information.

ADVANCED USAGE

Sunshine will work with the default settings for most users. In some cases you may want to configure Sunshine further.

6.1 Performance Tips

In Windows, enabling *Enhanced Sync* in AMD's settings may help reduce the latency by an additional frame. This applies to *amfenc* and *libx264*.

Enabling *Fast Sync* in Nvidia settings may help reduce latency.

6.2 Configuration

The default location for the configuration file is listed below. You can use another location if you choose, by passing in the full configuration file path as the first argument when you start Sunshine.

The default location of the `apps.json` is the same as the configuration file. You can use a custom location by modifying the configuration file.

Default File Location

Value	Description
Docker	/config/
Linux	~/.config/sunshine/
macOS	~/.config/sunshine/
Windows	%ProgramFiles%\Sunshine\config

Example

```
sunshine ~/sunshine_config.conf
```

Although it is recommended to use the configuration UI, it is possible manually configure sunshine by editing the *conf* file in a text editor. Use the examples as reference.

6.3 General

6.3.1 sunshine_name

Description

The name displayed by Moonlight

Default

PC hostname

Example

```
sunshine_name = Sunshine
```

6.3.2 min_log_level

Description

The minimum log level printed to standard out.

Choices

Value	Description
verbose	verbose logging
debug	debug logging
info	info logging
warning	warning logging
error	error logging
fatal	fatal logging
none	no logging

Default

info

Example

```
min_log_level = info
```

6.3.3 channels

Description

Sunshine can support multiple clients streaming simultaneously, at the cost of higher CPU and GPU usage.

Note: All connected clients share control of the same streaming session.

Warning: Some hardware encoders may have limitations that reduce performance with multiple streams.

Default

1

Example

```
channels = 1
```

6.3.4 global_prep_cmd

Description

A list of commands to be run before/after all applications. If any of the prep-commands fail, starting the application is aborted.

Default

```
[]
```

Example

```
global_prep_cmd = [{"do": "nircmd.exe setdisplay 1280 720 32 144", "undo": "nircmd.exe ↵
↵ setdisplay 2560 1440 32 144"}]
```

6.4 Input

6.4.1 controller

Description

Whether to allow controller input from the client.

Example

```
controller = enabled
```

6.4.2 gamepad

Description

The type of gamepad to emulate on the host.

Caution: Applies to Windows only.

Choices

Value	Description
auto	Selected based on information from client
x360	Xbox 360 controller
ds4	DualShock 4 controller (PS4)

Default

```
auto
```

Example

```
gamepad = auto
```

6.4.3 ds4_back_as_touchpad_click

Description

Hint: Only applies when gamepad is set to ds4 manually. Unused in other gamepad modes.

Allow Select/Back inputs to also trigger DS4 touchpad click. Useful for clients looking to emulate touchpad click on Xinput devices.

Default

enabled

Example

```
ds4_back_as_touchpad_click = enabled
```

6.4.4 motion_as_ds4

Description

Hint: Only applies when gamepad is set to auto.

If a client reports that a connected gamepad has motion sensor support, emulate it on the host as a DS4 controller. When disabled, motion sensors will not be taken into account during gamepad type selection.

Default

enabled

Example

```
motion_as_ds4 = enabled
```

6.4.5 touchpad_as_ds4

Description

Hint: Only applies when gamepad is set to auto.

If a client reports that a connected gamepad has a touchpad, emulate it on the host as a DS4 controller. When disabled, touchpad presence will not be taken into account during gamepad type selection.

Default

enabled

Example

```
touchpad_as_ds4 = enabled
```

6.4.6 back_button_timeout

Description

If the Back/Select button is held down for the specified number of milliseconds, a Home/Guide button press is emulated.

Tip: If `back_button_timeout < 0`, then the Home/Guide button will not be emulated.

Default

-1

Example

```
back_button_timeout = 2000
```

6.4.7 keyboard

Description

Whether to allow keyboard input from the client.

Example

```
keyboard = enabled
```

6.4.8 key_repeat_delay

Description

The initial delay, in milliseconds, before repeating keys. Controls how fast keys will repeat themselves.

Default

500

Example

```
key_repeat_delay = 500
```

6.4.9 key_repeat_frequency

Description

How often keys repeat every second.

Tip: This configurable option supports decimals.

Default

24.9

Example

```
key_repeat_frequency = 24.9
```

6.4.10 always_send_scancodes

Description

Sending scancodes enhances compatibility with games and apps but may result in incorrect keyboard input from certain clients that aren't using a US English keyboard layout.

Enable if keyboard input is not working at all in certain applications.

Disable if keys on the client are generating the wrong input on the host.

Caution: Applies to Windows only.

Default

enabled

Example

```
always_send_scancodes = enabled
```

6.4.11 key_rightalt_to_key_win

Description

It may be possible that you cannot send the Windows Key from Moonlight directly. In those cases it may be useful to make Sunshine think the Right Alt key is the Windows key.

Default

disabled

Example

```
key_rightalt_to_key_win = enabled
```

6.4.12 mouse

Description

Whether to allow mouse input from the client.

Example

```
mouse = enabled
```


6.4.13 high_resolution_scrolling

Description

When enabled, Sunshine will pass through high resolution scroll events from Moonlight clients.

This can be useful to disable for older applications that scroll too fast with high resolution scroll events.

Default

enabled

Example

```
high_resolution_scrolling = enabled
```

6.4.14 native_pen_touch

Description

When enabled, Sunshine will pass through native pen/touch events from Moonlight clients.

This can be useful to disable for older applications without native pen/touch support.

Default

enabled

Example

```
native_pen_touch = enabled
```

6.4.15 keybindings

Description

Sometimes it may be useful to map keybindings. Wayland won't allow clients to capture the Win Key for example.

Tip: See [virtual key codes](#)

Hint: keybindings needs to have a multiple of two elements.

Default

```
[
  0x10, 0xA0,
  0x11, 0xA2,
  0x12, 0xA4
]
```

Example

```
keybindings = [
  0x10, 0xA0,
  0x11, 0xA2,
  0x12, 0xA4,
```

(continues on next page)

(continued from previous page)

```
0x4A, 0x4B
]
```

Note: This option is not available in the UI. A PR would be welcome.

6.5 Audio/Video

6.5.1 audio_sink

Description

The name of the audio sink used for audio loopback.

Tip: To find the name of the audio sink follow these instructions.

Linux + pulseaudio

```
pacmd list-sinks | grep "name:"
```

Linux + pipewire

```
pactl info | grep Source
# in some causes you'd need to use the `Sink` device, if `Source` doesn't work, so
↪ try:
pactl info | grep Sink
```

macOS

Sunshine can only access microphones on macOS due to system limitations. To stream system audio use [Soundflower](#) or [BlackHole](#).

Windows

```
tools\audio-info.exe
```

Tip: If you have multiple audio devices with identical names, use the Device ID instead.

Tip: If you want to mute the host speakers, use [virtual_sink](#) instead.

Default

Sunshine will select the default audio device.

Examples

Linux

```
audio_sink = alsa_output.pci-0000_09_00.3.analog-stereo
```

macOS

```
audio_sink = BlackHole 2ch
```

Windows

```
audio_sink = Speakers (High Definition Audio Device)
```

6.5.2 virtual_sink

Description

The audio device that's virtual, like Steam Streaming Speakers. This allows Sunshine to stream audio, while muting the speakers.

Tip: See [audio_sink](#)!

Tip: These are some options for virtual sound devices.

- Stream Streaming Speakers (Linux, macOS, Windows)
 - Steam must be installed.
 - Enable [install_steam_audio_drivers](#) or use Steam Remote Play at least once to install the drivers.
 - [Virtual Audio Cable](#) (macOS, Windows)
-

Example

```
virtual_sink = Steam Streaming Speakers
```

6.5.3 install_steam_audio_drivers

Description

Installs the Steam Streaming Speakers driver (if Steam is installed) to support surround sound and muting host audio.

Tip: This option is only supported on Windows.

Default

enabled

Example

```
install_steam_audio_drivers = enabled
```

6.5.4 adapter_name

Description

Select the video card you want to stream.

Tip: To find the name of the appropriate values follow these instructions.

Linux + VA-API

Unlike with *amdvce* and *nvenc*, it doesn't matter if video encoding is done on a different GPU.

```
ls /dev/dri/renderD*  # to find all devices capable of VAAPI

# replace ``renderD129`` with the device from above to lists the name and
↳ capabilities of the device
vainfo --display drm --device /dev/dri/renderD129 | \
  grep -E "((VAProfileH264High|VAProfileHEVCMain|VAProfileHEVCMain10).
↳ *VAEntrypointEncSlice)|Driver version"
```

To be supported by Sunshine, it needs to have at the very minimum: VAProfileH264High : VAEntrypointEncSlice

Todo: macOS

Windows

```
tools\dxgi-info.exe
```

Note: For hybrid graphics systems, DXGI reports the outputs are connected to whichever graphics adapter that the application is configured to use, so it's not a reliable indicator of how the display is physically connected.

Default

Sunshine will select the default video card.

Examples

Linux

```
adapter_name = /dev/dri/renderD128
```

Todo: macOS

Windows

```
adapter_name = Radeon RX 580 Series
```

6.5.5 output_name

Description

Select the display number you want to stream.

Tip: To find the name of the appropriate values follow these instructions.

Linux

During Sunshine startup, you should see the list of detected monitors:

```
Info: Detecting connected monitors
Info: Detected monitor 0: DVI-D-0, connected: false
Info: Detected monitor 1: HDMI-0, connected: true
Info: Detected monitor 2: DP-0, connected: true
Info: Detected monitor 3: DP-1, connected: false
Info: Detected monitor 4: DVI-D-1, connected: false
```

You need to use the value before the colon in the output, e.g. 1.

Todo: macOS

Windows

```
tools\dxgi-info.exe
```

Default

Sunshine will select the default display.

Examples

Linux

```
output_name = 0
```

Todo: macOS

Windows

```
output_name = \\.\DISPLAY1
```

6.5.6 resolutions

Description

The resolutions advertised by Sunshine.

Note: Some versions of Moonlight, such as Moonlight-nx (Switch), rely on this list to ensure that the requested resolution is supported.

Default

```
[
  352x240,
  480x360,
  858x480,
  1280x720,
  1920x1080,
  2560x1080,
  3440x1440,
  1920x1200,
  3840x2160,
  3840x1600,
]
```

Example

```
resolutions = [
  352x240,
  480x360,
  858x480,
  1280x720,
  1920x1080,
  2560x1080,
  3440x1440,
  1920x1200,
  3840x2160,
  3840x1600,
]
```

6.5.7 fps

Description

The fps modes advertised by Sunshine.

Note: Some versions of Moonlight, such as Moonlight-nx (Switch), rely on this list to ensure that the requested fps is supported.

Default

[10, 30, 60, 90, 120]

Example

```
fps = [10, 30, 60, 90, 120]
```

6.6 Network

6.6.1 upnp

Description

Sunshine will attempt to open ports for streaming over the internet.

Choices

Value	Description
on	enable UPnP
off	disable UPnP

Default

disabled

Example

```
upnp = on
```

6.6.2 address_family

Description

Set the address family that Sunshine will use.

Value	Description
ipv4	IPv4 only
both	IPv4+IPv6

Default

ipv4

Example

```
address_family = both
```

6.6.3 port

Description

Set the family of ports used by Sunshine. Changing this value will offset other ports per the table below.

Port Description	Default Port	Difference from config port
HTTPS	47984 TCP	-5
HTTP	47989 TCP	0
Web	47990 TCP	+1
RTSP	48010 TCP	+21
Video	47998 UDP	+9
Control	47999 UDP	+10
Audio	48000 UDP	+11
Mic (unused)	48002 UDP	+13

Attention: Custom ports may not be supported by all Moonlight clients.

Default

47989

Range

1029-65514

Example

```
port = 47989
```

6.6.4 origin_web_ui_allowed

Description

The origin of the remote endpoint address that is not denied for HTTPS Web UI.

Choices

Value	Description
pc	Only localhost may access the web ui
lan	Only LAN devices may access the web ui
wan	Anyone may access the web ui

Default

lan

Example

```
origin_web_ui_allowed = lan
```


6.6.5 external_ip

Description

If no external IP address is given, Sunshine will attempt to automatically detect external ip-address.

Default

Automatic

Example

```
external_ip = 123.456.789.12
```

6.6.6 lan_encryption_mode

Description

This determines when encryption will be used when streaming over your local network.

Warning: Encryption can reduce streaming performance, particularly on less powerful hosts and clients.

Choices

Value	Description
0	encryption will not be used
1	encryption will be used if the client supports it
2	encryption is mandatory and unencrypted connections are rejected

Default

0

Example

```
lan_encryption_mode = 0
```

6.6.7 wan_encryption_mode

Description

This determines when encryption will be used when streaming over the Internet.

Warning: Encryption can reduce streaming performance, particularly on less powerful hosts and clients.

Choices

Value	Description
0	encryption will not be used
1	encryption will be used if the client supports it
2	encryption is mandatory and unencrypted connections are rejected

Default

1

Example

```
wan_encryption_mode = 1
```

6.6.8 ping_timeout

Description

How long to wait, in milliseconds, for data from Moonlight before shutting down the stream.

Default

10000

Example

```
ping_timeout = 10000
```

6.7 Config Files

6.7.1 file_apps

Description

The application configuration file path. The file contains a json formatted list of applications that can be started by Moonlight.

Default

OS and package dependent

Example

```
file_apps = apps.json
```

6.7.2 credentials_file

Description

The file where user credentials for the UI are stored.

Default

sunshine_state.json

Example

```
credentials_file = sunshine_state.json
```

6.7.3 log_path

Description

The path where the sunshine log is stored.

Default

sunshine.log

Example

```
log_path = sunshine.log
```

6.7.4 pkey

Description

The private key used for the web UI and Moonlight client pairing. For best compatibility, this should be an RSA-2048 private key.

Warning: Not all Moonlight clients support ECDSA keys or RSA key lengths other than 2048 bits.

Default

credentials/cakey.pem

Example

```
pkey = /dir/pkey.pem
```

6.7.5 cert

Description

The certificate used for the web UI and Moonlight client pairing. For best compatibility, this should have an RSA-2048 public key.

Warning: Not all Moonlight clients support ECDSA keys or RSA key lengths other than 2048 bits.

Default

credentials/cacert.pem

Example

```
cert = /dir/cert.pem
```

6.7.6 file_state

Description

The file where current state of Sunshine is stored.

Default

sunshine_state.json

Example

```
file_state = sunshine_state.json
```

6.8 Advanced

6.8.1 fec_percentage

Description

Percentage of error correcting packets per data packet in each video frame.

Warning: Higher values can correct for more network packet loss, but at the cost of increasing bandwidth usage.

Default

20

Range

1-255

Example

```
fec_percentage = 20
```

6.8.2 qp

Description

Quantization Parameter. Some devices don't support Constant Bit Rate. For those devices, QP is used instead.

Warning: Higher value means more compression, but less quality.

Default

28

Example

```
qp = 28
```

6.8.3 min_threads

Description

Minimum number of CPU threads used for encoding.

Note: Increasing the value slightly reduces encoding efficiency, but the tradeoff is usually worth it to gain the use of more CPU cores for encoding. The ideal value is the lowest value that can reliably encode at your desired streaming settings on your hardware.

Default

2

Example

```
min_threads = 2
```

6.8.4 hevc_mode

Description

Allows the client to request HEVC Main or HEVC Main10 video streams.

Warning: HEVC is more CPU-intensive to encode, so enabling this may reduce performance when using software encoding.

Choices

Value	Description
0	advertise support for HEVC based on encoder capabilities (recommended)
1	do not advertise support for HEVC
2	advertise support for HEVC Main profile
3	advertise support for HEVC Main and Main10 (HDR) profiles

Default

0

Example

```
hevc_mode = 2
```

6.8.5 av1_mode

Description

Allows the client to request AV1 Main 8-bit or 10-bit video streams.

Warning: AV1 is more CPU-intensive to encode, so enabling this may reduce performance when using software encoding.

Choices

Value	Description
0	advertise support for AV1 based on encoder capabilities (recommended)
1	do not advertise support for AV1
2	advertise support for AV1 Main 8-bit profile
3	advertise support for AV1 Main 8-bit and 10-bit (HDR) profiles

Default

0

Example

```
av1_mode = 2
```

6.8.6 capture

Description

Force specific screen capture method.

Caution: Applies to Linux only.

Choices

Value	Description
nvfb	Use NVIDIA Frame Buffer Capture to capture direct to GPU memory. This is usually the fastest method for NVIDIA cards. For GeForce cards it will only work with drivers patched with nvidia-patch or nvlax .
wlr	Capture for wlroots based Wayland compositors via DMA-BUF.
kms	DRM/KMS screen capture from the kernel. This requires that sunshine has cap_sys_admin capability. See Linux Setup .
x11	Uses XCB. This is the slowest and most CPU intensive so should be avoided if possible.

Default

Automatic. Sunshine will use the first capture method available in the order of the table above.

Example

```
capture = kms
```

6.8.7 encoder

Description

Force a specific encoder.

Choices

Value	Description
nvenc	For NVIDIA graphics cards
quicksync	For Intel graphics cards
amdvc	For AMD graphics cards
software	Encoding occurs on the CPU

Default

Sunshine will use the first encoder that is available.

Example

```
encoder = nvenc
```

6.9 NVIDIA NVENC Encoder

6.9.1 nvenc_preset

Description

NVENC encoder performance preset. Higher numbers improve compression (quality at given bitrate) at the cost of increased encoding latency. Recommended to change only when limited by network or decoder, otherwise similar effect can be accomplished by increasing bitrate.

Note: This option only applies when using NVENC *encoder*.

Choices

Value	Description
1	P1 (fastest)
2	P2
3	P3
4	P4
5	P5
6	P6
7	P7 (slowest)

Default

1

Example

```
nvenc_preset = 1
```

6.9.2 nvenc_twopass

Description

Enable two-pass mode in NVENC encoder. This allows to detect more motion vectors, better distribute bitrate across the frame and more strictly adhere to bitrate limits. Disabling it is not recommended since this can lead to occasional bitrate overshoot and subsequent packet loss.

Note: This option only applies when using NVENC *encoder*.

Choices

Value	Description
disabled	One pass (fastest)
quarter_res	Two passes, first pass at quarter resolution (faster)
full_res	Two passes, first pass at full resolution (slower)

Default

quarter_res

Example

```
nvenc_twopass = quarter_res
```

6.9.3 nvenc_spatial_aq

Description

Assign higher QP values to flat regions of the video. Recommended to enable when streaming at lower bitrates.

Note: This option only applies when using NVENC *encoder*.

Choices

Value	Description
disabled	Don't enable Spatial AQ (faster)
enabled	Enable Spatial AQ (slower)

Default

disabled

Example

```
nvenc_spatial_aq = disabled
```


6.9.4 nvenc_vbv_increase

Description

Single-frame VBV/HRD percentage increase. By default sunshine uses single-frame VBV/HRD, which means any encoded video frame size is not expected to exceed requested bitrate divided by requested frame rate. Relaxing this restriction can be beneficial and act as low-latency variable bitrate, but may also lead to packet loss if the network doesn't have buffer headroom to handle bitrate spikes. Maximum accepted value is 400, which corresponds to 5x increased encoded video frame upper size limit.

Note: This option only applies when using NVENC *encoder*.

Warning: Can lead to network packet loss.

Default

0

Range

0-400

Example

```
nvenc_vbv_increase = 0
```

6.9.5 nvenc_realtime_hags

Description

Use realtime gpu scheduling priority in NVENC when hardware accelerated gpu scheduling (HAGS) is enabled in Windows. Currently NVIDIA drivers may freeze in encoder when HAGS is enabled, realtime priority is used and VRAM utilization is close to maximum. Disabling this option lowers the priority to high, sidestepping the freeze at the cost of reduced capture performance when the GPU is heavily loaded.

Note: This option only applies when using NVENC *encoder*.

Caution: Applies to Windows only.

Choices

Value	Description
disabled	Use high priority
enabled	Use realtime priority

Default

enabled

Example

```
nvenc_realtime_hags = enabled
```

6.9.6 nvenc_latency_over_power

Description

Adaptive P-State algorithm which NVIDIA drivers employ doesn't work well with low latency streaming, so sunshine requests high power mode explicitly.

Note: This option only applies when using NVENC *encoder*.

Warning: Disabling it is not recommended since this can lead to significantly increased encoding latency.

Caution: Applies to Windows only.

Choices

Value	Description
disabled	Sunshine doesn't change GPU power preferences (not recommended)
enabled	Sunshine requests high power mode explicitly

Default

enabled

Example

```
nvenc_latency_over_power = enabled
```

6.9.7 nvenc_opengl_vulkan_on_dxgi

Description

Sunshine can't capture fullscreen OpenGL and Vulkan programs at full frame rate unless they present on top of DXGI. This is system-wide setting that is reverted on sunshine program exit.

Note: This option only applies when using NVENC *encoder*.

Caution: Applies to Windows only.

Choices

Value	Description
disabled	Sunshine leaves global Vulkan/OpenGL present method unchanged
enabled	Sunshine changes global Vulkan/OpenGL present method to "Prefer layered on DXGI Swapchain"

Default

enabled

Example

```
nvenc_opengl_vulkan_on_dxgi = enabled
```

6.9.8 nvenc_h264_cavlc

Description

Prefer CAVLC entropy coding over CABAC in H.264 when using NVENC. CAVLC is outdated and needs around 10% more bitrate for same quality, but provides slightly faster decoding when using software decoder.

Note: This option only applies when using H.264 format with NVENC *encoder*.

Choices

Value	Description
disabled	Prefer CABAC
enabled	Prefer CAVLC

Default

disabled

Example

```
nvenc_h264_cavlc = disabled
```

6.10 Intel QuickSync Encoder

6.10.1 qsv_preset

Description

The encoder preset to use.

Note: This option only applies when using quicksync *encoder*.

Choices

Value	Description
veryfast	fastest (lowest quality)
faster	faster (lower quality)
fast	fast (low quality)
medium	medium (default)
slow	slow (good quality)
slower	slower (better quality)
veryslow	slowest (best quality)

Default

medium

Example

```
qsv_preset = medium
```

6.10.2 qsv_coder

Description

The entropy encoding to use.

Note: This option only applies when using H264 with quicksync *encoder*.

Choices

Value	Description
auto	let ffmpeg decide
cabac	context adaptive binary arithmetic coding - higher quality
cavlc	context adaptive variable-length coding - faster decode

Default

auto

Example

```
qsv_coder = auto
```

6.10.3 qsv_slow_hevc

Description

This options enables use of HEVC on older Intel GPUs that only support low power encoding for H.264.

Caution: Streaming performance may be significantly reduced when this option is enabled.

Default

disabled

Example

```
qsv_slow_hevc = disabled
```

6.11 AMD AMF Encoder

6.11.1 amd_quality

Description

The encoder preset to use.

Note: This option only applies when using amdvc *encoder*.

Choices

Value	Description
speed	prefer speed
balanced	balanced
quality	prefer quality

Default

balanced

Example

```
amd_quality = balanced
```

6.11.2 amd_rc

Description

The encoder rate control.

Note: This option only applies when using amdvc *encoder*.

Choices

Value	Description
cqp	constant qp mode
cbr	constant bitrate
vbr_latency	variable bitrate, latency constrained
vbr_peak	variable bitrate, peak constrained

Default

vbr_latency

Example

```
amd_rc = vbr_latency
```

6.11.3 amd_usage

Description

The encoder usage profile, used to balance latency with encoding quality.

Note: This option only applies when using amdvc *encoder*.

Choices

Value	Description
transcoding	transcoding (slowest)
webcam	webcam (slow)
lowlatency	low latency (fast)
ultralowlatency	ultra low latency (fastest)

Default

ultralowlatency

Example

```
amd_usage = ultralowlatency
```

6.11.4 amd_preanalysis

Description

Preanalysis can increase encoding quality at the cost of latency.

Note: This option only applies when using amdvc *encoder*.

Default

disabled

Example

```
amd_preanalysis = disabled
```

6.11.5 amd_vbaq

Description

Variance Based Adaptive Quantization (VBAQ) can increase subjective visual quality.

Note: This option only applies when using amdvc *encoder*.

Default

enabled

Example

```
amd_vbaq = enabled
```

6.11.6 amd_coder

Description

The entropy encoding to use.

Note: This option only applies when using H264 with amdvc *encoder*.

Choices

Value	Description
auto	let ffmpeg decide
cabac	context adaptive variable-length coding - higher quality
cavlc	context adaptive binary arithmetic coding - faster decode

Default

auto

Example

```
amd_coder = auto
```

6.12 VideoToolbox Encoder

6.12.1 vt_coder

Description

The entropy encoding to use.

Note: This option only applies when using macOS.

Choices

Value	Description
auto	let ffmpeg decide
cabac	
cavlc	

Default

auto

Example

```
vt_coder = auto
```

6.12.2 vt_software

Description

Force Video Toolbox to use software encoding.

Note: This option only applies when using macOS.

Choices

Value	Description
auto	let ffmpeg decide
disabled	disable software encoding
allowed	allow software encoding
forced	force software encoding

Default

auto

Example

```
vt_software = auto
```

6.12.3 vt_realtime

Description

Realtime encoding.

Note: This option only applies when using macOS.

Warning: Disabling realtime encoding might result in a delayed frame encoding or frame drop.

Default

enabled

Example

```
vt_realtime = enabled
```

6.13 Software Encoder

6.13.1 sw_preset

Description

The encoder preset to use.

Note: This option only applies when using software *encoder*.

Note: From FFmpeg.

A preset is a collection of options that will provide a certain encoding speed to compression ratio. A slower preset will provide better compression (compression is quality per filesize). This means that, for example, if you target a certain file size or constant bit rate, you will achieve better quality with a slower preset. Similarly, for constant quality encoding, you will simply save bitrate by choosing a slower preset.

Use the slowest preset that you have patience for.

Choices

Value	Description
ultrafast	fastest
superfast	
veryfast	
faster	
fast	
medium	
slow	
slower	
veryslow	slowest

Default

superfast

Example

```
sw_preset = superfast
```

6.13.2 sw_tune

Description

The tuning preset to use.

Note: This option only applies when using software *encoder*.

Note: From FFmpeg.

You can optionally use -tune to change settings based upon the specifics of your input.

Choices

Value	Description
film	use for high quality movie content; lowers deblocking
animation	good for cartoons; uses higher deblocking and more reference frames
grain	preserves the grain structure in old, grainy film material
stillimage	good for slideshow-like content
fastdecode	allows faster decoding by disabling certain filters
zerolatency	good for fast encoding and low-latency streaming

Default

zerolatency

Example

```
sw_tune = zerolatency
```

GAMESTREAM

Nvidia announced that their GameStream service for Nvidia Games clients will be discontinued in February 2023. Luckily, Sunshine performance is now on par with Nvidia GameStream. Many users have even reported that Sunshine outperforms GameStream, so rest assured that Sunshine will be equally performant moving forward.

7.1 Migration

We have developed a simple migration tool to help you migrate your GameStream games and apps to Sunshine automatically. Please check out our [GSMS](#) project if you're interested in an automated migration option. GSMS offers the ability to migrate your custom and auto-detected games and apps. The working directory, command, and image are all set in Sunshine's `apps.json` file. The box-art image is also copied to a specified directory.

7.2 Internet Streaming

If you are using the Moonlight Internet Hosting Tool, you can remove it from your system when you migrate to Sunshine. To stream over the Internet with Sunshine and a UPnP-capable router, enable the UPnP option in the Sunshine Web UI.

Note: Running Sunshine together with versions of the Moonlight Internet Hosting Tool prior to v5.6 will cause UPnP port forwarding to become unreliable. Either uninstall the tool entirely or update it to v5.6 or later.

7.3 Limitations

Sunshine does have some limitations, as compared to Nvidia GameStream.

- Automatic game/application list.
- Changing game settings automatically, to optimize streaming.

8.1 Forgotten Credentials

If you forgot your credentials to the web UI, try this.

```
sunshine --creds {new-username} {new-password}
```

```
./sunshine.AppImage --creds {new-username} {new-password}
```

```
flatpak run --command=sunshine dev.lizardbyte.Sunshine --creds {new-username} {new-  
↪password}
```

8.2 Web UI Access

Can't access the web UI?

1. Check firewall rules.

8.3 Nvidia issues

NvFBC, NvENC, or general issues with Nvidia graphics card.

- Consumer grade Nvidia cards are software limited to a specific number of encodes. See [Video Encode and Decode GPU Support Matrix](#) for more info.
- You can usually bypass the restriction with a driver patch. See Keylase's [Linux](#) or [Windows](#) patches for more guidance.

9.1 Hardware Encoding fails

Due to legal concerns, Mesa has disabled hardware decoding and encoding by default.

```
Error: Could not open codec [h264_vaapi]: Function not implemented
```

If you see the above error in the Sunshine logs, compiling *Mesa* manually, may be required. See the official [Mesa3D Compiling and Installing](#) documentation for instructions.

Important: You must re-enable the disabled encoders. You can do so, by passing the following argument to the build system. You may also want to enable decoders, however that is not required for Sunshine and is not covered here.

```
-Dvideo-codecs=h264enc,h265enc
```

Note: Other build options are listed in the [meson options](#) file.

9.2 KMS Streaming fails

If screencasting fails with KMS, you may need to run the following to force unprivileged screencasting.

```
sudo setcap -r $(readlink -f $(which sunshine))
```

9.3 Gamescope compatibility

Some users have reported stuttering issues when streaming games running within Gamescope.

10.1 Dynamic session lookup failed

If you get this error:

Dynamic session lookup supported but failed: launchd did not provide a socket path, verify that org.freedesktop.dbus-session.plist is loaded!

Try this.

```
launchctl load -w /Library/LaunchAgents/org.freedesktop.dbus-session.plist
```


11.1 No gamepad detected

1. Verify that you've installed [Nefarius Virtual Gamepad](#).

11.2 Permission denied

Since Sunshine runs as a service on Windows, it may not have the same level of access that your regular user account has. You may get permission denied errors when attempting to launch a game or application from a non system drive.

You will need to modify the security permissions on your disk. Ensure that user/principal SYSTEM has full permissions on the disk.

BUILD

Sunshine binaries are built using [CMake](#). Cross compilation is not supported. That means the binaries must be built on the target operating system and architecture.

12.1 Building Locally

12.1.1 Clone

Ensure [git](#) is installed and run the following:

```
git clone https://github.com/lizardbyte/sunshine.git --recurse-submodules
cd sunshine && mkdir build && cd build
```

12.1.2 Compile

See the section specific to your OS.

- *Linux*
- *macOS*
- *Windows*

12.2 Remote Build

It may be beneficial to build remotely in some cases. This will enable easier building on different operating systems.

1. Fork the project
2. Activate workflows
3. Trigger the *CI* workflow manually
4. Download the artifacts/binaries from the workflow run summary

13.1 Requirements

13.1.1 Debian Bullseye/Bookworm

End of Life (Bullseye): July, 2024 End of Life (Bookworm): TBD

Install Requirements

```
sudo apt update && sudo apt install \  
  build-essential \  
  cmake \  
  libavdevice-dev \  
  libayatana-appindicator3-dev \  
  libboost-filesystem-dev \  
  libboost-locale-dev \  
  libboost-log-dev \  
  libboost-program-options-dev \  
  libcap-dev \ # KMS  
  libcurl4-openssl-dev \  
  libdrm-dev \ # KMS  
  libevdev-dev \  
  libminiupnpc-dev \  
  libmfx-dev \ # x86_64 only  
  libnotify-dev \  
  libnuma-dev \  
  libopus-dev \  
  libpulse-dev \  
  libssl-dev \  
  libva-dev \ # VA-API  
  libvdpau-dev \  
  libwayland-dev \ # Wayland  
  libx11-dev \ # X11  
  libxcb-shm0-dev \ # X11  
  libxcb-xfixes0-dev \ # X11  
  libxcb1-dev \ # X11  
  libxfixes-dev \ # X11  
  libxrandr-dev \ # X11  
  libxtst-dev \ # X11  
  nodejs \  
  npm \
```

(continues on next page)

(continued from previous page)

```
nvidia-cuda-dev \ # Cuda, NvFBC
nvidia-cuda-toolkit # Cuda, NvFBC
```

13.1.2 Fedora 38, 39

Install Requirements

```
sudo dnf update && \
sudo dnf group install "Development Tools" && \
sudo dnf install \
    boost-devel \
    cmake \
    gcc \
    gcc-c++ \
    intel-mediasdk-devel \ # x86_64 only
    libappindicator-gtk3-devel \
    libcap-devel \
    libcurl-devel \
    libdrm-devel \
    libevdev-devel \
    libnotify-devel \
    libva-devel \ # VA-API
    libvdpau-devel \
    libX11-devel \ # X11
    libxcb-devel \ # X11
    libXcursor-devel \ # X11
    libXfixes-devel \ # X11
    libXi-devel \ # X11
    libXinerama-devel \ # X11
    libXrandr-devel \ # X11
    libXtst-devel \ # X11
    mesa-libGL-devel \
    miniupnpc-devel \
    npm \
    numactl-devel \
    openssl-devel \
    opus-devel \
    pulseaudio-libs-devel \
    rpm-build \ # if you want to build an RPM binary package
    wget \ # necessary for cuda install with `run` file
    which \ # necessary for cuda install with `run` file
```


13.1.3 Ubuntu 20.04

End of Life: April 2030

Install Requirements

```
sudo apt update && sudo apt install \
  build-essential \
  cmake \
  g++-10 \
  libayatana-appindicator3-dev \
  libavdevice-dev \
  libboost-filesystem-dev \
  libboost-locale-dev \
  libboost-log-dev \
  libboost-program-options-dev \
  libcap-dev \ # KMS
  libcurl4-openssl-dev \
  libdrm-dev \ # KMS
  libevdev-dev \
  libminiupnpc-dev \
  libmfx-dev \ # x86_64 only
  libnotify-dev \
  libnuma-dev \
  libopus-dev \
  libpulse-dev \
  libssl-dev \
  libva-dev \ # VA-API
  libvdpau-dev \
  libwayland-dev \ # Wayland
  libx11-dev \ # X11
  libxcb-shm0-dev \ # X11
  libxcb-xfixes0-dev \ # X11
  libxcb1-dev \ # X11
  libxfixes-dev \ # X11
  libxrandr-dev \ # X11
  libxtst-dev \ # X11
  nodejs \
  npm \
  wget \ # necessary for cuda install with `run` file
```

Update gcc alias

```
update-alternatives --install \
  /usr/bin/gcc gcc /usr/bin/gcc-10 100 \
  --slave /usr/bin/g++ g++ /usr/bin/g++-10 \
  --slave /usr/bin/gcov gcov /usr/bin/gcov-10 \
  --slave /usr/bin/gcc-ar gcc-ar /usr/bin/gcc-ar-10 \
  --slave /usr/bin/gcc-ranlib gcc-ranlib /usr/bin/gcc-ranlib-10
```

13.1.4 Ubuntu 22.04

End of Life: April 2027

Install Requirements

```
sudo apt update && sudo apt install \  
  build-essential \  
  cmake \  
  libappindicator3-dev \  
  libavdevice-dev \  
  libboost-filesystem-dev \  
  libboost-locale-dev \  
  libboost-log-dev \  
  libboost-program-options-dev \  
  libcap-dev \ # KMS \  
  libcurl4-openssl-dev \  
  libdrm-dev \ # KMS \  
  libevdev-dev \  
  libminiupnpc-dev \  
  libmfx-dev \ # x86_64 only \  
  libnotify-dev \  
  libnuma-dev \  
  libopus-dev \  
  libpulse-dev \  
  libssl-dev \  
  libva-dev \ # VA-API \  
  libwayland-dev \ # Wayland \  
  libx11-dev \ # X11 \  
  libxcb-shm0-dev \ # X11 \  
  libxcb-xfixes0-dev \ # X11 \  
  libxcb1-dev \ # X11 \  
  libxfixes-dev \ # X11 \  
  libxrandr-dev \ # X11 \  
  libxtst-dev \ # X11 \  
  nodejs \  
  npm \  
  nvidia-cuda-dev \ # CUDA, NvFBC \  
  nvidia-cuda-toolkit \ # CUDA, NvFBC
```

13.2 CUDA

If the version of CUDA available from your distro is not adequate, manually install CUDA.

Tip: The version of CUDA you use will determine compatibility with various GPU generations. See [CUDA compatibility](#) for more info.

Select the appropriate run file based on your desired CUDA version and architecture according to [CUDA Toolkit Archive](#).

```
wget https://developer.download.nvidia.com/compute/cuda/11.8.0/local_installers/cuda_11.8.0_520.61.05_linux.run \
--progress=bar:force:noscroll -q --show-progress -O ./cuda.run
chmod a+x ./cuda.run
./cuda.run --silent --toolkit --toolkitpath=/usr --no-opengl-libs --no-man-page --no-drm
rm ./cuda.run
```

13.3 Build

Attention: Ensure you are in the build directory created during the clone step earlier before continuing.

```
cmake ..
make -j ${nproc}

cpack -G DEB # optionally, create a deb package
cpack -G RPM # optionally, create a rpm package
```


14.1 Requirements

macOS Big Sur and Xcode 12.5+

Use either [MacPorts](#) or [Homebrew](#)

14.1.1 MacPorts

Install Requirements

```
sudo port install avahi boost180 cmake curl libopus miniupnpc npm9 pkgconfig
```

14.1.2 Homebrew

Install Requirements

```
brew install boost cmake miniupnpc node opus pkg-config
```

If there are issues with an SSL header that is not found:

```
pushd /usr/local/include  
ln -s ../opt/openssl/include/openssl .  
popd
```

```
pushd /opt/homebrew/include  
ln -s ../opt/openssl/include/openssl .  
popd
```

14.2 Build

Attention: Ensure you are in the build directory created during the clone step earlier before continuing.

```
cmake ..  
make -j ${nproc}  
  
cpack -G DragNDrop # optionally, create a macOS dmg package
```

If cmake fails complaining to find Boost, try to set the path explicitly.

```
cmake -DBOOST_ROOT=[boost path] .., e.g., cmake -DBOOST_ROOT=/opt/local/libexec/boost/1.  
80 ..
```

WINDOWS

15.1 Requirements

First you need to install [MSYS2](#), then startup “MSYS2 MinGW 64-bit” and execute the following codes.

Update all packages:

```
pacman -Suy
```

Install dependencies:

```
pacman -S \
  base-devel \
  cmake \
  diffutils \
  gcc \
  git \
  make \
  mingw-w64-x86_64-binutils \
  mingw-w64-x86_64-boost \
  mingw-w64-x86_64-cmake \
  mingw-w64-x86_64-curl \
  mingw-w64-x86_64-miniupnpc \
  mingw-w64-x86_64-nlohmann-json \
  mingw-w64-x86_64-nodejs \
  mingw-w64-x86_64-onevpl \
  mingw-w64-x86_64-openssl \
  mingw-w64-x86_64-opus \
  mingw-w64-x86_64-toolchain
```

15.2 Build

Attention: Ensure you are in the build directory created during the clone step earlier before continuing.

```
cmake -G "MinGW Makefiles" ..
mingw32-make -j$(nproc)
```

(continues on next page)

(continued from previous page)

```
cpack -G NSIS # optionally, create a windows installer  
cpack -G ZIP  # optionally, create a windows standalone package
```


CONTRIBUTING

Read our contribution guide in our organization level [docs](#).

16.1 Web UI

The Web UI uses [Vite](#) as its build system, to handle the integration of the NPM libraries.

The HTML pages used by the Web UI are found in `src_assets/common/assets/web`.

[EJS](#) is used as a templating system for the pages (check `template_header.html` and `template_header_main.html`).

The Style System is provided by [Bootstrap](#).

The JS framework used by the more interactive pages is [Vue](#).

16.1.1 Building

Sunshine already builds the UI as part of its build process, but you can make faster changes by starting vite manually.

```
npm run dev
```


LOCALIZATION

Sunshine and related LizardByte projects are being localized into various languages. The default language is *en* (English).

17.1 CrowdIn

The translations occur on [CrowdIn](#). Anyone is free to contribute to localization there.

Translations Basics

- The brand names *LizardByte* and *Sunshine* should never be translated.
- Other brand names should never be translated. Examples:
 - AMD
 - Nvidia

CrowdIn Integration

How does it work?

When a change is made to sunshine source code, a workflow generates new translation templates that get pushed to CrowdIn automatically.

When translations are updated on CrowdIn, a push gets made to the *l10n_nightly* branch and a PR is made against the *nightly* branch. Once PR is merged, all updated translations are part of the project and will be included in the next release.

17.2 Extraction

There should be minimal cases where strings need to be extracted from source code; however it may be necessary in some situations. For example if a system tray icon is added it should be localized as it is user interfacing.

- Wrap the string to be extracted in a function as shown.

```
#include <boost/locale.hpp>
#include <string>

std::string msg = boost::locale::translate("Hello world!");
```

Tip: More examples can be found in the documentation for [boost locale](#).

Warning: This is for information only. Contributors should never include manually updated template files, or manually compiled language files in Pull Requests.

Strings are automatically extracted from the code to the *locale/sunshine.po* template file. The generated file is used by CrowdIn to generate language specific template files. The file is generated using the *.github/workflows/localize.yml* workflow and is run on any push event into the *nightly* branch. Jobs are only run if any of the following paths are modified.

```
- 'src/**'
```

When testing locally it may be desirable to manually extract, initialize, update, and compile strings. Python is required for this, along with the python dependencies in the *./scripts/requirements.txt* file. Additionally, [xgettext](#) must be installed.

Extract, initialize, and update

```
python ./scripts/_locale.py --extract --init --update
```

Compile

```
python ./scripts/_locale.py --compile
```

TESTING

18.1 Clang Format

Source code is tested against the *.clang-format* file for linting errors. The workflow file responsible for clang format testing is *.github/workflows/cpp-clang-format-lint.yml*.

Test clang-format locally.

```
find ./ -iname *.cpp -o -iname *.h -iname *.m -iname *.mm | xargs clang-format -i
```

18.2 Sphinx

Sunshine uses [Sphinx](#) for documentation building. Sphinx, along with other required python dependencies are included in the *.docs/requirements.txt* file. Python is required to build sphinx docs. Installation and setup of python will not be covered here.

Doxygen is used to generate the XML files required by Sphinx. Doxygen can be obtained from [Doxygen downloads](#). Ensure that the *doxygen* executable is in your path.

See also:

Sphinx is configured to use the graphviz extension. To obtain the dot executable from the Graphviz library, see the [library's downloads section](#).

The config file for Sphinx is *docs/source/conf.py*. This is already included in the repo and should not be modified.

The config file for Doxygen is *docs/Doxyfile*. This is already included in the repo and should not be modified.

Test with Sphinx

```
cd docs  
make html
```

Alternatively

```
cd docs  
sphinx-build -b html source build
```

Lint with rstcheck

```
rstcheck -r .
```

Check formatting with rstfmt

```
rstfmt --check --diff -w 120 .
```

Format inplace with rstfmt

```
rstfmt -w 120 .
```

18.3 Unit Testing

Todo: Sunshine does not currently have any unit tests. If you would like to help us improve please get in contact with us, or make a PR with suggested changes.

Attention: This documentation is for informational purposes only and is not intended as legal advice. If you have any legal questions or concerns about using Sunshine, we recommend consulting with a lawyer.

Sunshine is licensed under the GPL-3.0 license, which allows for free use and modification of the software. The full text of the license can be reviewed [here](#).

19.1 Commercial Use

Sunshine can be used in commercial applications without any limitations. This means that businesses and organizations can use Sunshine to create and sell products or services without needing to seek permission or pay a fee.

However, it is important to note that the GPL-3.0 license does not grant any rights to distribute or sell the encoders contained within Sunshine. If you plan to sell access to Sunshine as part of their distribution, you are responsible for obtaining the necessary licenses to do so. This may include obtaining a license from the Motion Picture Experts Group (MPEG-LA) and/or any other necessary licensing requirements.

In summary, while Sunshine is free to use, it is the user's responsibility to ensure compliance with all applicable licensing requirements when redistributing the software as part of a commercial offering. If you have any questions or concerns about using Sunshine in a commercial setting, we recommend consulting with a lawyer.

SOURCE CODE

We are in process of improving the source code documentation. Code should be documented using Doxygen syntax. Some examples exist in *main.h* and *main.cpp*. In order for documentation within the code to appear in the rendered docs, the definition of the object must be in a header file, although the documentation itself can (and should) be in the source file.

20.1 Example Documentation Blocks

file.h

```
// functions
int main(int argc, char *argv[]);
```

file.cpp (with markdown)

```
/**
 * @brief Main application entry point.
 * @param argc The number of arguments.
 * @param argv The arguments.
 *
 * EXAMPLES:
 * ```cpp
 * main(1, const char* args[] = {"hello", "markdown", nullptr});
 * ```
 */
int main(int argc, char *argv[]) {
    // do stuff
}
```

file.cpp (with ReStructuredText)

```
/**
 * @brief Main application entry point.
 * @param argc The number of arguments.
 * @param argv The arguments.
 * @rst
 * EXAMPLES:
 *
 * .. code-block:: cpp
 *     main(1, const char* args[] = {"hello", "rst", nullptr});
```

(continues on next page)

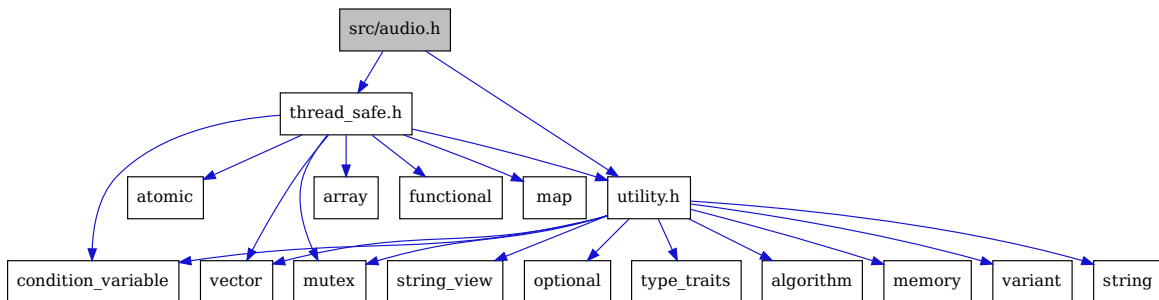
(continued from previous page)

```
* @endrst
*/
int main(int argc, char *argv[]) {
    // do stuff
}
```

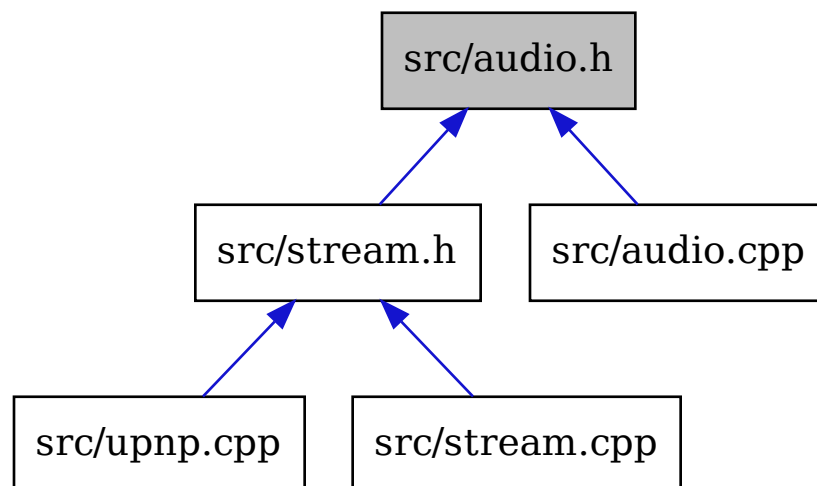
20.2 Source

20.2.1 audio

Include dependency graph for audio.h:



This graph shows which files directly or indirectly include audio.h:



todo

namespace **audio**

Typedefs

using **buffer_t** = util::buffer_t<std::uint8_t>

using **packet_t** = std::pair<void*, *buffer_t*>

Enums

enum **stream_config_e**

Values:

enumerator **STEREO**

enumerator **HIGH_STEREO**

enumerator **SURROUND51**

enumerator **HIGH_SURROUND51**

enumerator **SURROUND71**

enumerator **HIGH_SURROUND71**

enumerator **MAX_STREAM_CONFIG**

struct **config_t**

Public Types

enum **flags_e**

Values:

enumerator **HIGH_QUALITY**

enumerator **HOST_AUDIO**

enumerator **MAX_FLAGS**

Public Members

int **channels**

std::bitset<MAX_FLAGS> **flags**

int **mask**

int **packetDuration**

struct **opus_stream_config_t**

Public Members

int **bitrate**

int **channelCount**

int **coupledStreams**

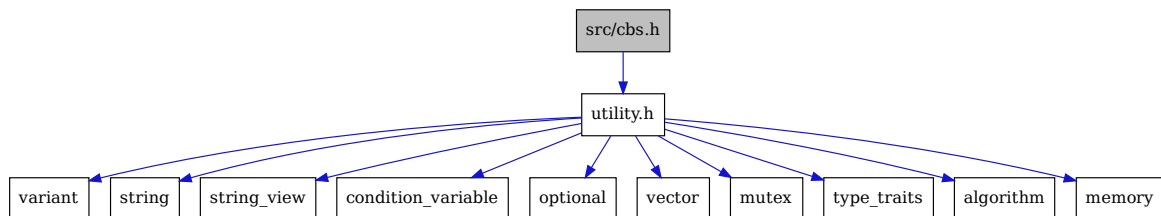
const std::uint8_t ***mapping**

std::int32_t **sampleRate**

int **streams**

20.2.2 cbs

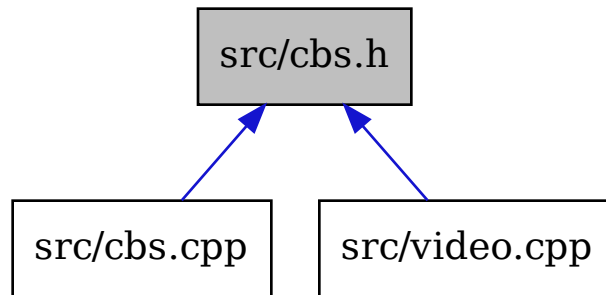
Include dependency graph for cbs.h:



This graph shows which files directly or indirectly include `cbs.h`:

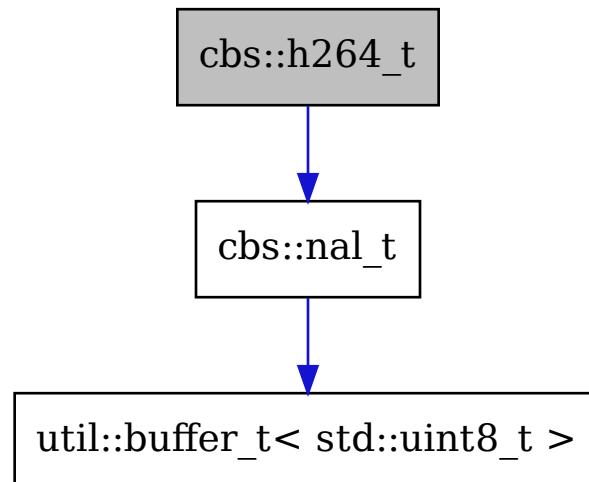
todo

namespace **cbs**



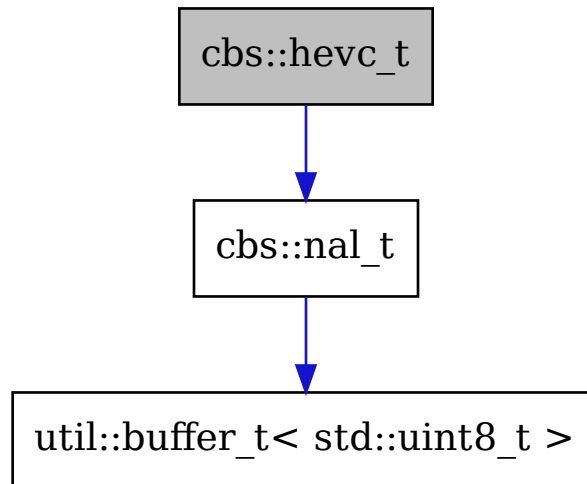
struct **h264_t**

Collaboration diagram for `cbs::h264_t`:



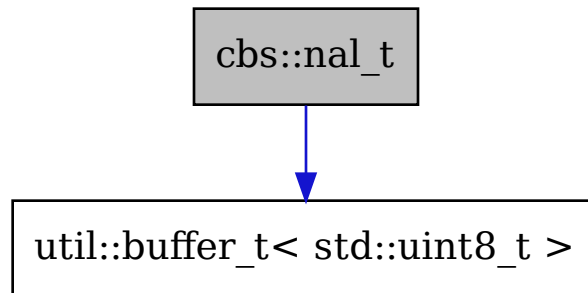
Public Members*nal_t* **sps**struct **hevc_t**

Collaboration diagram for cbs::hevc_t:

**Public Members***nal_t* **sps***nal_t* **vps**struct **nal_t**

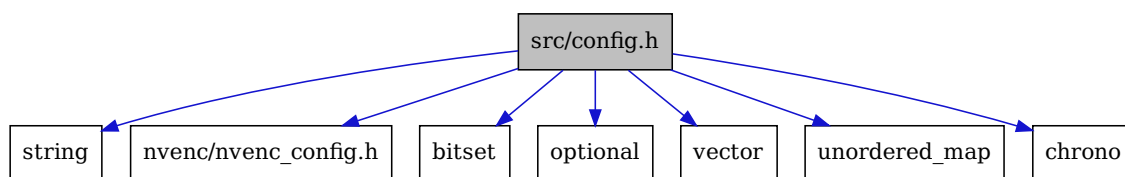
Collaboration diagram for cbs::nal_t:

Public Members`util::buffer_t<std::uint8_t> _new``util::buffer_t<std::uint8_t> old`



20.2.3 config

Include dependency graph for config.h:



This graph shows which files directly or indirectly include config.h:



todo

namespace **config**

Variables

```
constexpr int ENCRYPTION_MODE_MANDATORY = 2
```

```
constexpr int ENCRYPTION_MODE_NEVER = 0
```

```
constexpr int ENCRYPTION_MODE_OPPORTUNISTIC = 1
```

```
struct audio_t
```

Public Members

bool **install_steam_drivers**

std::string **sink**

std::string **virtual_sink**

struct **input_t**

Public Members

bool **always_send_scancodes**

std::chrono::milliseconds **back_button_timeout**

bool **controller**

bool **ds4_back_as_touchpad_click**

std::string **gamepad**

bool **high_resolution_scrolling**

std::chrono::milliseconds **key_repeat_delay**

std::chrono::duration<double> **key_repeat_period**

std::unordered_map<int, int> **keybindings**

bool **keyboard**

bool **motion_as_ds4**

bool **mouse**

bool **native_pen_touch**

bool **touchpad_as_ds4**

struct **nvhttp_t**

Public Members

std::string **cert**

std::string **external_ip**

std::string **file_state**

std::vector<int> **fps**

std::string **origin_web_ui_allowed**

std::string **pkey**

std::vector<std::string> **resolutions**

std::string **sunshine_name**

struct **prep_cmd_t**

Public Functions

inline explicit **prep_cmd_t**(std::string &&do_cmd, bool &&elevated)

inline **prep_cmd_t**(std::string &&do_cmd, std::string &&undo_cmd, bool &&elevated)

Public Members

std::string **do_cmd**

bool **elevated**

std::string **undo_cmd**

struct **stream_t**

Public Members

int **channels**

int **fec_percentage**

std::string **file_apps**

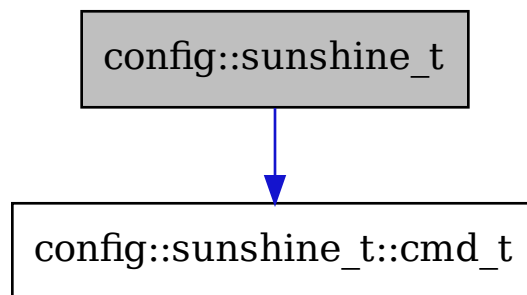
int **lan_encryption_mode**

std::chrono::milliseconds **ping_timeout**

int **wan_encryption_mode**

struct **sunshine_t**

Collaboration diagram for config::sunshine_t:

**Public Members**

std::string **address_family**

struct *config::sunshine_t::cmd_t* **cmd**

std::string **config_file**

std::string **credentials_file**

std::bitset<*flag::FLAG_SIZE*> **flags**

```
std::string log_file

int min_log_level

std::string password

std::uint16_t port

std::vector<prep_cmd_t> prep_cmds

std::string salt

std::string username

struct cmd_t
```

Public Members

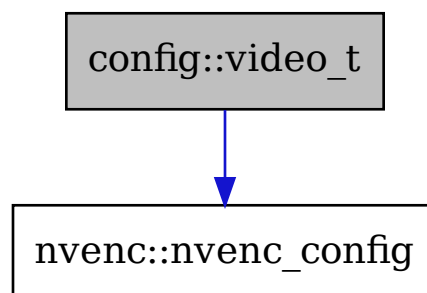
```
int argc

char **argv

std::string name

struct video_t
```

Collaboration diagram for config::video_t:



Public Members

```
std::string adapter_name

struct config::video_t::[anonymous] amd

int amd_coder

std::optional<int> amd_prealanalysis

std::optional<int> amd_quality_av1

std::optional<int> amd_quality_h264

std::optional<int> amd_quality_hevc

std::optional<int> amd_rc_av1

std::optional<int> amd_rc_h264

std::optional<int> amd_rc_hevc

std::optional<int> amd_usage_av1

std::optional<int> amd_usage_h264

std::optional<int> amd_usage_hevc

std::optional<int> amd_vbaq

int aq

int av1_mode

std::string capture

std::string encoder

int h264_coder

int hevc_mode
```

```
int min_threads

int multipass

nvenc::nvenc_config nv

struct config::video_t::[anonymous] nv_legacy

bool nv_opengl_vulkan_on_dxgi

bool nv_realtime_hags

bool nv_sunshine_high_power_mode

std::string output_name

int preset

int qp

struct config::video_t::[anonymous] qsv

std::optional<int> qsv_cavlc

std::optional<int> qsv_preset

bool qsv_slow_hevc

std::optional<int> svtav1_preset

struct config::video_t::[anonymous] sw

std::string sw_preset

std::string sw_tune

int vbv_percentage_increase

struct config::video_t::[anonymous] vt

int vt_allow_sw
```

```
int vt_coder
```

```
int vt_realtime
```

```
int vt_require_sw
```

```
namespace flag
```

Enums

```
enum flag_e
```

Values:

```
enumerator PIN_STDIN
```

```
enumerator FRESH_STATE
```

```
enumerator FORCE_VIDEO_HEADER_REPLACE
```

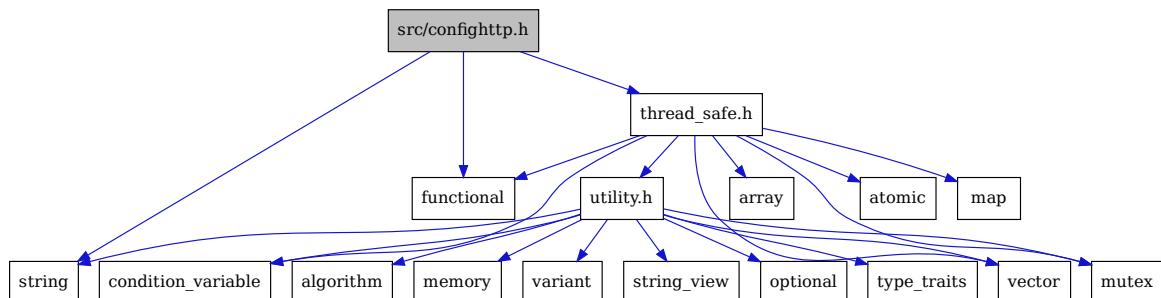
```
enumerator UPNP
```

```
enumerator CONST_PIN
```

```
enumerator FLAG_SIZE
```

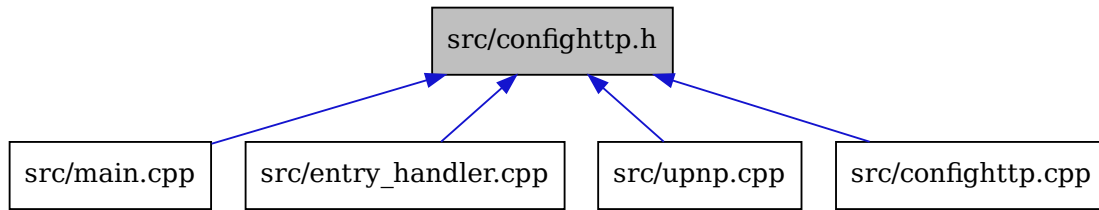
20.2.4 confighttp

Include dependency graph for confighttp.h:



This graph shows which files directly or indirectly include `confighttp.h`:

todo



Defines

`WEB_DIR`

Variables

```
const std::map<std::string, std::string> mime_types = {{"css", "text/css"}, {"gif", "image/gif"}, {"htm", "text/html"}, {"html", "text/html"}, {"ico", "image/x-icon"}, {"jpeg", "image/jpeg"}, {"jpg", "image/jpeg"}, {"js", "application/javascript"}, {"json", "application/json"}, {"png", "image/png"}, {"svg", "image/svg+xml"}, {"ttf", "font/ttf"}, {"txt", "text/plain"}, {"woff2", "font/woff2"}, {"xml", "text/xml"},}
```

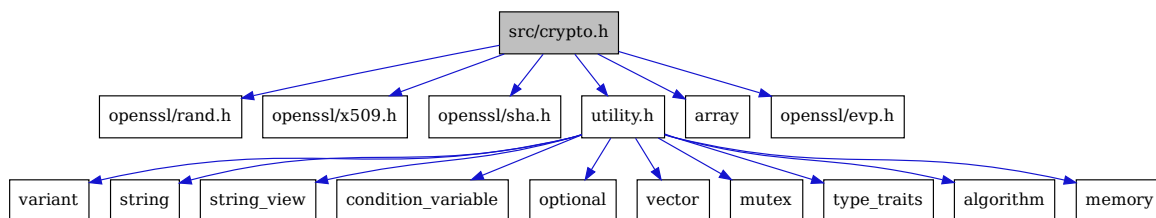
namespace **confighttp**

Variables

```
constexpr auto PORT_HTTPS = 1
```

20.2.5 crypto

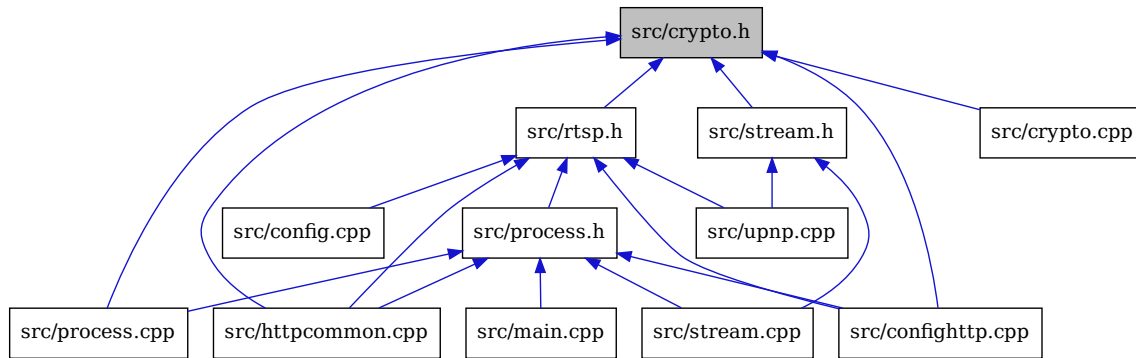
Include dependency graph for `crypto.h`:



This graph shows which files directly or indirectly include `crypto.h`:

`todo`

namespace **crypto**



Typedefs

```
using aes_t = std::vector<std::uint8_t>
```

```
using bignum_t = util::safe_ptr<BIGNUM, BN_free>
```

```
using bio_t = util::safe_ptr<BIO, BIO_free_all>
```

```
using cipher_ctx_t = util::safe_ptr<EVP_CIPHER_CTX, EVP_CIPHER_CTX_free>
```

```
using md_ctx_t = util::safe_ptr<EVP_MD_CTX, md_ctx_destroy>
```

```
using pkey_ctx_t = util::safe_ptr<EVP_PKEY_CTX, EVP_PKEY_CTX_free>
```

```
using pkey_t = util::safe_ptr<EVP_PKEY, EVP_PKEY_free>
```

```
using sha256_t = std::array<std::uint8_t, SHA256_DIGEST_LENGTH>
```

```
using x509_store_ctx_t = util::safe_ptr<X509_STORE_CTX, X509_STORE_CTX_free>
```

```
using x509_store_t = util::safe_ptr<X509_STORE, X509_STORE_free>
```

```
using x509_t = util::safe_ptr<X509, X509_free>
```

```
class cert_chain_t
```


Public Functions

void **add**(*x509_t* &&cert)

cert_chain_t()

cert_chain_t(*cert_chain_t*&&) noexcept = default

cert_chain_t &**operator**=(*cert_chain_t*&&) noexcept = default

const char ***verify**(*x509_t*::element_type *cert)

When certificates from two or more instances of Moonlight have been added to *x509_store_t*, only one of them will be verified by *X509_verify_cert*, resulting in only a single instance of Moonlight to be able to use Sunshine

To circumvent this, *x509_store_t* instance will be created for each instance of the certificates.

Private Members

x509_store_ctx_t **_cert_ctx**

std::vector<std::pair<*x509_t*, *x509_store_t*>> **_certs**

struct **creds_t**

Public Members

std::string **pkey**

std::string **x509**

namespace **cipher**

Functions

constexpr std::size_t **round_to_pkcs7_padded**(std::size_t size)

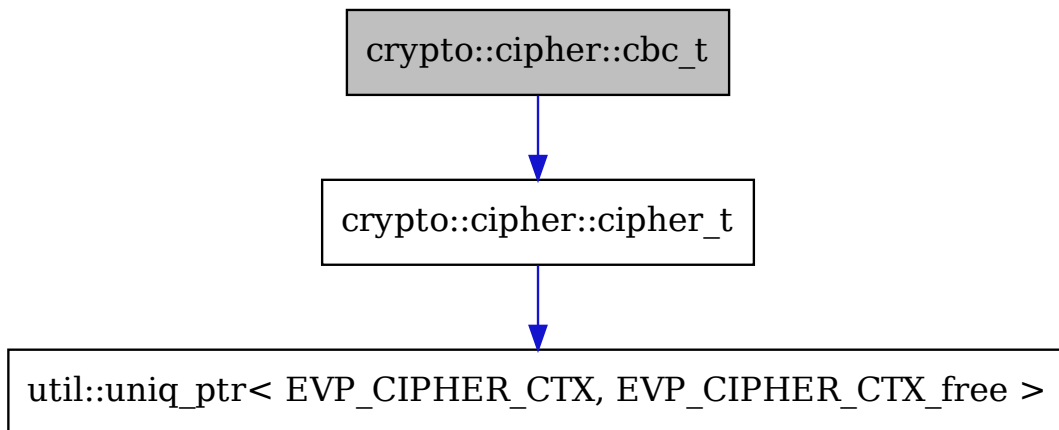
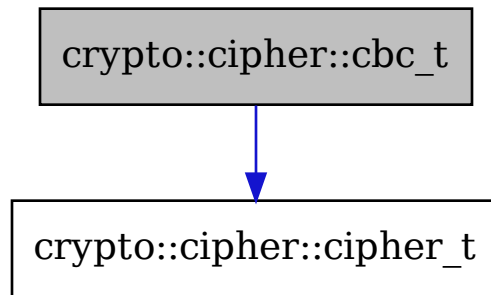
Variables

constexpr std::size_t **tag_size** = 16

class **cbc_t** : public *crypto::cipher::cipher_t*

Inheritance diagram for *crypto::cipher::cbc_t*:

Collaboration diagram for *crypto::cipher::cbc_t*:



Public Functions

cbc_t() = default

cbc_t(*cbc_t*&&) noexcept = default

cbc_t(const *crypto::aes_t* &key, bool padding = true)

int **encrypt**(const std::string_view &plaintext, std::uint8_t *cipher, *aes_t* *iv)

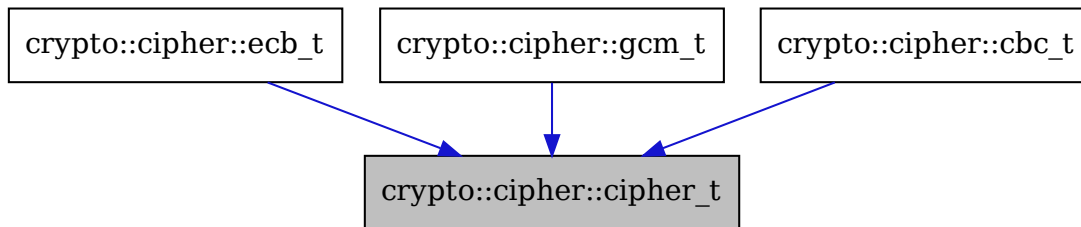
length of cipher must be at least: round_to_pkcs7_padded(plaintext.size())

return -1 on error return bytes written on success

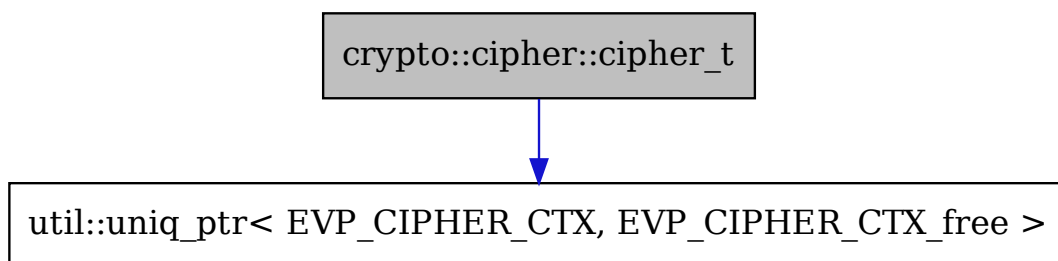
cbc_t &**operator**=(*cbc_t*&&) noexcept = default

class **cipher_t**

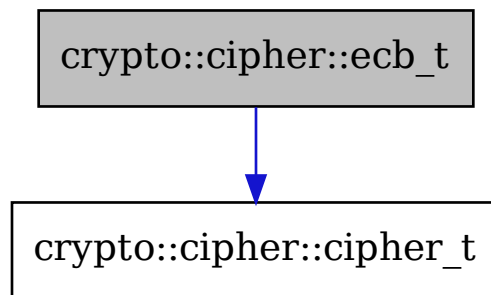
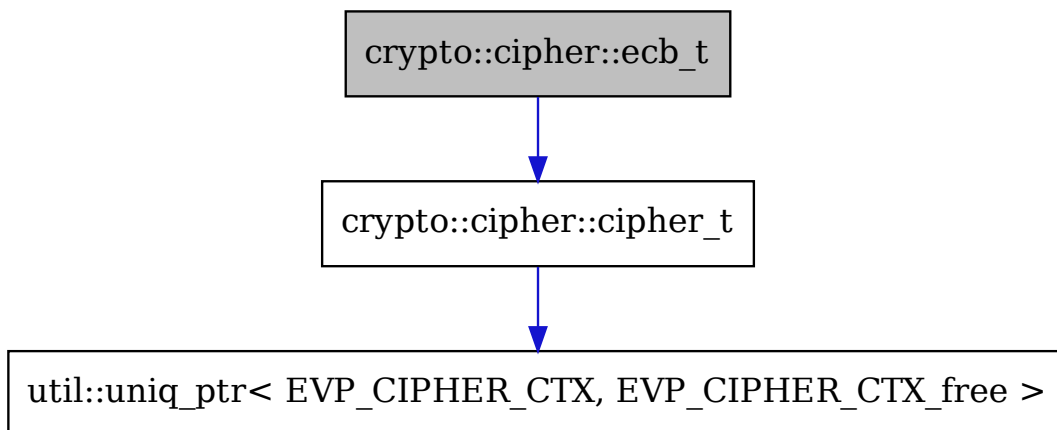
Inheritance diagram for `crypto::cipher::cipher_t`:



Collaboration diagram for `crypto::cipher::cipher_t`:



Subclassed by *crypto::cipher::cbc_t*, *crypto::cipher::ecb_t*, *crypto::cipher::gcm_t*

Public Members*cipher_ctx_t* **decrypt_ctx***cipher_ctx_t* **encrypt_ctx***aes_t* **key**bool **padding**class **ecb_t** : public *crypto::cipher::cipher_t*Inheritance diagram for *crypto::cipher::ecb_t*:Collaboration diagram for *crypto::cipher::ecb_t*:

Public Functions

int **decrypt**(const std::string_view &cipher, std::vector<std::uint8_t> &plaintext)

ecb_t() = default

ecb_t(const *aes_t* &key, bool padding = true)

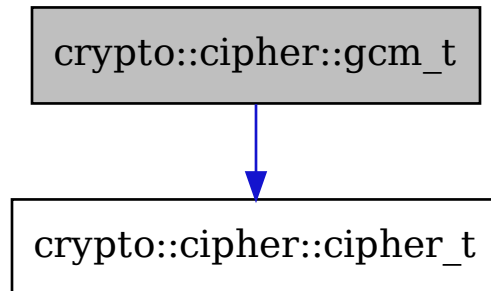
ecb_t(*ecb_t*&&) noexcept = default

int **encrypt**(const std::string_view &plaintext, std::vector<std::uint8_t> &cipher)

ecb_t &**operator**=(*ecb_t*&&) noexcept = default

class **gcm_t** : public *crypto::cipher::cipher_t*

Inheritance diagram for crypto::cipher::gcm_t:



Collaboration diagram for crypto::cipher::gcm_t:

Public Functions

int **decrypt**(const std::string_view &cipher, std::vector<std::uint8_t> &plaintext, *aes_t* *iv)

int **encrypt**(const std::string_view &plaintext, std::uint8_t *tagged_cipher, *aes_t* *iv)

length of cipher must be at least: round_to_pkcs7_padded(plaintext.size()) +
crypto::cipher::tag_size

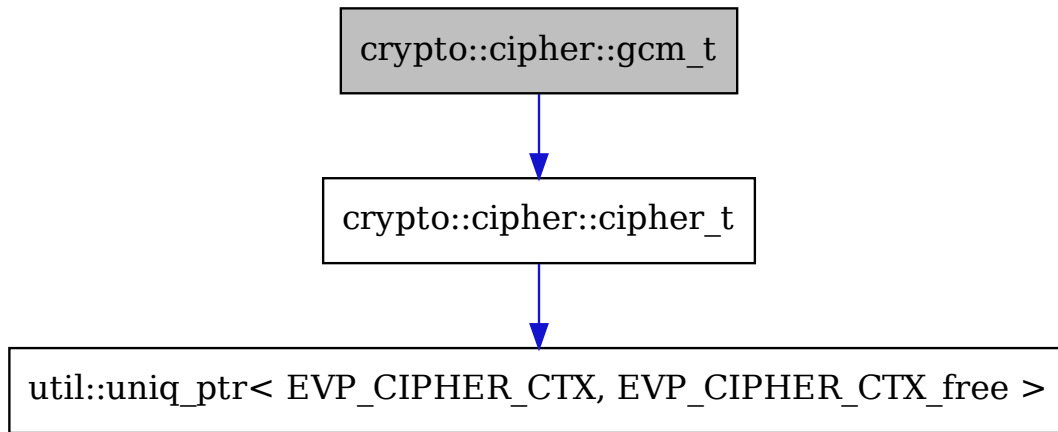
return -1 on error return bytes written on success

gcm_t() = default

gcm_t(const *crypto::aes_t* &key, bool padding = true)

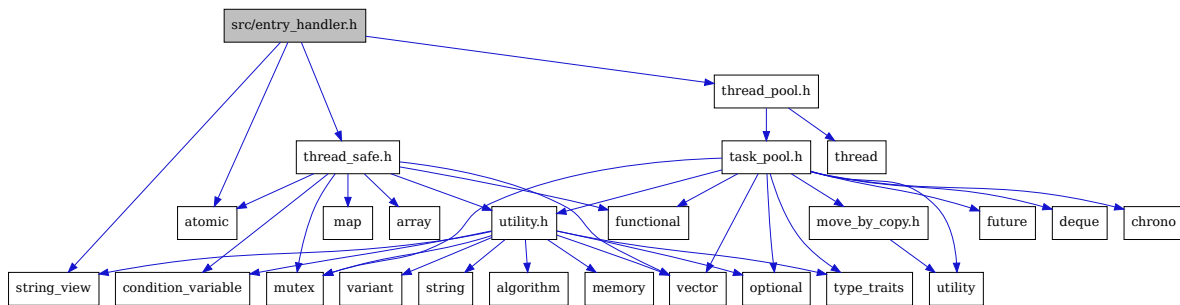
gcm_t(*gcm_t*&&) noexcept = default

gcm_t &**operator**=(*gcm_t*&&) noexcept = default

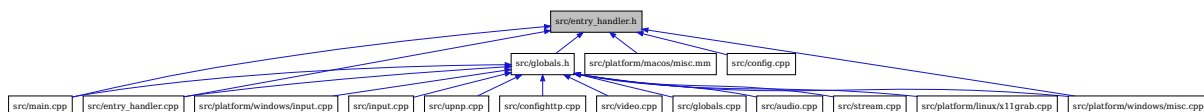


20.2.6 entry_handler

Include dependency graph for entry_handler.h:



This graph shows which files directly or indirectly include entry_handler.h:



Header file for entry point functions.

Functions

void **launch_ui**()

Launch the Web UI.

EXAMPLES:

```
launch_ui();
```

void **launch_ui_with_path**(std::string path)

Launch the Web UI at a specific endpoint.

EXAMPLES:

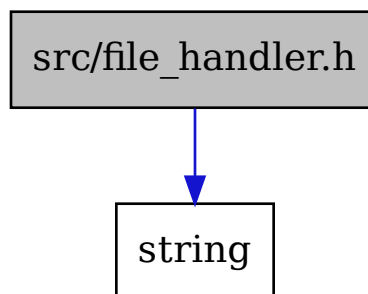
```
launch_ui_with_path("/pin");
```

namespace **args**

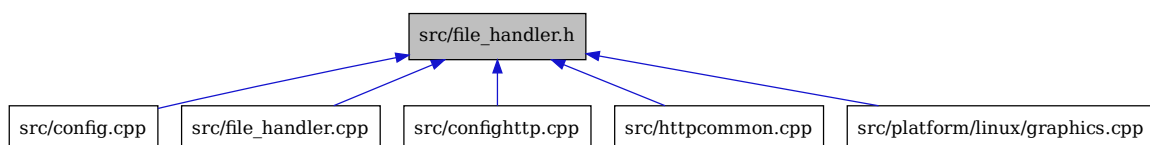
namespace **lifetime**

20.2.7 file_handler

Include dependency graph for file_handler.h:



This graph shows which files directly or indirectly include file_handler.h:

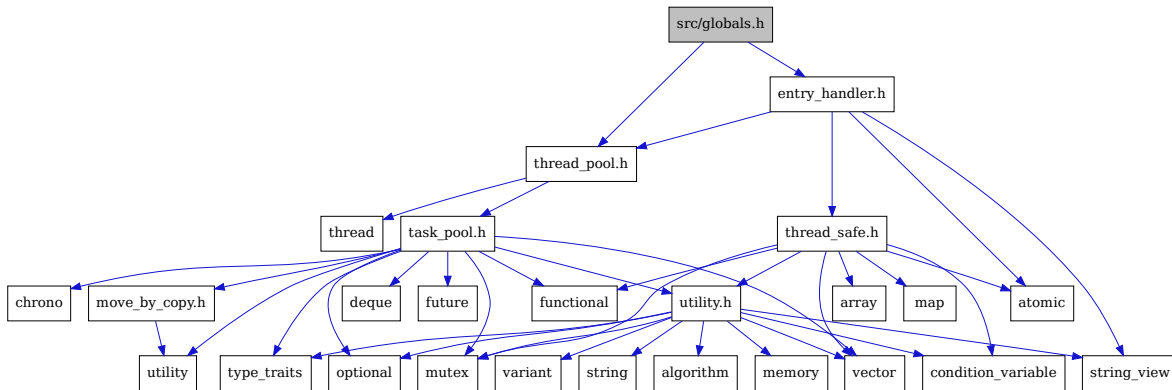


Header file for file handling functions.

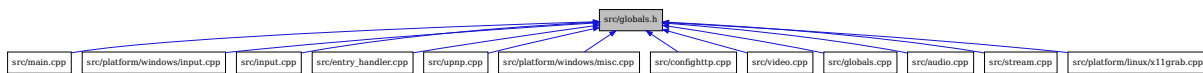
namespace **file_handler**

20.2.8 globals

Include dependency graph for globals.h:



This graph shows which files directly or indirectly include globals.h:



Header for globally accessible variables and functions.

Defines

MAIL(x)

Variables

bool **display_cursor**

A boolean flag to indicate whether the cursor should be displayed.

thread_pool_util::ThreadPool **task_pool**

A thread pool for processing tasks.

namespace **mail**

Variables

```
constexpr auto audio_packets = std::string_view{"audio_packets"}
```

```
constexpr auto broadcast_shutdown = std::string_view{"broadcast_shutdown"}
```

```
constexpr auto gamepad_feedback = std::string_view{"gamepad_feedback"}
```

```
constexpr auto hdr = std::string_view{"hdr"}
```

```
constexpr auto idr = std::string_view{"idr"}
```

```
constexpr auto invalidate_ref_frames = std::string_view{"invalidate_ref_frames"}
```

safe::mail_t **man**

A process-wide communication mechanism.

```
constexpr auto shutdown = std::string_view{"shutdown"}
```

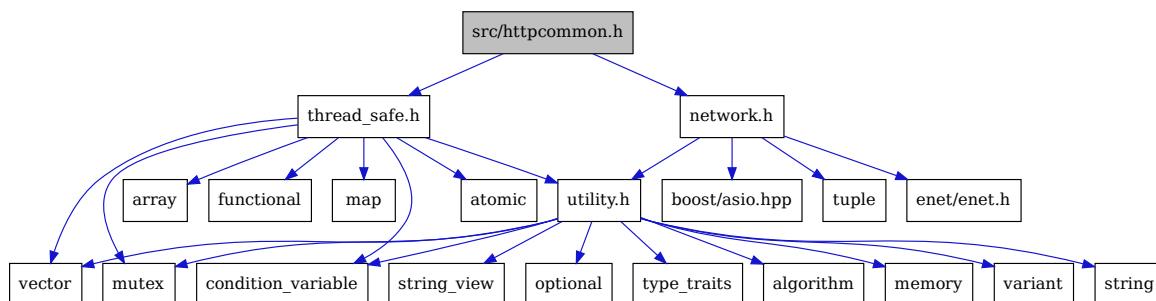
```
constexpr auto switch_display = std::string_view{"switch_display"}
```

```
constexpr auto touch_port = std::string_view{"touch_port"}
```

```
constexpr auto video_packets = std::string_view{"video_packets"}
```

20.2.9 httpcommon

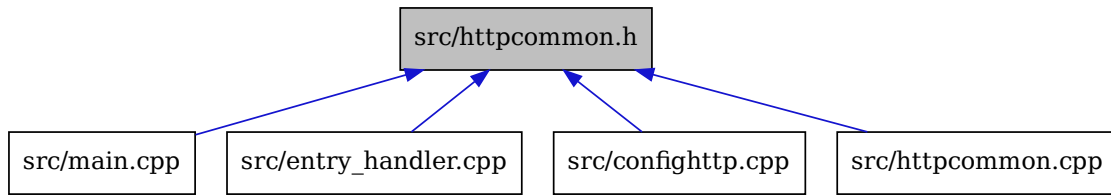
Include dependency graph for httpcommon.h:



This graph shows which files directly or indirectly include httpcommon.h:

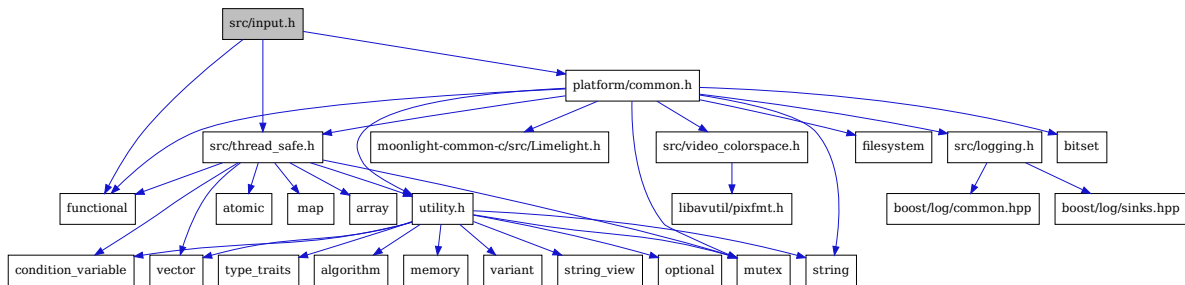
todo

namespace **http**



20.2.10 input

Include dependency graph for input.h:



This graph shows which files directly or indirectly include input.h:



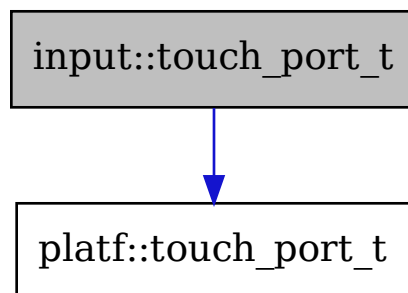
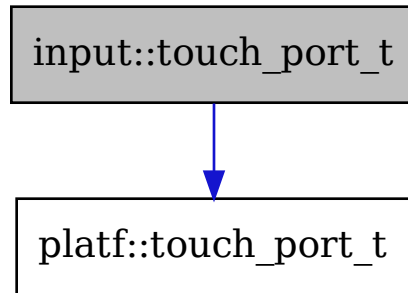
todo

namespace **input**

```
struct touch_port_t : public platf::touch_port_t
```

Inheritance diagram for `input::touch_port_t`:

Collaboration diagram for `input::touch_port_t`:



Public Members

float **client_offsetX**

float **client_offsetY**

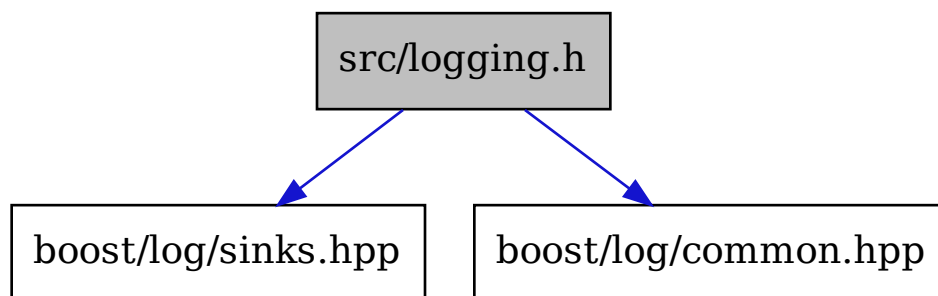
int **env_height**

int **env_width**

float **scalar_inv**

20.2.11 logging

Include dependency graph for logging.h:



This graph shows which files directly or indirectly include logging.h:



Logging header file for the Sunshine application.

Typedefs

using **text_sink** = boost::log::sinks::asynchronous_sink<boost::log::sinks::text_ostream_backend>

Functions

void **log_flush**()

Flush the log.

EXAMPLES:

```
log_flush();
```

void **print_help**(const char *name)

Print help to stdout.

EXAMPLES:

```
print_help("sunshine");
```

Parameters

name – The name of the program.

Variables

boost::log::sources::severity_logger<int> **debug**

boost::log::sources::severity_logger<int> **error**

boost::log::sources::severity_logger<int> **fatal**

boost::log::sources::severity_logger<int> **info**

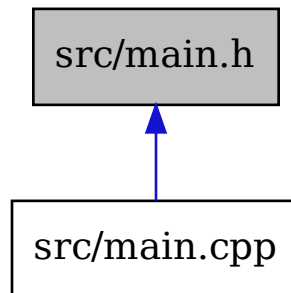
boost::shared_ptr<boost::log::sinks::asynchronous_sink<boost::log::sinks::text_ostream_backend>> **sink**

boost::log::sources::severity_logger<int> **verbose**

boost::log::sources::severity_logger<int> **warning**

20.2.12 main

This graph shows which files directly or indirectly include main.h:



Main header file for the Sunshine application.

Functions

int **main**(int argc, char *argv[])

Main application entry point.

EXAMPLES:

```
main(1, const char* args[] = {"sunshine", nullptr});
```

Parameters

- **argc** – The number of arguments.
- **argv** – The arguments.

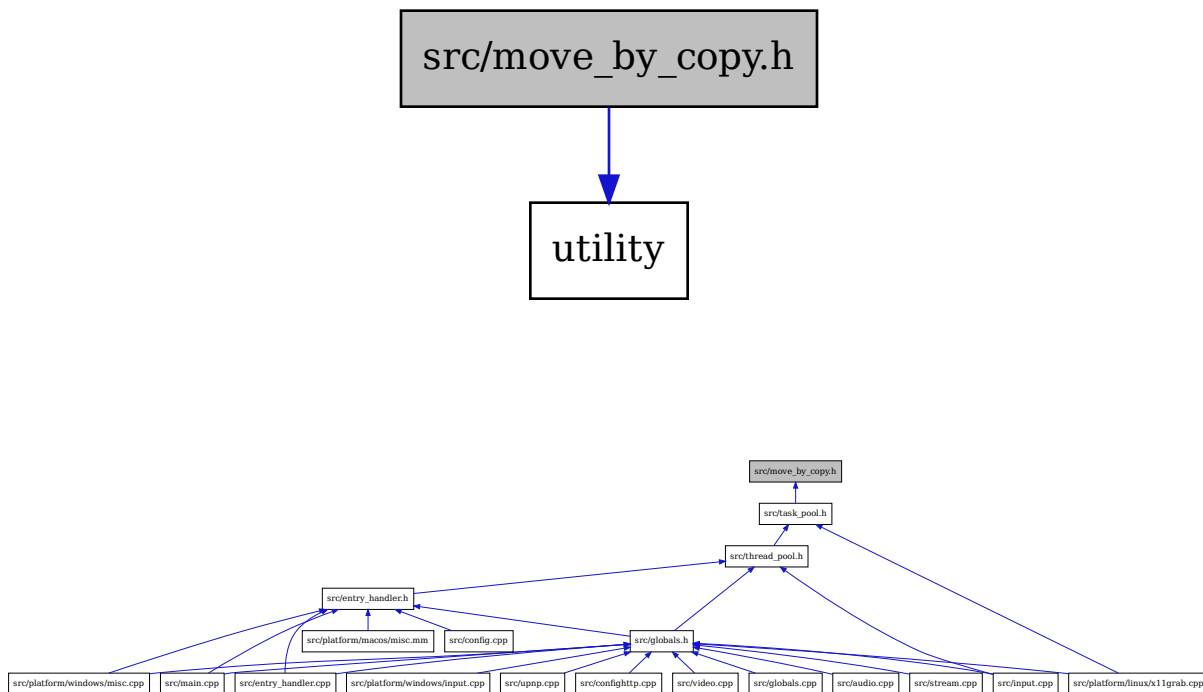
20.2.13 move_by_copy

Include dependency graph for move_by_copy.h:

This graph shows which files directly or indirectly include move_by_copy.h:

todo

namespace **move_by_copy_util**



Functions

```
template<class T>
MoveByCopy<T> cmove(T &movable)
```

```
template<class T>
MoveByCopy<T> const_cmove(const T &movable)
```

```
template<class T>
```

```
class MoveByCopy
```

#include <src/move_by_copy.h> When a copy is made, it moves the object This allows you to move an object when a move can't be done.

Public Types

```
typedef T move_type
```

Public Functions

```
inline MoveByCopy(const MoveByCopy &other)
```

```
inline explicit MoveByCopy(move_type &&to_move)
```

```
MoveByCopy(MoveByCopy &&other) = default
```

```
inline operator move_type()
```

```
inline MoveByCopy &operator=(const MoveByCopy &other)
```

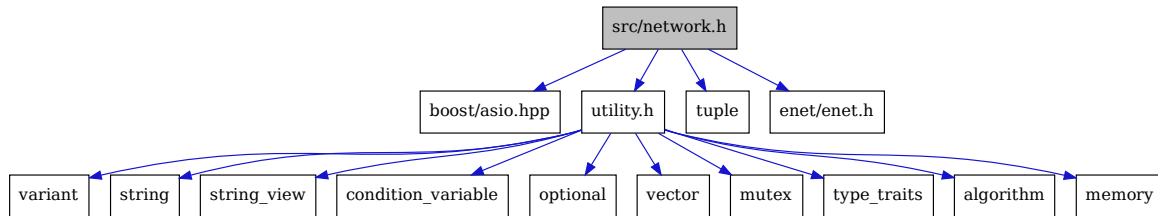
```
MoveByCopy &operator=(MoveByCopy &&other) = default
```


Private Members

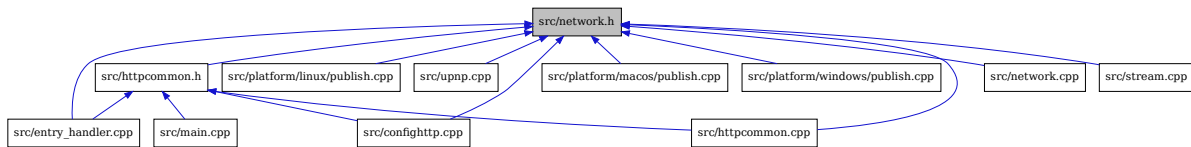
move_type _to_move

20.2.14 network

Include dependency graph for network.h:



This graph shows which files directly or indirectly include network.h:



todo

namespace **net**

Typedefs

using **host_t** = util::safe_ptr<ENetHost, free_host>

using **packet_t** = util::safe_ptr<ENetPacket, enet_packet_destroy>

using **peer_t** = ENetPeer*

Enums

enum **af_e**

Values:

enumerator **IPV4**

enumerator **BOTH**

enum **net_e**

Values:

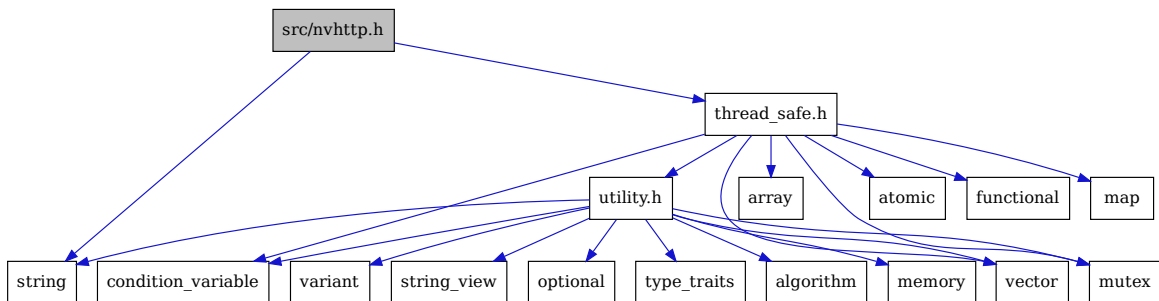
enumerator **PC**

enumerator **LAN**

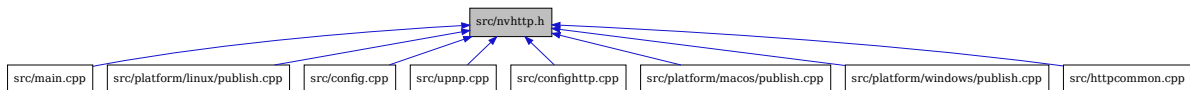
enumerator **WAN**

20.2.15 nvhttp

Include dependency graph for nvhttp.h:



This graph shows which files directly or indirectly include `nvhttp.h`:



todo

namespace **nvhttp**

This namespace contains all the functions and variables related to the `nvhttp` (GameStream) server.

Defines

`__kernel_entry`

namespace **proc**

Typedefs

typedef *config::prep_cmd_t* **cmd_t**

using **file_t** = util::safe_ptr_v2<FILE, int, fclose>

struct **ctx_t**

#include <src/process.h> **pre_cmds** — guaranteed to be executed unless any of the commands fail. detached — commands detached from Sunshine **cmd** — Runs indefinitely until: No session is running and a different set of commands it to be executed Command exits **working_dir** — the process working directory. This is required for some games to run properly. **cmd_output** — empty — The output of the commands are appended to the output of sunshine “null” — The output of the commands are discarded **filename** — The output of the commands are appended to filename

Public Members

bool **auto_detach**

std::string **cmd**

std::vector<std::string> **detached**

Some applications, such as Steam, either exit quickly, or keep running indefinitely.

Apps that launch normal child processes and terminate will be handled by the process grouping logic (**wait_all**). However, apps that launch child processes indirectly or into another process group (such as UWP apps) can only be handled by the auto-detach heuristic which catches processes that exit 0 very quickly, but we won’t have proper process tracking for those.

For cases where users just want to kick off a background process and never manage the lifetime of that process, they can use detached commands for that.

bool **elevated**

std::chrono::seconds **exit_timeout**

std::string **id**

std::string **image_path**

```

std::string name

std::string output

std::vector<cmd_t> prep_cmds

bool wait_all

std::string working_dir

```

```
class proc_t
```

Public Functions

```

int execute(int app_id, std::shared_ptr<rtsp_stream::launch_session_t> launch_session)

std::string get_app_image(int app_id)

std::vector<ctx_t> &get_apps()

const std::vector<ctx_t> &get_apps() const

std::string get_last_run_app_name()

proc_t &operator=(proc_t&&) = default

proc_t() = default

inline proc_t(boost::process::environment &&env, std::vector<ctx_t> &&apps)

proc_t(proc_t&&) = default

int running()
    Returns
        _app_id if a process is running, otherwise returns 0

void terminate()

~proc_t()

```

Private Members

```

ctx_t _app

int _app_id

std::chrono::steady_clock::time_point _app_launch_time

```

```
std::vector<cmd_t>::const_iterator _app_prep_begin
```

```
std::vector<cmd_t>::const_iterator _app_prep_it
```

```
std::vector<ctx_t> _apps
```

```
boost::process::environment _env
```

```
file_t _pipe
```

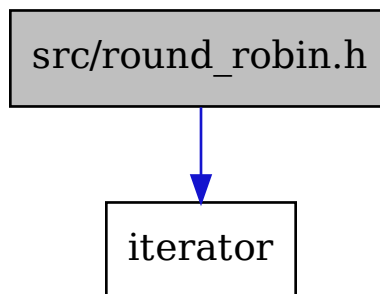
```
boost::process::child _process
```

```
boost::process::group _process_group
```

```
bool placebo = { }
```

20.2.17 round_robin

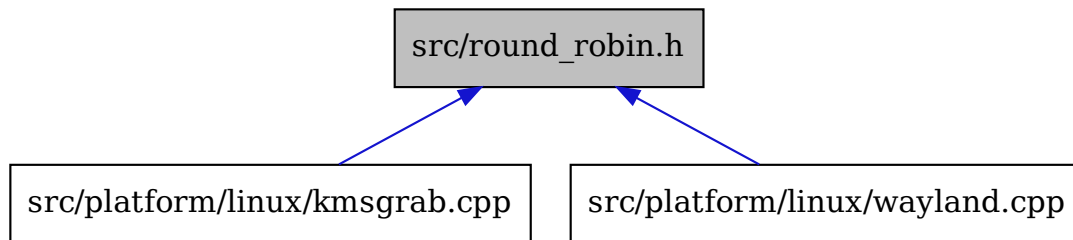
Include dependency graph for round_robin.h:



This graph shows which files directly or indirectly include round_robin.h:

todo

namespace **round_robin_util**



Functions

```
template<class V, class It>
round_robin_t<V, It> make_round_robin(It begin, It end)
```

```
template<class V, class T>
class it_wrap_t
```

Public Types

```
using const_pointer = V const*
```

```
using const_reference = V const&
```

```
typedef std::ptrdiff_t diff_t
```

```
using difference_type = V
```

```
typedef T iterator
```

```
using iterator_category = std::random_access_iterator_tag
```

```
using pointer = V*
```

```
using reference = V&
```

```
using value_type = V
```

Public Functions

```
inline bool operator!=(const iterator &other) const  
  
inline reference operator*()  
  
inline const_reference operator*() const  
  
inline iterator operator+(diff_t step)  
  
inline iterator operator++()  
  
inline iterator operator++(int)  
  
inline iterator operator+=(diff_t step)  
  
inline iterator operator-(diff_t step)  
  
inline diff_t operator-(iterator first)  
  
inline iterator operator--()  
  
inline iterator operator--(int)  
  
inline iterator operator--(diff_t step)  
  
inline pointer operator->()  
  
inline const_pointer operator->() const  
  
inline bool operator<(const iterator &other) const  
  
inline bool operator<=(const iterator &other) const  
  
inline bool operator==(const iterator &other) const  
  
inline bool operator>(const iterator &other) const  
  
inline bool operator>=(const iterator &other) const
```

Private Functions

```
inline iterator &_this()  
  
inline const iterator &_this() const  
  
template<class V, class It>  
class round_robin_t : public round_robin_util::it_wrap_t<V, round_robin_t<V, It>>  
    Inheritance diagram for round_robin_util::round_robin_t:  
    Collaboration diagram for round_robin_util::round_robin_t:
```



```
graph TD; A[round_robin_util::round_robin_t< V, It >] --> B[round_robin_util::it_wrap_t< V, round_robin_t< V, It > >];
```

`round_robin_util::round_robin_t< V, It >`

`round_robin_util::it_wrap_t< V, round_robin_t< V, It > >`

```
graph TD; A[round_robin_util::round_robin_t< V, It >] --> B[round_robin_util::it_wrap_t< V, round_robin_t< V, It > >];
```

`round_robin_util::round_robin_t< V, It >`

`round_robin_util::it_wrap_t< V, round_robin_t< V, It > >`

Public Types

```
using iterator = It
```

```
using pointer = V*
```

Public Functions

```
inline void dec()
```

```
inline bool eq(const round_robin_t &other) const
```

```
inline pointer get() const
```

```
inline void inc()
```

```
inline round_robin_t(iterator begin, iterator end)
```

Private Members

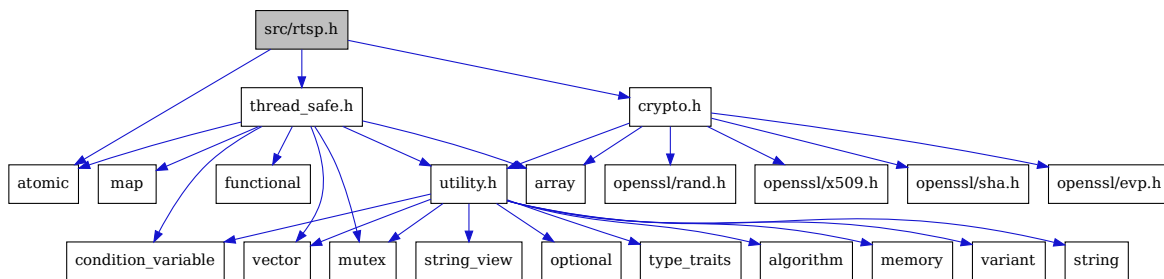
```
It _begin
```

```
It _end
```

```
It _pos
```

20.2.18 rtsp

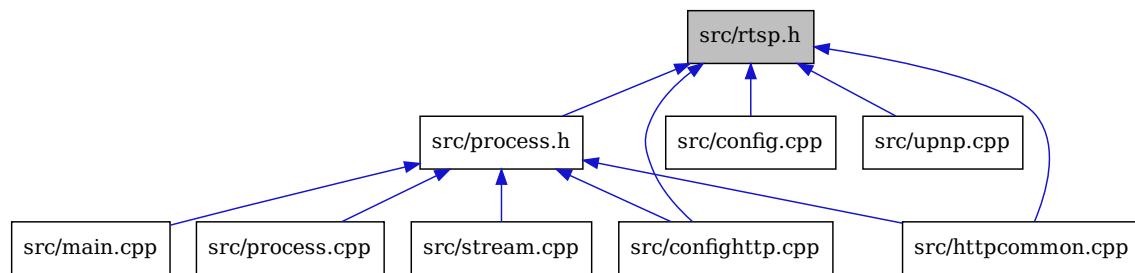
Include dependency graph for rtsp.h:



This graph shows which files directly or indirectly include `rtsp.h`:

```
todo
```

```
namespace rtsp_stream
```



Variables

```
constexpr auto RTSP_SETUP_PORT = 21
```

```
struct launch_session_t
```

Public Members

```
int appid
```

```
std::string av_ping_payload
```

```
uint32_t control_connect_data
```

```
bool enable_hdr
```

```
bool enable_sops
```

```
int fps
```

```
crypto::aes_t gcm_key
```

```
int gcmmap
```

```
int height
```

```
bool host_audio
```

```
uint32_t id
```

```
crypto::aes_t iv
```

```
std::optional<crypto::cipher::gcm_t> rtsp_cipher
```

```
uint32_t rtsp_iv_counter
```

```
std::string rtsp_url_scheme
```

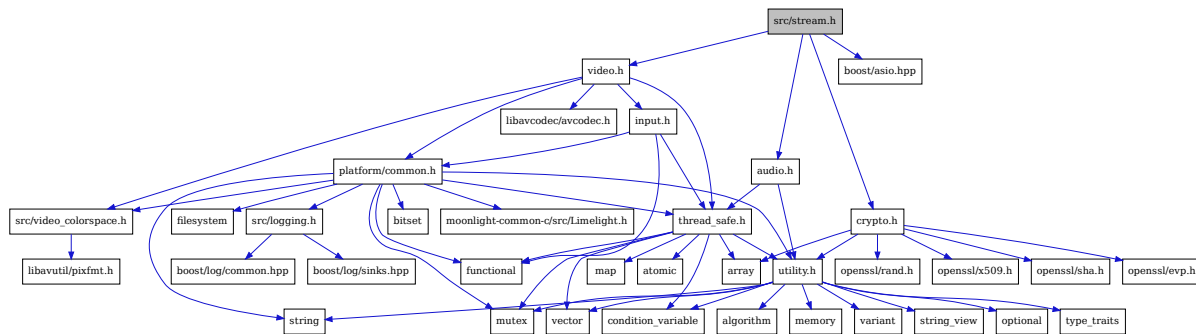
```
int surround_info
```

```
std::string unique_id
```

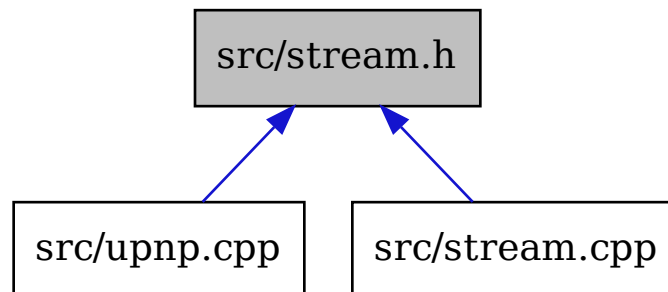
```
int width
```

20.2.19 stream

Include dependency graph for stream.h:



This graph shows which files directly or indirectly include stream.h:



todo

namespace **stream**

Variables

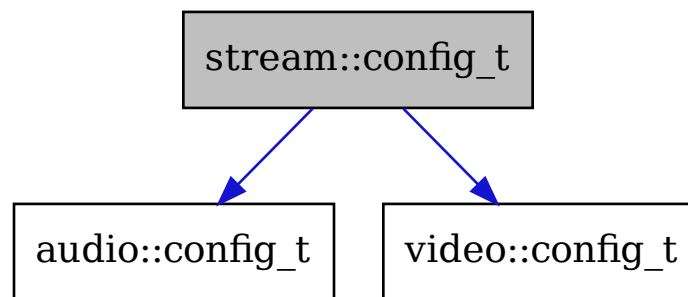
```
constexpr auto AUDIO_STREAM_PORT = 11
```

```
constexpr auto CONTROL_PORT = 10
```

```
constexpr auto VIDEO_STREAM_PORT = 9
```

```
struct config_t
```

Collaboration diagram for stream::config_t:



Public Members

audio::config_t **audio**

int **audioQosType**

int **controlProtocolType**

uint32_t **encryptionFlagsEnabled**

std::optional<int> **gcmap**

int **minRequiredFecPackets**

```
int mlFeatureFlags
```

```
video::config_t monitor
```

```
int packetSize
```

```
int videoQosType
```

```
namespace session
```

Enums

```
enum class state_e : int
```

```
    Values:
```

```
    enumerator STOPPED
```

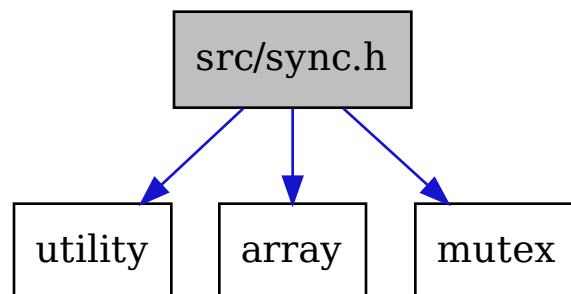
```
    enumerator STOPPING
```

```
    enumerator STARTING
```

```
    enumerator RUNNING
```

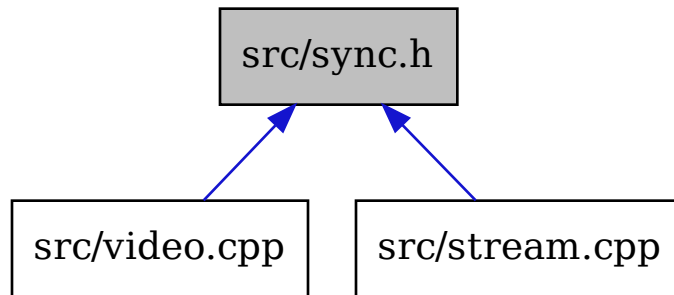
20.2.20 sync

Include dependency graph for sync.h:



This graph shows which files directly or indirectly include sync.h:

todo



namespace **sync_util**

template<class **T**, class **M** = std::mutex>

class **sync_t**

Public Types

using **mutex_t** = *M*

using **value_t** = *T*

Public Functions

inline std::lock_guard<*mutex_t*> **lock**()

inline *value_t* &**operator***()

inline const *value_t* &**operator***() const

inline *value_t* ***operator->**()

inline *sync_t* &**operator**=(const *value_t* &val) noexcept

inline *sync_t* &**operator**=(*sync_t* &&other) noexcept

inline *sync_t* &**operator**=(*sync_t* &other) noexcept

template<class **V**>

inline *sync_t* &**operator**=(*V* &&val)

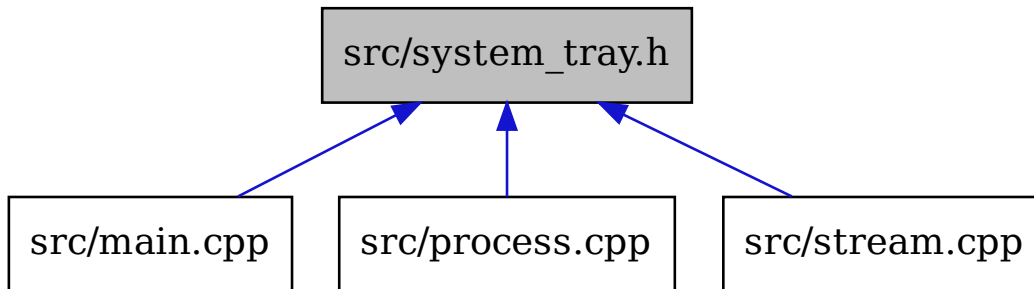
inline *sync_t* &**operator**=(*value_t* &&val) noexcept

template<class ...**Args**>

inline **sync_t**(*Args*&&... args)

Public Members*value_t* **raw****Private Members***mutex_t* **_lock****20.2.21 system_tray**

This graph shows which files directly or indirectly include system_tray.h:



todo

namespace **system_tray**

Functions

int **end_tray**()

int **run_tray**()

int **system_tray**()

void **tray_donate_github_cb**(struct tray_menu *item)

void **tray_donate_mee6_cb**(struct tray_menu *item)

void **tray_donate_patreon_cb**(struct tray_menu *item)

void **tray_donate_paypal_cb**(struct tray_menu *item)

void **tray_open_ui_cb**(struct tray_menu *item)


```

void tray_quit_cb(struct tray_menu *item)

void update_tray_pausing(std::string app_name)

void update_tray_playing(std::string app_name)

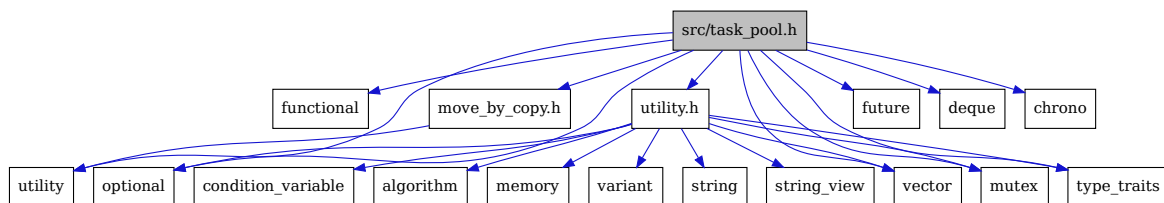
void update_tray_require_pin()

void update_tray_stopped(std::string app_name)

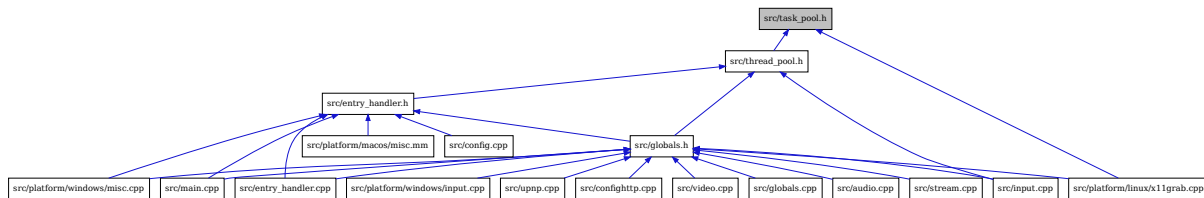
```

20.2.22 task_pool

Include dependency graph for task_pool.h:



This graph shows which files directly or indirectly include task_pool.h:



todo

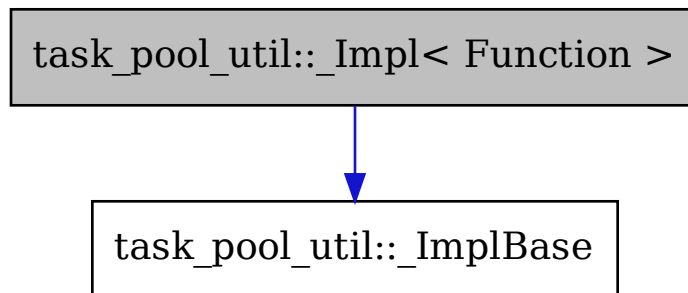
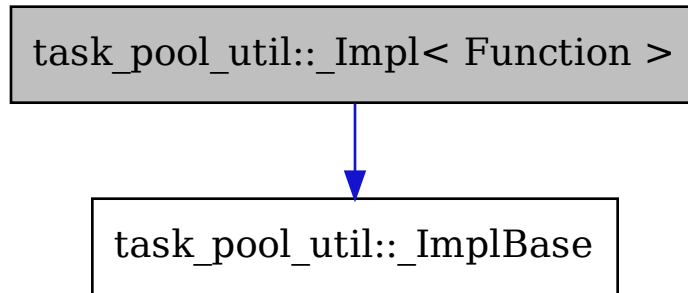
namespace **task_pool_util**

```
template<class Function>
```

```
class _Impl : public task_pool_util::_ImplBase
```

Inheritance diagram for task_pool_util::_Impl:

Collaboration diagram for task_pool_util::_Impl:



Public Functions

```
inline _Impl(Function &&f)
```

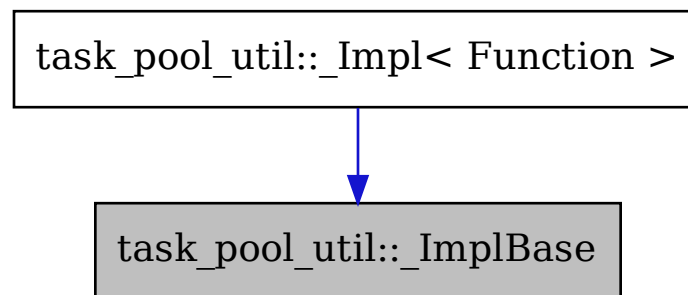
```
inline virtual void run() override
```

Private Members

```
Function _func
```

```
class _ImplBase
```

Inheritance diagram for task_pool_util::_ImplBase:



Subclassed by *task_pool_util::_Impl< Function >*

Public Functions

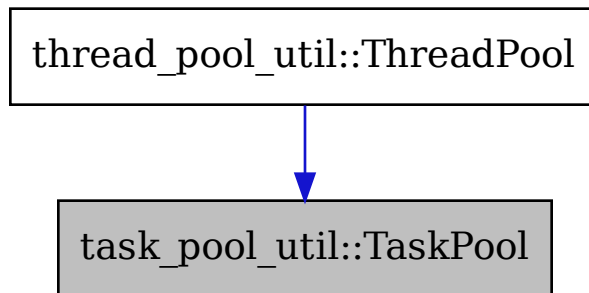
```
virtual void run() = 0
```

```
inline virtual ~_ImplBase() = default
```

```
class TaskPool
```

Inheritance diagram for task_pool_util::TaskPool:

Subclassed by *thread_pool_util::ThreadPool*



Public Types

```
typedef std::unique_ptr<_ImplBase> __task
```

```
typedef std::chrono::steady_clock::time_point __time_point
```

```
typedef _ImplBase *task_id_t
```

Public Functions

```
inline bool cancel(task_id_t task_id)
```

```
template<class X, class Y>
```

```
inline void delay(task_id_t task_id, std::chrono::duration<X, Y> duration)
```

Parameters

- **task_id** – The id of the task to delay.
- **duration** – The delay before executing the task.

```
inline std::optional<__time_point> next()
```

```
inline TaskPool &operator=(TaskPool &&other) noexcept
```

```
inline std::optional<__task> pop()
```

```
inline std::optional<std::pair<__time_point, __task>> pop(task_id_t task_id)
```

```
template<class Function, class ...Args>
```

```
inline auto push(Function &&newTask, Args&&... args)
```

```
template<class Function, class X, class Y, class ...Args>
```

```
inline auto pushDelayed(Function &&newTask, std::chrono::duration<X, Y> duration, Args&&... args)
```

Returns

an id to potentially delay the task.

```
inline void pushDelayed(std::pair<__time_point, __task> &&task)
```

```

inline bool ready()

TaskPool() = default

inline TaskPool(TaskPool &&other) noexcept

```

Protected Attributes

```

std::mutex _task_mutex

std::deque<__task> _tasks

std::vector<std::pair<__time_point, __task>> _timer_tasks

```

Private Functions

```

template<class Function>
inline std::unique_ptr<__ImplBase> toRunnable(Function &&f)

```

```

template<class R>

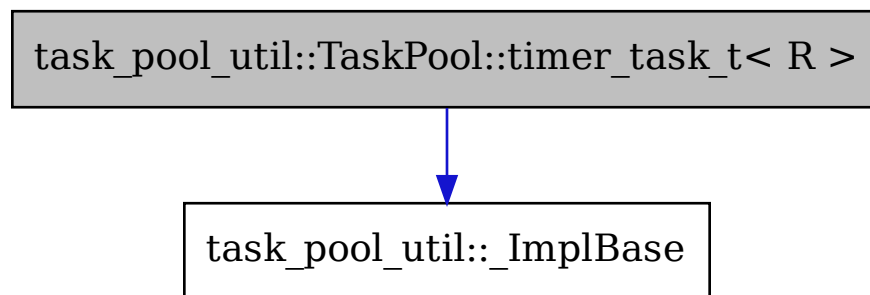
```

```

class timer_task_t

```

Collaboration diagram for task_pool_util::TaskPool::timer_task_t:



Public Functions

```

inline timer_task_t(task_id_t task_id, std::future<R> &future)

```

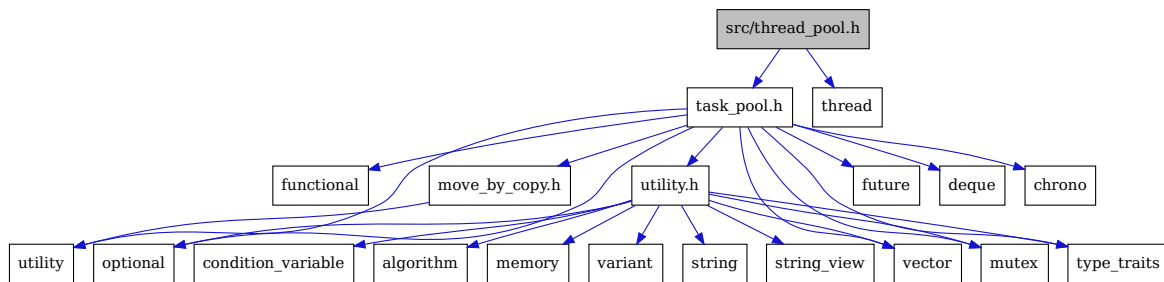
Public Members

`std::future<R> future`

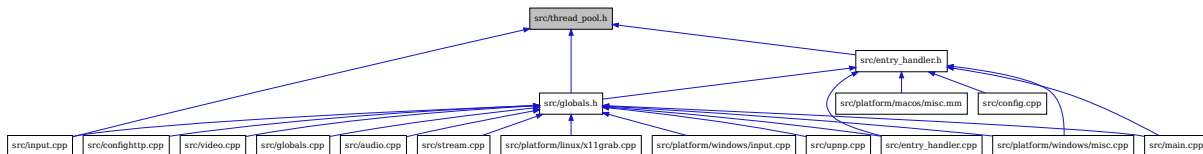
`task_id_t task_id`

20.2.23 thread_pool

Include dependency graph for thread_pool.h:



This graph shows which files directly or indirectly include thread_pool.h:



todo

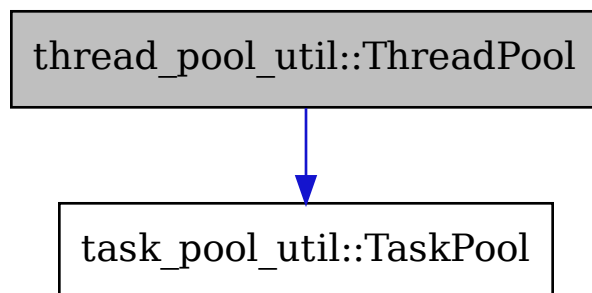
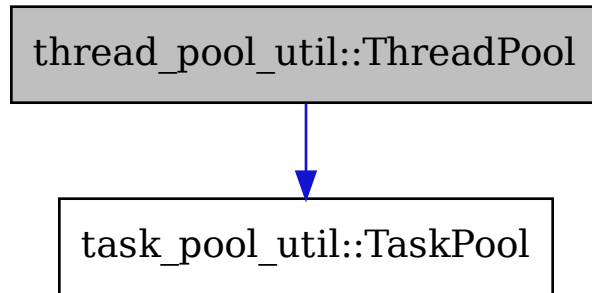
namespace **thread_pool_util**

class **ThreadPool** : public *task_pool_util::TaskPool*

`#include <src/thread_pool.h>` Inheritance diagram for thread_pool_util::ThreadPool:

Collaboration diagram for thread_pool_util::ThreadPool:

Allow threads to execute unhindered while keeping full control over the threads.



Public Types

```
typedef TaskPool::__task __task
```

Public Functions

```
inline void _main()
```

```
inline void join()
```

```
template<class Function, class ...Args>
```

```
inline auto push(Function &&newTask, Args&&... args)
```

```
template<class Function, class X, class Y, class ...Args>
```

```
inline auto pushDelayed(Function &&newTask, std::chrono::duration<X, Y> duration, Args&&... args)
```

```
inline void pushDelayed(std::pair<__time_point, __task> &&task)
```

```
inline void start(int threads)
```

```
inline void stop()
```

```
inline ThreadPool()
```

```
inline explicit ThreadPool(int threads)
```

```
inline ~ThreadPool() noexcept
```

Private Members

```
bool _continue
```

```
std::condition_variable _cv
```

```
std::mutex _lock
```

```
std::vector<std::thread> _thread
```

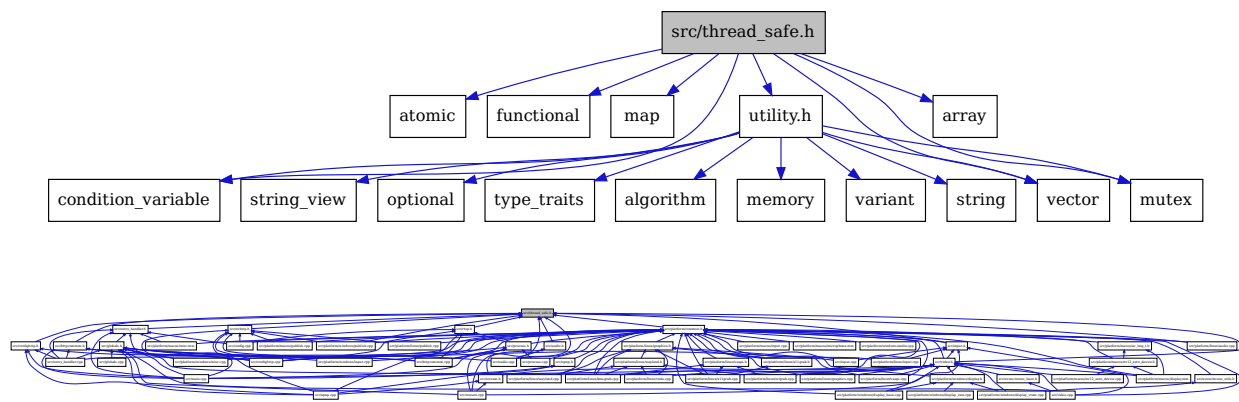
20.2.24 thread_safe

Include dependency graph for thread_safe.h:

This graph shows which files directly or indirectly include thread_safe.h:

todo

namespace **safe**



Typedefs

```
template<class T>
using alarm_t = std::shared_ptr<alarm_raw_t<T>>

using mail_t = std::shared_ptr<mail_raw_t>

using signal_t = event_t<bool>
```

Functions

```
inline void cleanup(mail_raw_t*)

template<class T>
inline auto lock(const std::weak_ptr<void> &wp)

template<class T>
alarm_t<T> make_alarm()

template<class T, class F_Construct, class F_Destruct>
auto make_shared(F_Construct &&fc, F_Destruct &&fd)

template<class T>
class alarm_raw_t
```

Public Types

```
using status_t = util::optional_t<T>
```

Public Functions

```
inline void reset()

inline void ring(const status_t &status)

inline void ring(status_t &&status)

inline status_t &status()

inline const status_t &status() const

inline auto wait()

template<class Pred>
inline auto wait(Pred &&pred)

template<class Rep, class Period>
inline auto wait_for(const std::chrono::duration<Rep, Period> &rel_time)

template<class Rep, class Period, class Pred>
inline auto wait_for(const std::chrono::duration<Rep, Period> &rel_time, Pred &&pred)

template<class Rep, class Period>
inline auto wait_until(const std::chrono::duration<Rep, Period> &rel_time)

template<class Rep, class Period, class Pred>
inline auto wait_until(const std::chrono::duration<Rep, Period> &rel_time, Pred &&pred)
```

Private Members

```
std::condition_variable _cv

std::mutex _lock

bool _rang = {false}

status_t _status = {util::false_v<status_t>}

template<class T>
class event_t
```

Public Types

```
using status_t = util::optional_t<T>
```

Public Functions

```

inline bool peek()

inline status_t pop()

template<class Rep, class Period>
inline status_t pop(std::chrono::duration<Rep, Period> delay)

template<class ...Args>
inline void raise(Args&&... args)

inline void reset()

inline bool running() const

inline void stop()

inline status_t view()

template<class Rep, class Period>
inline status_t view(std::chrono::duration<Rep, Period> delay)

```

Private Members

```

bool _continue = {true}

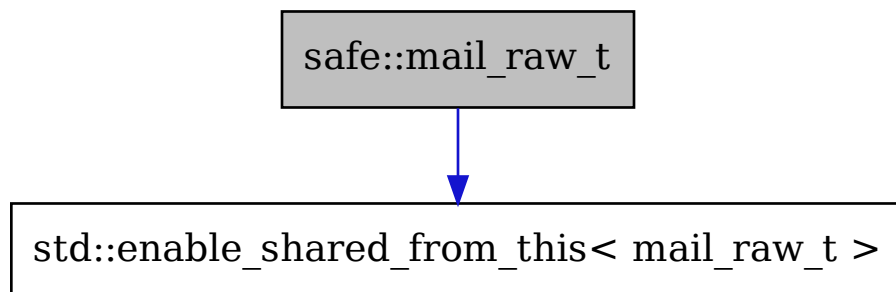
std::condition_variable _cv

std::mutex _lock

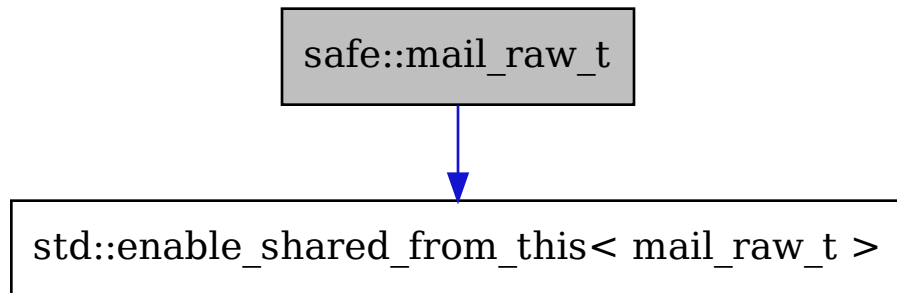
status_t _status = {util::false_v<status_t>}

class mail_raw_t : public std::enable_shared_from_this<mail_raw_t>
  Inheritance diagram for safe::mail_raw_t:

```



Collaboration diagram for `safe::mail_raw_t`:



Public Types

```
template<class T>
using event_t = std::shared_ptr<post_t<event_t<T>>>>

template<class T>
using queue_t = std::shared_ptr<post_t<queue_t<T>>>>
```

Public Functions

```
inline void cleanup()

template<class T>
inline event_t<T> event(const std::string_view &id)

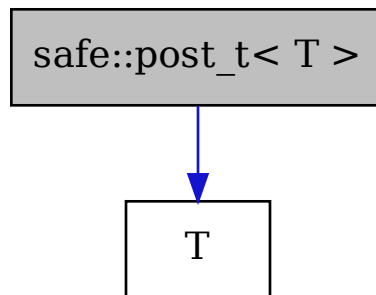
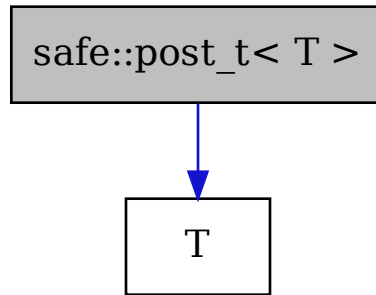
template<class T>
inline queue_t<T> queue(const std::string_view &id)
```

Public Members

```
std::map<std::string, std::weak_ptr<void>, std::less<>> id_to_post

std::mutex mutex

template<class T>
class post_t : public T
    Inheritance diagram for safe::post_t:
    Collaboration diagram for safe::post_t:
```



Public Functions

```
template<class ...Args>
inline post_t(mail_t mail, Args&&... args)

inline ~post_t()
```

Public Members

```
mail_t mail
```

```
template<class T>
class queue_t
```

Public Types

```
using status_t = util::optional_t<T>
```

Public Functions

```
inline bool peek()

inline status_t pop()

template<class Rep, class Period>
inline status_t pop(std::chrono::duration<Rep, Period> delay)

inline queue_t(std::uint32_t max_elements = 32)

template<class ...Args>
inline void raise(Args&&... args)

inline bool running() const

inline void stop()

inline std::vector<T> &unsafe()
```

Private Members

```
bool _continue = {true}

std::condition_variable _cv

std::mutex _lock

std::uint32_t _max_elements
```

```

std::vector<T> _queue

template<class T>

class shared_t

```

Public Types

```

using construct_f = std::function<int(element_type&)>

using destruct_f = std::function<void(element_type&)>

using element_type = T

```

Public Functions

```

inline ptr_t ref()

template<class FC, class FD>
inline shared_t(FC &&fc, FD &&fd)

```

Private Members

```

construct_f _construct

std::uint32_t _count

destruct_f _destruct

std::mutex _lock

std::array<std::uint8_t, sizeof(element_type)> _object_buf

struct ptr_t

```

Collaboration diagram for `safe::shared_t::ptr_t`:

Public Functions

```

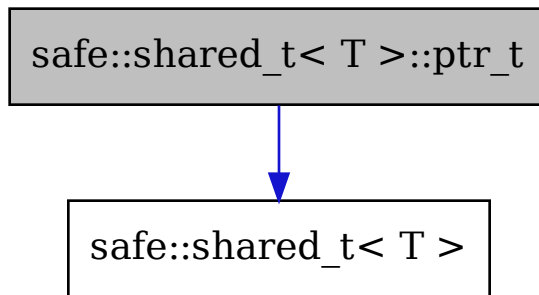
inline element_type *get() const

inline operator bool() const

inline element_type *operator->()

inline ptr_t &operator=(const ptr_t &ptr) noexcept

```



```

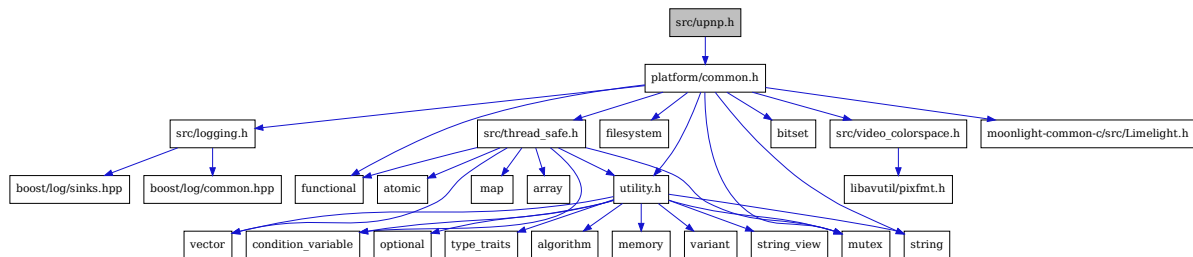
inline ptr_t &operator=(ptr_t &&ptr) noexcept
inline ptr_t()
inline ptr_t(const ptr_t &ptr) noexcept
inline ptr_t(ptr_t &&ptr) noexcept
inline explicit ptr_t(shared_t *owner)
inline void release()
inline ~ptr_t()
  
```

Public Members

shared_t ***owner**

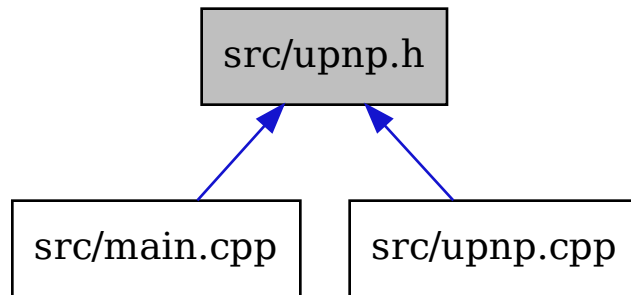
20.2.25 upnp

Include dependency graph for upnp.h:



This graph shows which files directly or indirectly include upnp.h:

todo



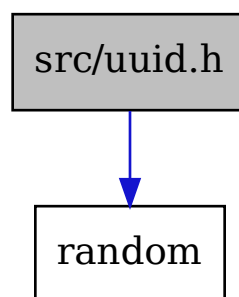
namespace **upnp**

20.2.26 utility

Todo: Add utility.h

20.2.27 uuid

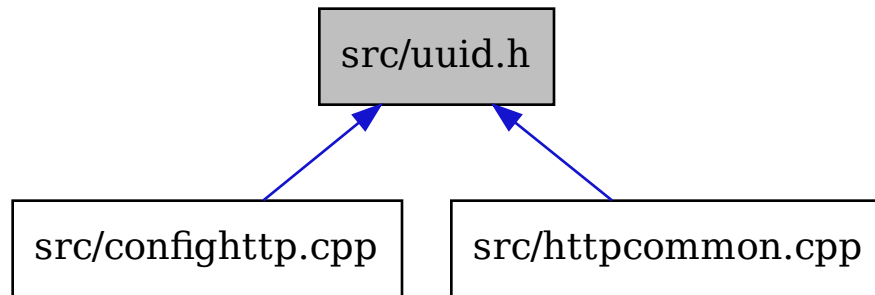
Include dependency graph for uuid.h:



This graph shows which files directly or indirectly include uuid.h:

todo

namespace **uuid_util**



union **uuid_t**

Public Functions

inline constexpr bool **operator**<(const *uuid_t* &other) const

inline constexpr bool **operator**==(const *uuid_t* &other) const

inline constexpr bool **operator**>(const *uuid_t* &other) const

inline std::string **string**() const

Public Members

std::uint16_t **b16**[8]

std::uint32_t **b32**[4]

std::uint64_t **b64**[2]

std::uint8_t **b8**[16]

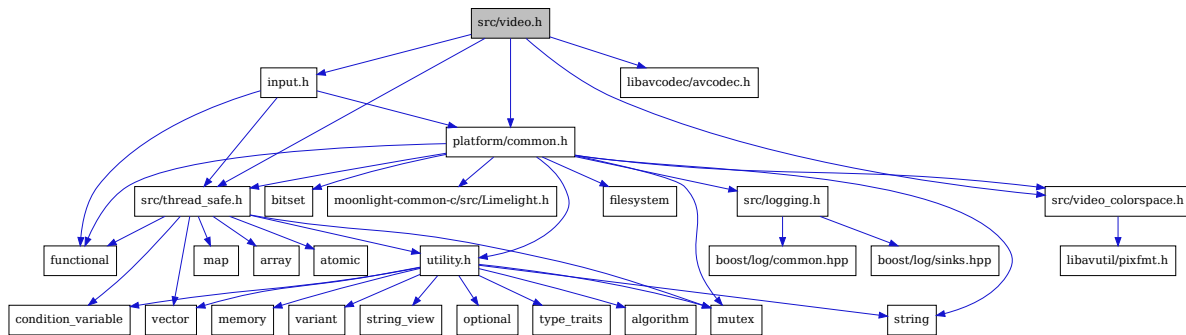
Public Static Functions

static inline *uuid_t* **generate**()

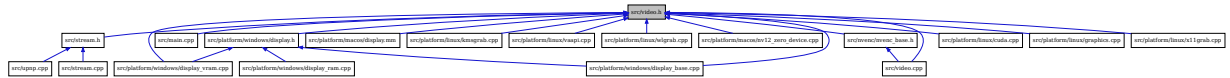
static inline *uuid_t* **generate**(std::default_random_engine &engine)

20.2.28 video

Include dependency graph for video.h:



This graph shows which files directly or indirectly include video.h:



todo

namespace **video**

Typedefs

using **hdr_info_t** = std::unique_ptr<hdr_info_raw_t>

using **packet_t** = std::unique_ptr<packet_raw_t>

struct **config_t**

Public Members

int **bitrate**

int **dynamicRange**

int **encoderCscMode**

int **framerate**

int **height**

```
int numRefFrames
```

```
int slicesPerFrame
```

```
int videoFormat
```

```
int width
```

```
struct hdr_info_raw_t
```

Public Functions

```
inline explicit hdr_info_raw_t(bool enabled)
```

```
inline explicit hdr_info_raw_t(bool enabled, const SS_HDR_METADATA &metadata)
```

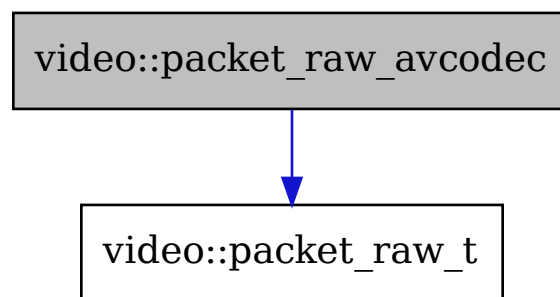
Public Members

```
bool enabled
```

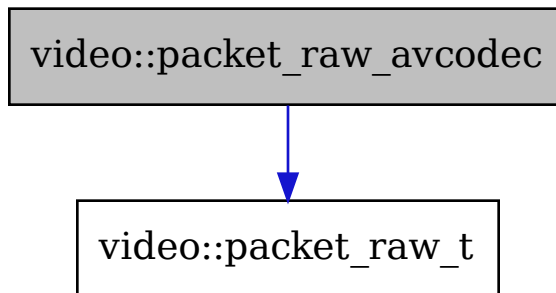
```
SS_HDR_METADATA metadata
```

```
struct packet_raw_avcodec : public video::packet_raw_t
```

Inheritance diagram for `video::packet_raw_avcodec`:



Collaboration diagram for `video::packet_raw_avcodec`:



Public Functions

```

inline virtual uint8_t *data() override
inline virtual size_t data_size() override
inline virtual int64_t frame_index() override
inline virtual bool is_idr() override
inline packet_raw_avcodec()
inline ~packet_raw_avcodec()

```

Public Members

```

AVPacket *av_packet

```

```

struct packet_raw_generic : public video::packet_raw_t

```

Inheritance diagram for `video::packet_raw_generic`:

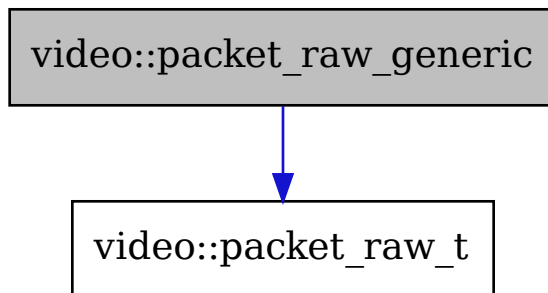
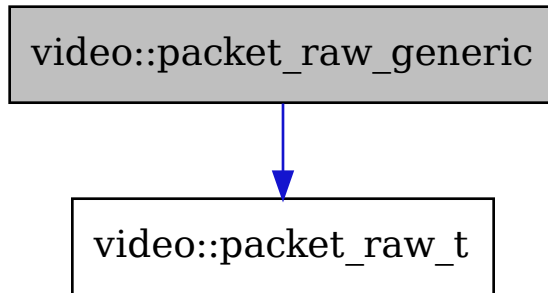
Collaboration diagram for `video::packet_raw_generic`:

Public Functions

```

inline virtual uint8_t *data() override
inline virtual size_t data_size() override
inline virtual int64_t frame_index() override
inline virtual bool is_idr() override
inline packet_raw_generic(std::vector<uint8_t> &&frame_data, int64_t frame_index, bool idr)

```



Public Members

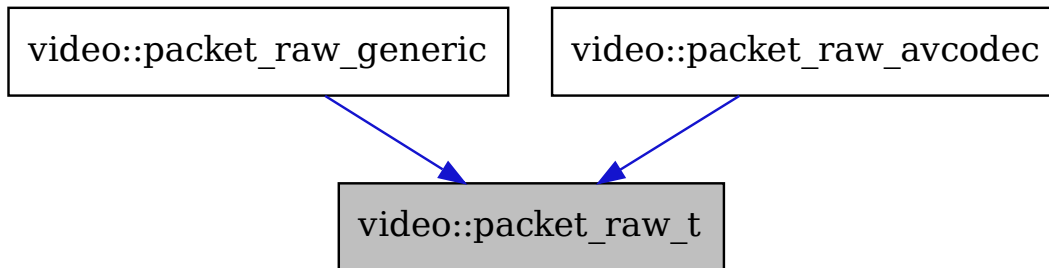
std::vector<uint8_t> **frame_data**

bool **idr**

int64_t **index**

struct **packet_raw_t**

Inheritance diagram for video::packet_raw_t:



Subclassed by *video::packet_raw_avcodec*, *video::packet_raw_generic*

Public Functions

virtual uint8_t ***data**() = 0

virtual size_t **data_size**() = 0

virtual int64_t **frame_index**() = 0

virtual bool **is_idr**() = 0

virtual ~**packet_raw_t**() = default

Public Members

bool **after_ref_frame_invalidation** = false

void ***channel_data** = nullptr

std::optional<std::chrono::steady_clock::time_point> **frame_timestamp**

```
std::vector<replace_t> *replacements = nullptr
```

```
struct replace_t
```

Public Functions

```
replace_t &operator=(replace_t&&) noexcept = default
```

```
replace_t(replace_t&&) noexcept = default
```

```
inline replace_t(std::string_view old, std::string_view _new) noexcept
```

Public Members

```
std::string_view _new
```

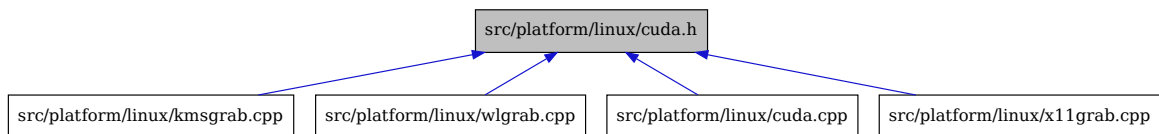
```
std::string_view old
```

20.2.29 common

Todo: Add common.h

20.2.30 cuda

This graph shows which files directly or indirectly include cuda.h:



todo

20.2.31 graphics

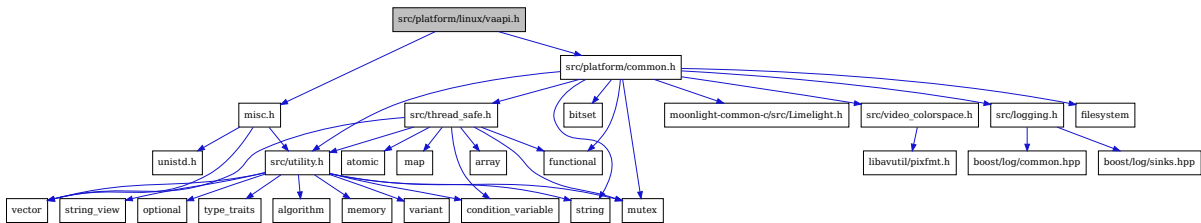
Todo: Add graphics.h

20.2.32 misc

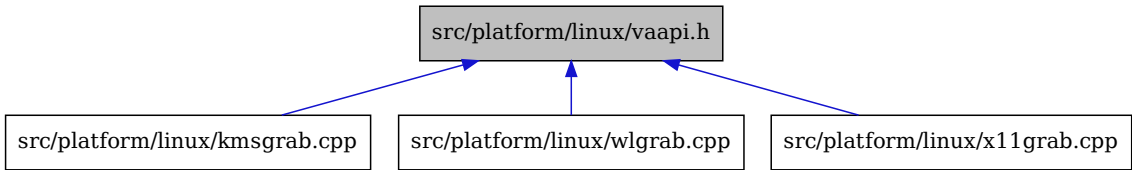
Todo: Add misc.h

20.2.33 vaapi

Include dependency graph for vaapi.h:



This graph shows which files directly or indirectly include vaapi.h:



todo

namespace **egl**

namespace **va**

20.2.34 wayland

Include dependency graph for wayland.h:

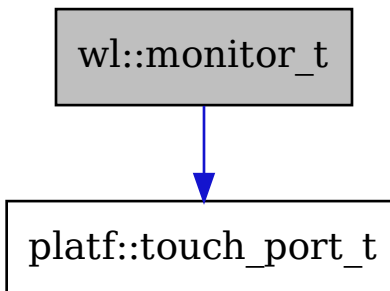
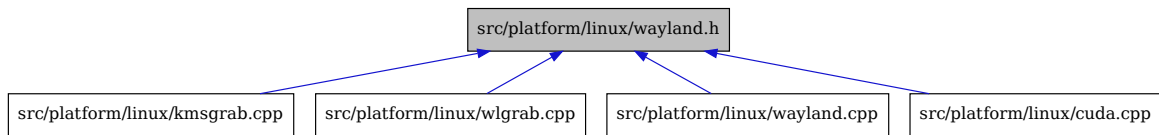
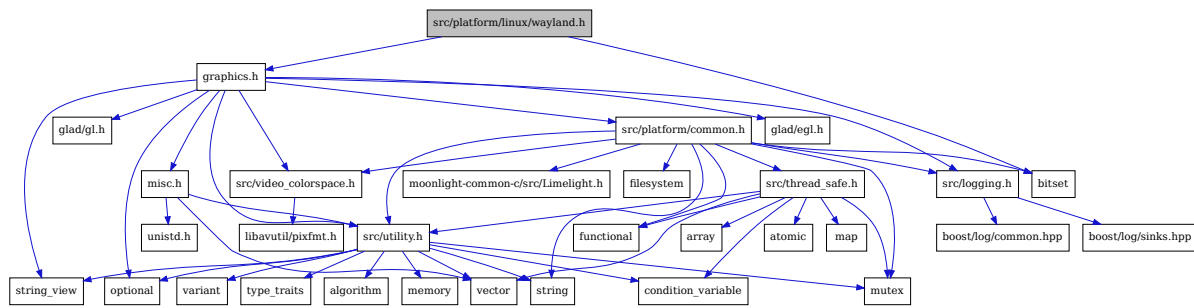
This graph shows which files directly or indirectly include wayland.h:

todo

namespace **wl**

class **monitor_t**

Collaboration diagram for `wl::monitor_t`:



Public Functions

void **listen**(zxdg_output_manager_v1 *output_manager)

monitor_t(const *monitor_t*&) = delete

monitor_t(*monitor_t*&&) = delete

inline **monitor_t**(wl_output *output)

monitor_t &**operator**=(const *monitor_t*&) = delete

monitor_t &**operator**=(*monitor_t*&&) = delete

Public Members

std::string **description**

std::string **name**

wl_output ***output**

platf::touch_port_t **viewport**

20.2.35 x11grab

Todo: Add x11grab.h

20.2.36 av_audio

Todo: Add av_audio.h

20.2.37 av_img_t

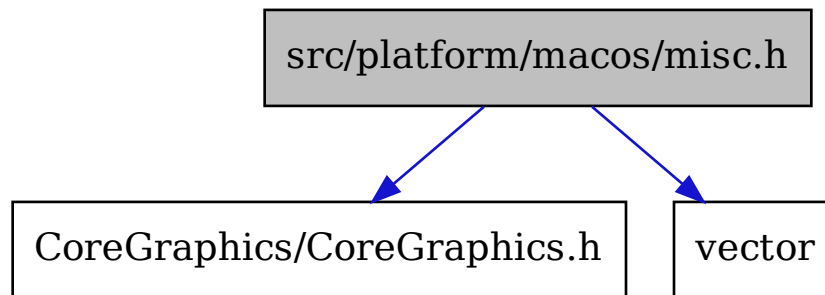
Todo: Add av_img_t.h

20.2.38 av_video

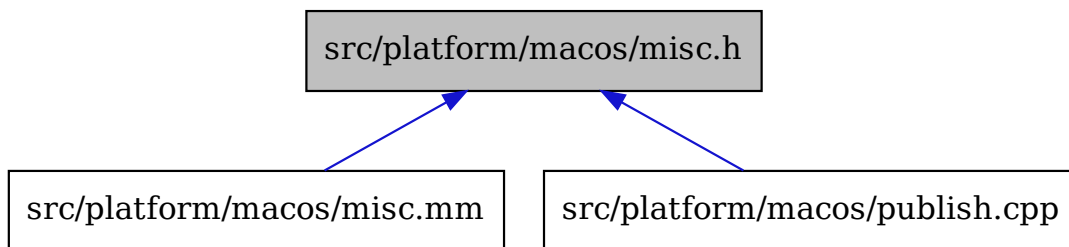
Todo: Add av_video.h

20.2.39 misc

Include dependency graph for misc.h:



This graph shows which files directly or indirectly include misc.h:



todo

namespace **dyn**

20.2.40 nv12_zero_device

Todo: Add nv12_zero_device.h

20.2.41 PolicyConfig

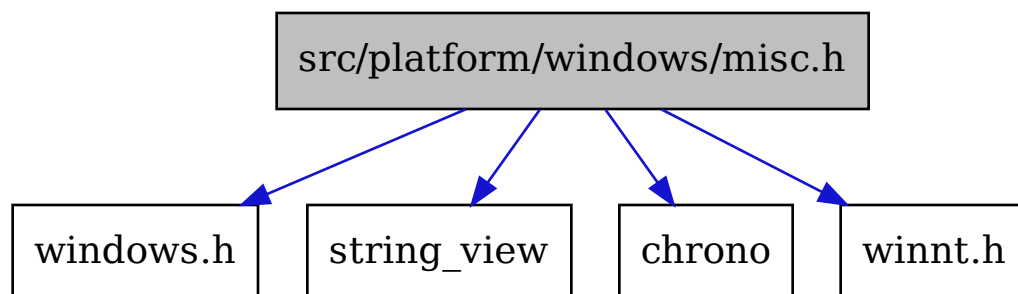
Todo: Add PolicyConfig.h

20.2.42 display

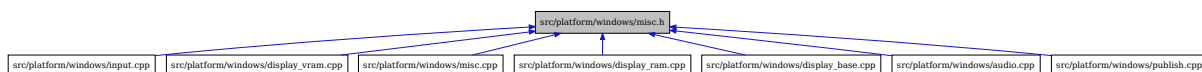
Todo: Add display.h

20.2.43 misc

Include dependency graph for misc.h:



This graph shows which files directly or indirectly include misc.h:



todo

namespace **platf**

CHANGELOG

21.1 0.22.1 - 2024-03-13

Breaking

- (ArchLinux) Drop support for standalone PKGBUILD files. Use the binary Arch package or install via AUR instead.
- (macOS) Drop support for experimental dmg package. Use Homebrew or MacPorts instead.

Added

- (macOS) Added Homebrew support

Changed

- (Process/Windows) The working directory is now searched first when the command contains a relative path
- (ArchLinux) The kmsgrab capture backend is now compiled by default to support Wayland capture on non-wlroots-based compositors
- (Capture/Linux) X11 capture is now preferred over kmsgrab for cards that lack atomic modesetting support to ensure cursor capture works
- (Capture/Linux) Kmsgrab will only choose NVENC by default if the display is connected to the Nvidia GPU to avoid possible EGL import failures

Fixed

- (Config) Fix unsupported resolution error with some Moonlight clients
- (Capture/Windows) Fix crash when streaming Ryujinx, Red Alert 2, and other apps that use unusually sized monochrome cursors
- (Capture/Linux) Fix crash in KMS cursor capture when running on Arch-based distros
- (Capture/Linux) Fix crash if CUDA GPU has a PCI ID with hexadecimal digits greater than 9
- (Process/Windows) Fix starting apps when the working directory is enclosed in quotes
- (Process/Windows) Fix process tree tracking when the app is launched via a cmd.exe trampoline
- (Installer/Windows) Fix slow operation during ViGEmBus installation that may cause the installer to appear stuck
- (Build/macOS) Fix issues building on macOS 13 and 14
- (Build/Linux) Fix missing install script in the Arch binary package
- (Build/Linux) Fix missing optional dependencies in the Arch binary package
- (Build/Linux) Ensure correct Arch pkg is published to GitHub releases

- (Capture/Linux) Fix mismatched case and unhandled exception in CUDA device lookup
- (Config) Add missing resolution to default config ui
- (Linux) Fix udev rules for uinput access not working until after reboot
- (Linux) Fix wrong path in desktop files
- (Tray) Cache icons to avoid possible DRM issues
- (Tray) Fix attempt to update tray icon after it was destroyed
- (Linux) Migrate old config files to new location if env `SUNSHINE_MIGRATE_CONFIG=1` is set (automatically set for Flatpak)
- (Linux/Fedora) Re-enable CUDA support and bump to 12.4.0

Misc

- (Build/Windows) Adjust Windows debuginfo artifact to reduce confusion with real release binaries

21.2 0.22.0 - 2024-03-03

Breaking

- (Network) Clients must now be paired with the host before they can use Wake-on-LAN
- (Build/Linux) Drop Fedora 37 support

Added

- (Input/Linux) Add native/pen touch support for Linux
- (Capture/Linux) Add HDR streaming support for Linux using KMS capture backend
- (Capture/Linux) Add KMS capture support for Nvidia GPUs running Wayland
- (Network) Add support for full E2E stream encryption, configurable for LAN and WAN independently
- (Process) Add process group tracking to automatically handle launchers that spawn other child processes
- (Capture/Windows) Add setting for controlling GPU power saving and encoding latency tradeoff for NVENC
- (Capture/Windows) Add additional encoding settings for NVENC
- (Process/Windows) Add experimental support for launching URLs and other non-exe files
- (Capture/Windows) Add setting to allow use of slower HEVC encoding on older Intel GPUs
- (Input/Windows) Add settings to control automatic gamepad type selection heuristics
- (Input/Windows) Add setting to allow DS4 back/select button to trigger touchpad click
- (Input) Add setting to disable high resolution scrolling and native pen/touch support
- (Network) Add support for certificate types other than RSA-2048
- (Build/Linux) Add Fedora 39 docker image and rpm package
- (Capture/Linux) Display monitor indexes in logs for wlroots and KMS capture backends
- (UI) Add link to logs inside fatal error container
- (UI) Add hash handler and ids for all configuration categories and settings

Changed

- (UI) Several configuration options have been moved to more suitable locations

- (Network) Client-selected bitrate is now adjusted for FEC percentage and other stream overhead
- (Capture/Linux) Improve VA-API encoding performance on Intel GPUs
- (Capture) Connection establishment delay is reduced by eliminating many encoder probing operations
- (Process) Graceful termination of running processes is attempted first when stopping apps
- (Capture) Improve software encoding performance by enabling multi-threaded color conversion
- (Capture) Adjust default CPU thread count for software encoding from 1 to 2 for improved performance
- (Steam/Windows) Modernized the default Steam app shortcut to avoid depending on Steam's install location and support app termination
- (Linux) Updated desktop files
- (Config) Add 2560x1440 to default resolutions
- (Network) Use the configured ping timeout for the initial launch event timeout
- (UI) Migrate UI to Vite and Vue3, and various UX improvements
- (Logging) Adjust wording and severity of some log messages
- (Build) Use a single submodule for ffmpeg
- (Install/Windows) Skip ViGEmBus installation if a supported version is already installed
- (Build/Linux) Optionally, allow using the system installation of wayland-protocols
- (Build/Linux) Make vaapi optional
- (Windows) Replace boost::json with nlohmann/json

Fixed

- (Network/Windows) Fix auto-discovery of hosts by iOS/tvOS clients
- (Network) Fix immediate connection termination when streaming over some Internet connections
- (Capture/Linux) Fix missing mouse cursor when using KMS capture on a GPU with hardware cursor support
- (Capture/Windows) Add workaround for Nvidia driver bug causing Sunshine to crash when RTX HDR is globally enabled
- (Capture/Windows) Add workaround for AMD driver bug on pre-RDNA GPUs causing hardware encoding failure
- (Capture/Windows) Reintroduce support for NVENC on older Nvidia GPU drivers (v456.71-v522.25)
- (Capture/Windows) Fix encoding on old Intel GPUs that don't support low-power H.264 encoding
- (Capture/Linux) Fix GL errors or corrupt video output on GPUs that use aux planes such as Intel Arc
- (Capture/Linux) Fix GL errors or corrupt video output on GPUs that use DRM modifiers on YUV buffers
- (Input/Windows) Fix non-functional duplicate controllers appearing in rare cases
- (Input/Windows) Avoid triggering crash in ViGEmBus when the system goes to sleep
- (Input/Linux) Fix scrolling in applications that don't support high-resolution scrolling
- (Input/Linux) Fix absolute mouse input being interpreted as touch input
- (Capture/Linux) Fix wlroots capture causing GL errors and crashes
- (Capture/Linux) Fix wlroots capture failing when the display scale factor was not 1
- (Capture/Linux) Fix excessive CPU usage when using wlroots capture backend

- (Capture/Linux) Fix capture of virtual displays created by the amdgpu kernel driver
- (Audio/Windows) Fix audio capture failures on Insider Preview versions of Windows 11
- (Capture/Windows) Fix incorrect portrait mode rotation
- (Capture/Windows) Fix capture recovery when a driver update/crash occurs while streaming
- (Capture/Windows) Fix delay displaying UAC dialogs when the mouse cursor is not moving
- (Capture/Linux) Fix corrupt video output or stream disconnections if the display resolution changes while streaming
- (Capture/Linux) Fix color of aspect ratio padding in the capture image with VA-API
- (Capture/Linux) Fix NVENC initialization error when using X11 capture with some GPUs
- (Tray/Linux) Fix random crash when the tray icon is updating
- (Network) Fix QoS tagging when running in IPv4+IPv6 mode
- (Process) Fix termination of child processes upon app quit when the parent has already terminated
- (Process) Fix notification of graceful termination to connected clients when Sunshine quits
- (Capture) Fix corrupt output or green aspect-ratio padding when using software encoding with some video resolutions
- (Windows) Fix crashes when processing file paths or other strings with certain non-ASCII characters
- (Capture) Ensure user supplied framerates are used exclusively in place of pre-defined framerates
- (CMake/Linux) Skip including unnecessary headers
- (Capture/Linux) Replace vaTerminate method with dl handle
- (Capture/Linux) Fix capture when DRM is enabled and x11 is disabled
- (Tray) Use PROJECT_NAME definition for tooltip
- (CMake) Use GNUInstallDirs to install data and lib directories
- (macOS) Replace deprecated code
- (API) Allow trailing slashes in on API endpoints
- (API) Add additional pin validation
- (Linux) Use XDG spec for fetching config directory
- (CMake) Properly find evdev
- (Config) Properly save global_prep_cmd and fps settings

Dependencies

- Bump third-party/wayland-protocols from 681c33c to 46f201b
- Bump third-party/nv-codec-headers from 9402b5a to 22441b5
- Bump third-party/nanors from 395e5ad to e9e242e
- Bump third-party/Simple-Web-Server from 2f29926 to 27b41f5
- Bump ffmpeg
- Bump third-party/tray from 2664388 to 2bf1c61
- Bump actions/setup-python from 4 to 5
- Bump actions/upload-artifact from 3 to 4

- Bump @fortawesome/fontawesome-free from 6.4.2 to 6.5.1
- Bump babel from 2.13.0 to 2.14.0
- Move miniupnpc from submodule to system installed package
- Bump furo from 2023.9.10 to 2024.1.29
- Bump third-party/moonlight-common-c from f78f213 to cbd0ec1
- Bump third-party/ViGEmClient from 1920260 to 8d71f67
- Bump peter-evans/create-pull-request from 5 to 6
- Bump bootstrap from 5.3.2 to 5.3.3

Misc

- (Build) Update global workflows
- (Docs/Linux) Add example for setting custom resolution with NVIDIA
- (Docs) Fix broken links
- (Docs/Windows) Add information about disk permissions
- (Docs) Fix failing images
- (Docs) Use glob pattern to match source code docs
- (CI/macOS) Install boost from source
- (Docs) Add reset credentials examples for unique packages
- (Docs) Refactor and general cleanup
- (Docs) Cross-reference config settings to the UI
- (Docs/Docker) Add podman notes
- (Build) Use CMAKE_SOURCE_DIR property everywhere
- (Build/Docker) Add docker toolchain file for CLion
- (macOS) Various code style fixes
- (Deps) Alphabetize git submodules
- (Docs/Examples) Update URI examples
- (Refactor) Refactored some code in preparation for unit testing implementation
- (CMake) Add option to skip cuda inheriting compile options
- (CMake) Add option to error build on warnings

21.3 0.21.0 - 2023-10-15

Added

- (Input) Add support for automatically selecting the emulated controller type based on the physical controller connected to the client
- (Input/Windows) Add support for Applications (context menu) key
- (Input/Windows) Implement touchpad, motion sensors, battery state, and LED control for the emulated Dual-Shock 4 controller

- (Input) Advertise support for new input features to clients
- (Linux/Debian) Added Debian Bookworm package
- (Prep-Commands) Expose connection environment variables
- (Input/Windows) Implement pen and touch support
- (Capture/Windows) Add standalone NVENC encoder
- (Capture) Implement AV1 encoding
- (Network) Implement IPv6 support
- (Capture/Windows) Add option to disable realtime hags
- (Graphics/NVIDIA) Add an option to decrease GPU scheduling priority to workaround HAGS video hang
- (Capture/Linux) Add FFmpeg powerpc64le architecture for self compiling Sunshine
- (Capture/Windows) Add support for capturing rotated displays
- (System Tray) Implement streaming event notifications
- (UI) Add port configuration table
- (Applications) Added option to automatically treat launcher type apps as detached commands
- (Input/Gamepad) Allow the Misc button to work as Guide on emulated Xbox 360 controllers

Changed

- (Input) Reduce latency by implementing input batching
- (Logging) Move input packet debug prints off the control stream thread
- (Input) Refactor gamepad emulation code to use DS4 extended report format
- (Graphics/NVIDIA) Modify and restore NVIDIA control panel settings before and after stream, respectively
- (Graphics/NVIDIA) New config page for NVENC
- (Graphics/Windows) Refactor DX shaders
- (Input/Windows) Use our own keycode mapping to avoid installing the US English keyboard layout

Fixed

- (UI) Fix update notifications
- (Dependencies/Linux) Replace libboost chrono and thread with standard chrono and thread
- (Input) Increase maximum gamepad limit to 16
- (Network) Allow use of multiple ENet channels
- (Network) Consider link-local addresses on LAN
- (Input) Fixed issue where button may sometimes stick on Windows
- (Input) Fix “ControllerNumber not allocated” warning when a gamepad is removed
- (Input) Fix handling of gamepad feedback with multiple clients connected
- (Input) Fix clamping mouse position to aspect ratio adjusted viewport
- (Graphics/AMD) Fix crash during startup on some older AMD GPUs
- (Logging) Fix crash when non-ASCII characters are logged
- (Prep-Commands) Fix resource exhaustion bug which could occur when many prep commands were used

- (Subprocesses) Fix race condition when inserting new processes
- (Logging) Log error if encoder doesn't produce IDR frame on demand
- (Audio) Improve audio capture logic and logging
- (Logging) Fix AMF logging to match configured log level
- (Logging) Log FFmpeg to log file instead of stdout
- (Capture) Reject codecs that are not supported by display device
- (Capture) Add fallbacks for unsupported codec settings
- (Capture) Avoid probing HEVC or AV1 codecs in some cases
- (Capture) Remove DwmFlush()
- (Capture/Windows) Improvements to capture sleeps for better frame stability
- (Capture/Windows) Adjust capture rate to better match with display
- (Linux/ArchLinux) Fix package version in PKGBUILD and precompiled package
- (UI) Highlight fatal log messages in web ui
- (Commands) Allow stream if prep command fails
- (Capture/Linux) Fix KMS grab VRAM capture with libva 2.20
- (Capture/macOS) Fix video capture backend
- (Misc/Windows) Don't start the session monitor window when launched in command mode
- (Linux/AppImage) Use the linuxdeploy GTK plugin to correctly deploy GTK3 dependencies
- (Input/Windows) Fix reWASD not recognizing emulated DualShock 4 input

Dependencies

- Bump bootstrap from 5.2.3 to 5.3.2
- Bump third-party/moonlight-common-c from c9426a6 to 7a6d12f
- Bump gcc-10 in Ubuntu 20.04 docker image
- Bump furo from 2023.5.20 to 2023.9.10
- Bump sphinx from 7.0.1 to 7.2.6
- Bump cmake from 3.26 to 3.27 in Fedora docker images
- Move third-party/nv-codec-headers from sdk/11.1 branch to sdk/12.0 branch
- Automatic bump ffmpeg
- Bump actions/checkout from 3 to 4
- Bump boost from 1.80 to 1.81 in Macport manifest
- Bump @fontawesome/fontawesome-free from 6.4.0 to 6.4.2

Misc

- (Docs) Force badges to use svg
- (Docs) Add Linux SSH example
- (Docs) Add information about mesa for Linux
- (CI) Free additional space on Docker, Flatpak, and AppImage builds due to internal changes on GitHub runners

- (Docs/Logging/UI) Corrected various typos
- (Docs) Add blurb about Gamescope compatibility
- (Installer/Windows) Use system proxy to download ViGEmBus
- (CI) Ignore third-party directory for clang-format
- (Docs/Linux) Add Plasma-Compatible resolution example
- (Docs) Add Sunshine website available at <https://app.lizardbyte.dev/Sunshine>
- (Build/Windows) Fix audio code build with new MinGW headers
- (Build/Windows) Fix QoS code build with new MinGW headers
- (CI/Windows) Prevent winget action from creating an update when running in a fork
- (CI/Windows) Change winget job to ubuntu-latest runner
- (CI) Add CodeQL analysis
- (CI/Docker) Fix ArchLinux image caching issue
- (Windows) Manifest improvements
- (CI/macOS) Simplify macport build
- (Docs) Remove deprecated options from readthedocs config
- (CI/Docs) Lint rst files
- (Docs) Update localization information (after consolidating Crowdin projects)
- (Cmake) Split CMakelists into modules
- (Docs) Add Linux Headless/SSH Guide

21.4 0.20.0 - 2023-05-28

Breaking

- (Windows) The Windows installer version of Sunshine is now always launched by the Sunshine Service. Manually launching Sunshine.exe from Program Files is no longer supported. This was necessary to address security issues caused by non-admin users having access to Sunshine's config data. If you have set up Task Scheduler or other mechanisms to launch Sunshine automatically, remove those from your system before updating.
- (Windows) The Start Menu shortcut has been redesigned for use with the Sunshine Service. It now launches Sunshine in the background (if not already running) and opens the Web UI, avoiding the persistent Command Prompt window present in prior versions. The Start Menu shortcut is now the recommended method for opening the Web UI and launching Sunshine.
- (Network/UPnP) If the Moonlight Internet Hosting Tool is installed alongside Sunshine, you must remove it or upgrade to v5.6 or later to prevent conflicts with Sunshine's UPnP support. As a reminder, the Moonlight Internet Hosting Tool is not required to stream over the Internet with Sunshine. Instead, simply enable UPnP in the Sunshine Web UI.
- (Windows) If Steam is installed, the Steam Streaming Speakers driver will be automatically installed when starting a stream for the first time. This behavior can be disabled in the Audio/Video tab of the Web UI. This Steam driver enables support for surround sound and muting host audio without requiring any manual configuration.
- (Input) The Back Button Timeout option has been renamed to Guide Button Emulation Timeout and has been disabled by default to ensure long presses on the Back button work by default. The previous behavior can be restored by setting the Guide Button Emulation Timeout to 2000.

- (Windows) The service name of SunshineSvc has been changed to SunshineService to address a false positive in MalwareBytes. If you have any scripts or other logic on your system that is using the service name, you will need to update that for the new name.
- (Windows) To support new installer features, install-service.bat no longer sets the service to auto-start by default. Users executing install-service.bat manually on the Sunshine portable build must also execute autostart-service.bat to start Sunshine on boot. However, installing the service manually like this is not recommended. Instead, use the Sunshine installer which handles service installation and configuration automatically.
- (Linux/Fedora) Fedora 36 builds are removed due to upstream end of support

Added

- (Windows) Added an option to launch apps and prep/undo commands as administrator
- (Installer/Windows) Added an option to choose whether Sunshine launches on boot. If not configured to launch on boot, use the Start Menu shortcut to start Sunshine when desired.
- (Input/Windows) Added option to send VK codes instead of scancodes for compatibility with iOS/Android devices using non-English keyboard layouts
- (UI) The Apply/Restart option is now available in the Web UI for all platforms
- (Systray) Added a Restart option to the system tray context menu
- (Video/Windows) Added host latency data to video frames. This requires future updates to Moonlight to display in the on-screen overlay.
- (Audio) Added support for matching Audio Sink and Virtual Sink values on device names
- (Client) Added friendly error messages for clients when streaming fails to start
- (Video/Windows) Added warning log messages when Windows is hiding DRM-protected content from display capture
- (Interop/Windows) Added warning log messages when GeForce Experience is currently using the same ports as Sunshine
- (Linux/Fedora) Fedora 38 builds are now available

Changed

- (Video) Encoder selection now happens at each stream start for more reliable GPU detection
- (Video/Windows) The host display now stays on while clients are actively streaming
- (Audio) Streaming will no longer fail if audio capture is unavailable
- (Audio/Windows) Sunshine will automatically switch back to the Virtual Sink if the default audio device is changed while streaming
- (Audio) Changes to the host audio playback option will now take effect when resuming a session from Moonlight
- (Audio/Windows) Sunshine will switch to a different default audio device if Steam Streaming Speakers are the default when Sunshine starts. This handles cases where Sunshine terminates unexpectedly without restoring the default audio device.
- (Apps) The Connection Terminated dialog will no longer appear in Moonlight when the app on the host exits normally
- (Systray/Windows) Quitting Sunshine via the system tray will now stop the Sunshine Service rather than leaving it running and allowing it to restart Sunshine
- (UI) The 100.64.0.0/10 CGN IP address range is now treated as a LAN address range
- (Video) Removed a workaround for some versions of Moonlight prior to mid-2022

- (UI) The PIN field is now cleared after successfully pairing
- (UI) User names are now treated as case-insensitive
- (Linux) Changed udev rule to automatically grant access to virtual input devices
- (UI) Several item descriptions were adjusted to reflect newer configuration recommendations
- (UI) Placeholder text opacity has been reduced to improve contrast with non-placeholder text
- (Video/Windows) Minor capture performance improvements

Fixed

- (Video) VRAM usage while streaming is significantly reduced, particularly with higher display resolutions and HDR
- (Network/UPnP) UPnP support was rewritten to fix several major issues handling router restarts, IP address changes, and port forwarding expiration
- (Input) Fixed modifier keys from the software keyboard on Android clients
- (Audio) Fixed handling of default audio device changes while streaming
- (Windows) Fixed streaming after Microsoft Remote Desktop or Fast User Switching has been used
- (Input) Fixed some games not recognizing emulated Guide button presses
- (Video/Windows) Fixed incorrect gamma when using an Advanced Color SDR display on the host
- (Installer/Windows) The installer is no longer blurry on High DPI systems
- (Systray/Windows) Fixed multiple system tray icons appearing if Sunshine exits unexpectedly
- (Systray/Windows) Fixed the system tray icon not appearing in several situations
- (Windows) Fixed hang on shutdown/restart if mDNS registration fails
- (UI) Fixed missing response headers
- (Stability) Fixed several possible crashes in cases where the client did not successfully connect
- (Stability) Fixed several memory leaks
- (Input/Windows) Back/Select input now correctly generates the Share button when emulating DS4 controllers
- (Audio/Windows) Fixed various bugs in audio-info.exe that led to inaccurate output on some systems
- (Video/Windows) Fixed incorrect resolution values from dxgi-info.exe on High DPI systems
- (Video/Linux) Fixed poor quality encoding from H.264 on Intel encoders
- (Config) Fixed a couple of typos in predefined resolutions

Dependencies

- Bump sphinx-copybutton from 0.5.1 to 0.5.2
- Bump sphinx from 6.13 to 7.0.1
- Bump third-party/nv-codec-headers from 2055784 to 2cd175b
- Bump furo from 2023.3.27 to 2023.5.20

Misc

- (Build/Linux) Add X11 to PLATFORM_LIBRARIES when found
- (Build/macOS) Fix compilation with Clang 14
- (Docs) Fix nvlax URL

- (Docs) Add diagrams using graphviz
- (Docs) Improvements to source code documentation
- (Build) Unpin docker dependencies
- (Build/Linux) Honor install prefix for tray icon
- (Build/Windows) Unstripped binaries are now provided as a debuginfo package to support crash dump debugging
- (Config) Config directories are now created recursively

21.5 0.19.1 - 2023-03-30

Fixed

- (Audio) Fixed no audio issue introduced in v0.19.0

21.6 0.19.0 - 2023-03-29

Breaking

- (Linux/Flatpak) Moved Flatpak to org.freedesktop.Platform 22.08 and Cuda 12.0.0 This will drop support for Nvidia GPUs with compute capability 3.5

Added

- (Input) Added option to suppress input from gamepads, keyboards, or mice
- (Input/Linux) Added unicode support for remote pasting (may not work on all DEs)
- (Input/Linux) Added XTest input fallback
- (UI) Added version notifications to web UI
- (Linux/Windows) Add system tray icon
- (Windows) Added ability to safely elevate commands that fail due to insufficient permissions when running as a service
- (Config) Added global prep commands, and ability to exclude an app from using global prep commands
- (Installer/Windows) Automatically install ViGEmBus if selected

Changed

- (Logging) Changed client verified messages to debug to prevent spamming the log
- (Config) Only save non default config values
- (Service/Linux) Use xdg-desktop-autostart for systemd service
- (Linux) Added config option to force capture method
- (Windows) Execute prep command in context of current user
- (Linux) Allow disconnected X11 outputs

Fixed

- (Input/Windows) Fix issue where internation keys were not translated correct, and modifier keys appeared stuck
- (Linux) Fixed startup when /dev/dri didn't exist

- (UI) Changes software encoding settings to select menu instead of text input
- (Initialization) Do not terminate upon failure, allowing access to the web UI

Dependencies

- Bump third-party/moonlight-common-c from 07beb0f to c9426a6
- Bump babel from 2.11.0 to 2.12.1
- Bump @fontawesome/fontawesome-free from 6.2.1 to 6.4.0
- Bump third-party/ViGEmClient from 9e842ba to 726404e
- Bump ffmpeg
- Bump third-party/miniupnp from 014c9df to e439318
- Bump furo from 2022.12.7 to 2023.3.27
- Bump third-party/nanors from 395e5ad to e9e242e

Misc

- (GitHub) Shared feature request board with Moonlight
- (Docs) Improved application examples
- (Docs) Added WIP documentation for source code using Doxygen and Breathe
- (Build) Fix linux clang build errors
- (Build/Archlinux) Skip irrelevant submodules
- (Build/Archlinux) Disable download timeout
- (Build/macOS) Support compiling for earlier releases of macOS
- (Docs) Add favicon
- (Docs) Add missing config default values
- (Build) Fix compiler warnings due to depreciated elements in C++17
- (Build) Fix libcurl link errors
- (Clang) Adjusted formatting rules

21.7 0.18.4 - 2023-02-20

Fixed

- (Linux/AUR) Drop support of AUR package
- (Docker) General enhancements to docker images

21.8 0.18.3 - 2023-02-13

Added

- (Linux) Added PKGBUILD for Archlinux based distros to releases
- (Linux) Added precompiled package for Archlinux based distros to releases
- (Docker) Added archlinux docker image (x86_64 only)

21.9 0.18.2 - 2023-02-13

Fixed

- (Video/KMV/Linux) Fixed wayland capture on Nvidia for KMS
- (Video/Linux) Implement vaSyncBuffer stuff for libva <2.9.0
- (UI) Fix issue where mime type was not being set for node_modules when using a reverse proxy
- (UI/macOS) Added missing audio sink config options
- (Linux) Specify correct Boost dependency versions
- (Video/AMF) Add missing encoder tunables

21.10 0.18.1 - 2023-01-31

Fixed

- (Linux) Fixed missing dependencies for deb and rpm packages
- (Linux) Use dynamic boost

21.11 0.18.0 - 2023-01-29

Attention, this release contains critical security fixes. Please update as soon as possible. Additionally, we are encouraging users to change your Sunshine password, especially if you expose the web UI (i.e. port 47790 by default) to the internet, or have ever uploaded your logs with verbose output to a public resource.

Added

- (Windows) Add support for Intel QuickSync
- (Linux) Added aarch64 deb and rpm packages
- (Windows) Add support for hybrid graphics systems, such as laptops with both integrated and discrete GPUs
- (Linux) Add support for streaming from Steam Deck Gaming Mode
- (Windows) Add HDR support, see <https://docs.lizardbyte.dev/projects/sunshine/en/latest/about/usage.html#hdr-support>

Fixed

- (Network) Refactor code for UPnP port forwarding
- (Video) Enforce 10 FPS encoding frame rate minimum to improve static image quality

- (Linux) deb and rpm packages are now specific to destination distro and version
- (Docs) Add nvidia/nvenc preset migration guide
- (Network) Performance optimizations
- (Video/Windows) Fix streaming to multiple clients from hardware encoder
- (Linux) Fix child process spawning
- (Security) Fix security vulnerability in implementation of SimpleWebServer
- (Misc) Rename “Steam BigPicture” to “Steam Big Picture” in default apps.json
- (Security) Scrub basic authorization header from logs
- (Linux) The systemd service will now restart in the event of a crash
- (Video/KMS/Linux) Fixed error: couldn't import RGB Image: 00003002 and 00003004
- (Video/Windows) Fix stream freezing triggered by the resolution changed
- (Installer/Windows) Fixes silent installation and other miscellaneous improvements
- (CPU) Significantly improved CPU usage

21.12 0.17.0 - 2023-01-08

If you are running Sunshine as a service on Windows, we are strongly urging you to update to v0.17.0 as soon as possible. Older Windows versions of Sunshine had a security flaw in which the binary was located in a user-writable location which is problematic when running as a service or on a multi-user system. Additionally, when running Sunshine as a service, games and applications were launched as SYSTEM. This could lead to issues with save files and other game settings. In v0.17.0, games now run under your user account without elevated privileges.

Breaking

- (Apps) Removed automatic desktop entry (Re-add by adding an empty application named “Desktop” with no commands, “desktop.png” can be added as the image.)
- (Windows) Improved user upgrade experience (Suggest to manually uninstall existing Sunshine version before this upgrade. Do NOT select to remove everything, if prompted. Make a backup of config files before uninstall.)
- (Windows) Move config files to specific directory (files will be migrated automatically if using Windows installer)
- (Dependencies) Fix npm path (breaking change for package maintainers)

Added

- (macOS) Added initial support for arm64 on macOS through Macports portfile
- (Input) Added support for foreign keyboard input
- (Misc) Logs inside the WebUI and log to file
- (UI/Windows) Added an Apply button to configuration page when running as a service
- (Input/Windows) Enable Mouse Keys while streaming for systems with no physical mouse

Fixed

- (Video) Improved capture performance
- (Audio) Improved audio bitrate and quality handling
- (Apps/Windows) Fixed PATH environment variable handling

- (Apps/Windows) Use the proper environment variable for the Program Files (x86) folder
- (Service/Windows) Fix SunshineSvc hanging if an error occurs during startup
- (Service/Windows) Spawn Sunshine.exe in a job object, so it is terminated if SunshineSvc.exe dies
- (Video) windows/vram: fix fringing in NV12 colour conversion
- (Apps/Windows) Launch games under the correct user account
- (Video) nvenc, amdvc: rework all user presets/options
- (Network) Generate certificates with unique serial numbers
- (Service/Windows) Graceful termination on shutdown, logoff, and service stop
- (Apps/Windows) Fix launching apps when Sunshine is running as admin
- (Misc) Remove/fix calls to std::abort()
- (Misc) Remove prompt to press enter after Sunshine exits
- (Misc) Make log priority consistent for execution messages
- (Apps) Applications in Moonlight clients are now updated automatically after editing
- (Video/Linux) Fix wayland capture on nvidia
- (Audio) Fix 7.1 surround channel mapping
- (Video) Fix NVENC profile values not applying
- (Network) Fix origin_web_ui_allowed binding
- (Service/Windows) Self terminate/restart service if process hangs for 10 seconds
- (Input/Windows) Fix Windows masked cursor blending with GPU encoders
- (Video) Color conversion fixes and BT.2020 support

Dependencies

- Bump ffmpeg from 4.4 to 5.1
- ffmpeg_patches: add amfenc delay/buffering fix
- CBS moved to ffmpeg submodules
- Migrate to upstream Simple-Web-Server submodule
- Bump third-party/TPCircularBuffer from bce9170 to 8833b3a
- Bump third-party/moonlight-common-c from 8169a31 to ef9ad52
- Bump third-party/miniupnp from 6f848ae to 207cf44
- Bump third-party/ViGEmClient from f719a1d to 9e842ba
- Bump bootstrap from 5.0.0 to 5.2.3
- Bump @fontawesome/fontawesome-free from 6.2.0 to 6.2.1

21.13 0.16.0 - 2022-12-13

Added

- Add cover finder
- (Docker) Add arm64 docker image
- (Flatpak) Add installation helper scripts
- (Windows) Add support for Unicode input messages

Fixed

- (Linux) Reintroduce Ubuntu 20.04 and 22.04 specific deb packages
- (Linux) Fixed udev and systemd file locations

Dependencies

- Bump babel from 2.10.3 to 2.11.0
- Bump sphinx-copybutton from 0.5.0 to 0.5.1
- Bump KSXGitHub/github-actions-deploy-aur from 2.5.0 to 2.6.0
- Use npm for web dependencies (breaking change for third-party package maintainers)
- Update moonlight-common-c
- Use pre-built ffmpeg from LizardByte/build-deps for all sunshine builds (breaking change for third-party package maintainers)
- Bump furo from 2022.9.29 to 2022.12.7

Misc

- Misc org level workflow updates
- Fix misc typos in docs
- Fix winget release

21.14 0.15.0 - 2022-10-30

Added

- (Windows) Add firewall rules scripts
- (Windows) Automatically add and remove firewall rules at install/uninstall
- (Windows) Automatically add and remove service at install/uninstall
- (Docker) Official image added
- (Linux) Add aarch64 flatpak package

Changed

- (Windows/Linux/macOS) - Move default config and apps file to assets directory
- (MacOS) Bump boost to 1.80 for macport builds
- (Linux) Remove backup and restore of config files

Fixed

- (Linux) - Create sunshine config directory if it doesn't exist
- (Linux) Remove portable home and config directories for AppImage
- (Windows) Include service install and uninstall scripts again
- (Windows) Automatically delete start menu entry upon uninstall
- (Windows) Automatically delete program install directory upon uninstall, with user prompt
- (Linux) Handle the case of no default audio sink
- (Windows/Linux/macOS) Fix default image paths
- (Linux) Fix CUDA RGBA to NV12 conversion

21.15 0.14.1 - 2022-08-09

Added

- (Linux) Flatpak package added
- (Linux) AUR package automated updates
- (Windows) Winget package automated updates

Changed

- (General) Moved repo to @LizardByte GitHub org
- (WebUI) Fixed button spacing on home page
- (WebUI) Added Discord WidgetBot Crate

Fixed

- (Linux/Mac) Default config and app files now copied to user home directory
- (Windows) Default config and app files now copied to working directory

21.16 0.14.0 - 2022-06-15

Added

- (Documentation) Added Sphinx documentation available at <https://sunshinestream.readthedocs.io/en/latest/>
- (Development) Initial support for Localization
- (Linux) Add rpm package as release asset
- (macOS) Add Portfile as release asset
- (Windows) Add DwmFlush() call to improve capture
- (Windows) Add Windows installer

Fixed

- (AMD) Fixed hwdevice being destroyed before context
- (Linux) Added missing dependencies to AppImage
- (Linux) Fixed rumble events causing game to freeze
- (Linux) Improved Pulse/Pipewire compatibility

- (Linux) Moved to single deb package
- (macOS) Fixed missing TPCircularBuffer submodule
- (Stream) Properly catch exceptions in stream broadcast handlers
- (Stream/Video) AVPacket fix

21.17 0.13.0 - 2022-02-27

Added

- (macOS) Initial support for macOS (#40)

21.18 0.12.0 - 2022-02-13

Added

- New command line argument `--version`
- Custom png poster support

Changed

- Correct software bitrate calculation
- Increase vbv-buFSIZE to 1/10 of requested bitrate
- Improvements to Web UI

21.19 0.11.1 - 2021-10-04

Changed

- (Linux) Fix search path for config file and assets

21.20 0.11.0 - 2021-10-04

Added

- (Linux) Added support for wlroots based compositors on Wayland.
- (Windows) Added an icon for the executable

Changed

- Fixed a bug causing segfault when connecting multiple controllers.
- (Linux) Improved NVENC, it now offloads converting images from RGB to NV12
- (Linux) Fixed a bug causes stuttering

21.21 0.10.1 - 2021-08-21

Changed

- (Linux) Re-enabled KMS

21.22 0.10.0 - 2021-08-20

Added

- Added support for Rumble with gamepads.
- Added support for keyboard shortcuts <— See the README for details.
- (Windows) A very basic script has been added in Sunshine-Windowstools <— This will start Sunshine at boot with the highest privileges which is needed to display the login prompt.

Changed

- Some cosmetic changes to the WebUI.
- The first time the WebUI is opened, it will request the creation of a username/password pair from the user.
- Fixed audio crackling introduced in version 0.8.0
- (Linux) VA-API hardware encoding now works on Intel i7-6700 at least. <— For the best experience, using ffmpeg version 4.3 or higher is recommended.
- (Windows) Installing from debian package shouldn't overwrite your configuration files anymore. <— It's recommended that you back up /etc/sunshine/ before testing this.

21.23 0.9.0 - 2021-07-11

Added

- Added audio encryption
- (Linux) Added basic NVENC support on Linux
- (Windows) The Windows version can now capture the lock screen and the UAC prompt as long as it's run through PsExec.exe <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec>

Changed

- Sunshine will now accept expired or not-yet-valid certificates, as long as they are signed properly.
- Fixed compatibility with iOS version of Moonlight
- Drastically reduced chance of being forced to skip error correction due to video frame size
- (Linux) sunshine.service will be installed automatically.

21.24 0.8.0 - 2021-06-30

Added

- Added mDNS support: Moonlight will automatically find Sunshine.
- Added UPnP support. It's off by default.

21.25 0.7.7 - 2021-06-24

Added

- (Linux) Added installation package for Debian

Changed

- Fixed incorrect scaling for absolute mouse coordinates when using multiple monitors.
- Fixed incorrect colors when scaling for software encoder

21.26 0.7.1 - 2021-06-18

Changed

- (Linux) Fixed an issue where it was impossible to start sunshine on ubuntu 20.04

21.27 0.7.0 - 2021-06-16

Added

- Added a Web Manager. Accessible through: <https://localhost:47990> or <https://:47990>
- (Linux) Added hardware encoding support for AMD on Linux

Changed

- (Linux) Moved certificates and saved pairings generated during runtime to `.config/sunshine` on Linux

21.28 0.6.0 - 2021-05-26

Added

- Added support for surround audio

Changed

- Maintain aspect ratio when scaling video
- Fix issue where Sunshine is forced to drop frames when they are too large

21.29 0.5.0 - 2021-05-13

Added

- Added support for absolute mouse coordinates
- (Linux) Added support for streaming specific monitor on Linux
- (Windows) Added support for AMF on Windows

21.30 0.4.0 - 2020-05-03

Changed

- prep-cmd is now optional in apps.json
- Fixed bug causing video artifacts
- Fixed bug preventing Moonlight from closing app on exit
- Fixed bug causing preventing keyboard keys from repeating on latest version of Moonlight
- Fixed bug causing segfault when another session of sunshine was already running
- Fixed bug causing crash when monitor has resolution 1366x768

21.31 0.3.1 - 2020-04-24

Changed

- Fix a memory leak.

21.32 0.3.0 - 2020-04-23

Changed

- Hardware acceleration on NVidia GPU's for Video encoding on Windows

21.33 0.2.0 - 2020-03-21

Changed

- Multicasting is now supported: You can set the maximum simultaneous connections with the configurable option: channels
- Configuration variables can be overwritten on the command line: "name=value" -> it can be useful to set min_log_level=debug without modifying the configuration file
- Switches to make testing the pairing mechanism more convenient has been added, see "sunshine -help" for details

21.34 0.1.1 - 2020-01-30

Added

- (Linux) Added deb package and service for Linux

21.35 0.1.0 - 2020-01-27

Added

- The first official release for Sunshine!

Symbols

`__kernel_entry` (*C macro*), 134

A

`args` (*C++ type*), 121

`audio` (*C++ type*), 100

`audio::buffer_t` (*C++ type*), 101

`audio::config_t` (*C++ struct*), 101

`audio::config_t::channels` (*C++ member*), 102

`audio::config_t::flags` (*C++ member*), 102

`audio::config_t::flags_e` (*C++ enum*), 101

`audio::config_t::flags_e::HIGH_QUALITY` (*C++ enumerator*), 101

`audio::config_t::flags_e::HOST_AUDIO` (*C++ enumerator*), 101

`audio::config_t::flags_e::MAX_FLAGS` (*C++ enumerator*), 101

`audio::config_t::mask` (*C++ member*), 102

`audio::config_t::packetDuration` (*C++ member*), 102

`audio::opus_stream_config_t` (*C++ struct*), 102

`audio::opus_stream_config_t::bitrate` (*C++ member*), 102

`audio::opus_stream_config_t::channelCount` (*C++ member*), 102

`audio::opus_stream_config_t::coupledStreams` (*C++ member*), 102

`audio::opus_stream_config_t::mapping` (*C++ member*), 102

`audio::opus_stream_config_t::sampleRate` (*C++ member*), 102

`audio::opus_stream_config_t::streams` (*C++ member*), 102

`audio::packet_t` (*C++ type*), 101

`audio::stream_config_e` (*C++ enum*), 101

`audio::stream_config_e::HIGH_STEREO` (*C++ enumerator*), 101

`audio::stream_config_e::HIGH_SURROUND51` (*C++ enumerator*), 101

`audio::stream_config_e::HIGH_SURROUND71` (*C++ enumerator*), 101

`audio::stream_config_e::MAX_STREAM_CONFIG` (*C++ enumerator*), 101

`audio::stream_config_e::STEREO` (*C++ enumerator*), 101

`audio::stream_config_e::SURROUND51` (*C++ enumerator*), 101

`audio::stream_config_e::SURROUND71` (*C++ enumerator*), 101

C

`cbs` (*C++ type*), 102

`cbs::h264_t` (*C++ struct*), 102

`cbs::h264_t::sps` (*C++ member*), 104

`cbs::hevc_t` (*C++ struct*), 104

`cbs::hevc_t::sps` (*C++ member*), 104

`cbs::hevc_t::vps` (*C++ member*), 104

`cbs::nal_t` (*C++ struct*), 104

`cbs::nal_t::_new` (*C++ member*), 104

`cbs::nal_t::old` (*C++ member*), 104

`config` (*C++ type*), 105

`config::audio_t` (*C++ struct*), 105

`config::audio_t::install_steam_drivers` (*C++ member*), 106

`config::audio_t::sink` (*C++ member*), 106

`config::audio_t::virtual_sink` (*C++ member*), 106

`config::ENCRYPTION_MODE_MANDATORY` (*C++ member*), 105

`config::ENCRYPTION_MODE_NEVER` (*C++ member*), 105

`config::ENCRYPTION_MODE_OPPORTUNISTIC` (*C++ member*), 105

`config::flag` (*C++ type*), 112

`config::flag::flag_e` (*C++ enum*), 112

`config::flag::flag_e::CONST_PIN` (*C++ enumerator*), 112

`config::flag::flag_e::FLAG_SIZE` (*C++ enumerator*), 112

`config::flag::flag_e::FORCE_VIDEO_HEADER_REPLACE` (*C++ enumerator*), 112

`config::flag::flag_e::FRESH_STATE` (*C++ enumerator*), 112

`config::flag::flag_e::PIN_STDIN` (C++ *enumerator*), 112
`config::flag::flag_e::UPNP` (C++ *enumerator*), 112
`config::input_t` (C++ *struct*), 106
`config::input_t::always_send_scancodes` (C++ *member*), 106
`config::input_t::back_button_timeout` (C++ *member*), 106
`config::input_t::controller` (C++ *member*), 106
`config::input_t::ds4_back_as_touchpad_click` (C++ *member*), 106
`config::input_t::gamepad` (C++ *member*), 106
`config::input_t::high_resolution_scrolling` (C++ *member*), 106
`config::input_t::key_repeat_delay` (C++ *member*), 106
`config::input_t::key_repeat_period` (C++ *member*), 106
`config::input_t::keybindings` (C++ *member*), 106
`config::input_t::keyboard` (C++ *member*), 106
`config::input_t::motion_as_ds4` (C++ *member*), 106
`config::input_t::mouse` (C++ *member*), 106
`config::input_t::native_pen_touch` (C++ *member*), 106
`config::input_t::touchpad_as_ds4` (C++ *member*), 106
`config::nvhttp_t` (C++ *struct*), 106
`config::nvhttp_t::cert` (C++ *member*), 107
`config::nvhttp_t::external_ip` (C++ *member*), 107
`config::nvhttp_t::file_state` (C++ *member*), 107
`config::nvhttp_t::fps` (C++ *member*), 107
`config::nvhttp_t::origin_web_ui_allowed` (C++ *member*), 107
`config::nvhttp_t::pkey` (C++ *member*), 107
`config::nvhttp_t::resolutions` (C++ *member*), 107
`config::nvhttp_t::sunshine_name` (C++ *member*), 107
`config::prep_cmd_t` (C++ *struct*), 107
`config::prep_cmd_t::do_cmd` (C++ *member*), 107
`config::prep_cmd_t::elevated` (C++ *member*), 107
`config::prep_cmd_t::prep_cmd_t` (C++ *function*), 107
`config::prep_cmd_t::undo_cmd` (C++ *member*), 107
`config::stream_t` (C++ *struct*), 107
`config::stream_t::channels` (C++ *member*), 108
`config::stream_t::fec_percentage` (C++ *member*), 108
`config::stream_t::file_apps` (C++ *member*), 108
`config::stream_t::lan_encryption_mode` (C++ *member*), 108
`config::stream_t::ping_timeout` (C++ *member*), 108
`config::stream_t::wan_encryption_mode` (C++ *member*), 108
`config::sunshine_t` (C++ *struct*), 108
`config::sunshine_t::address_family` (C++ *member*), 108
`config::sunshine_t::cmd` (C++ *member*), 108
`config::sunshine_t::cmd_t` (C++ *struct*), 109
`config::sunshine_t::cmd_t::argc` (C++ *member*), 109
`config::sunshine_t::cmd_t::argv` (C++ *member*), 109
`config::sunshine_t::cmd_t::name` (C++ *member*), 109
`config::sunshine_t::config_file` (C++ *member*), 108
`config::sunshine_t::credentials_file` (C++ *member*), 108
`config::sunshine_t::flags` (C++ *member*), 108
`config::sunshine_t::log_file` (C++ *member*), 108
`config::sunshine_t::min_log_level` (C++ *member*), 109
`config::sunshine_t::password` (C++ *member*), 109
`config::sunshine_t::port` (C++ *member*), 109
`config::sunshine_t::prep_cmds` (C++ *member*), 109
`config::sunshine_t::salt` (C++ *member*), 109
`config::sunshine_t::username` (C++ *member*), 109
`config::video_t` (C++ *struct*), 109
`config::video_t::adapter_name` (C++ *member*), 110
`config::video_t::amd` (C++ *member*), 110
`config::video_t::amd_coder` (C++ *member*), 110
`config::video_t::amd_prealanalysis` (C++ *member*), 110
`config::video_t::amd_quality_av1` (C++ *member*), 110
`config::video_t::amd_quality_h264` (C++ *member*), 110
`config::video_t::amd_quality_hevc` (C++ *member*), 110
`config::video_t::amd_rc_av1` (C++ *member*), 110
`config::video_t::amd_rc_h264` (C++ *member*), 110
`config::video_t::amd_rc_hevc` (C++ *member*), 110
`config::video_t::amd_usage_av1` (C++ *member*), 110
`config::video_t::amd_usage_h264` (C++ *member*), 110
`config::video_t::amd_usage_hevc` (C++ *member*), 110
`config::video_t::amd_vbaq` (C++ *member*), 110
`config::video_t::aq` (C++ *member*), 110
`config::video_t::av1_mode` (C++ *member*), 110

config::video_t::capture (C++ member), 110
 config::video_t::encoder (C++ member), 110
 config::video_t::h264_coder (C++ member), 110
 config::video_t::hevc_mode (C++ member), 110
 config::video_t::min_threads (C++ member), 110
 config::video_t::multipass (C++ member), 111
 config::video_t::nv (C++ member), 111
 config::video_t::nv_legacy (C++ member), 111
 config::video_t::nv_opengl_vulkan_on_dxgi (C++ member), 111
 config::video_t::nv_realtime_hags (C++ member), 111
 config::video_t::nv_sunshine_high_power_mode (C++ member), 111
 config::video_t::output_name (C++ member), 111
 config::video_t::preset (C++ member), 111
 config::video_t::qp (C++ member), 111
 config::video_t::qsv (C++ member), 111
 config::video_t::qsv_cavlc (C++ member), 111
 config::video_t::qsv_preset (C++ member), 111
 config::video_t::qsv_slow_hevc (C++ member), 111
 config::video_t::svtav1_preset (C++ member), 111
 config::video_t::sw (C++ member), 111
 config::video_t::sw_preset (C++ member), 111
 config::video_t::sw_tune (C++ member), 111
 config::video_t::vbv_percentage_increase (C++ member), 111
 config::video_t::vt (C++ member), 111
 config::video_t::vt_allow_sw (C++ member), 111
 config::video_t::vt_coder (C++ member), 111
 config::video_t::vt_realtime (C++ member), 112
 config::video_t::vt_require_sw (C++ member), 112
 confighttp (C++ type), 113
 confighttp::PORT_HTTPS (C++ member), 113
 crypto (C++ type), 113
 crypto::aes_t (C++ type), 114
 crypto::bignum_t (C++ type), 114
 crypto::bio_t (C++ type), 114
 crypto::cert_chain_t (C++ class), 114
 crypto::cert_chain_t::_cert_ctx (C++ member), 115
 crypto::cert_chain_t::_certs (C++ member), 115
 crypto::cert_chain_t::add (C++ function), 115
 crypto::cert_chain_t::cert_chain_t (C++ function), 115
 crypto::cert_chain_t::operator= (C++ function), 115
 crypto::cert_chain_t::verify (C++ function), 115
 crypto::cipher (C++ type), 115
 crypto::cipher::cbc_t (C++ class), 115
 crypto::cipher::cbc_t::cbc_t (C++ function), 117
 crypto::cipher::cbc_t::encrypt (C++ function), 117
 crypto::cipher::cbc_t::operator= (C++ function), 117
 crypto::cipher::cipher_t (C++ class), 117
 crypto::cipher::cipher_t::decrypt_ctx (C++ member), 118
 crypto::cipher::cipher_t::encrypt_ctx (C++ member), 118
 crypto::cipher::cipher_t::key (C++ member), 118
 crypto::cipher::cipher_t::padding (C++ member), 118
 crypto::cipher::ecb_t (C++ class), 118
 crypto::cipher::ecb_t::decrypt (C++ function), 119
 crypto::cipher::ecb_t::ecb_t (C++ function), 119
 crypto::cipher::ecb_t::encrypt (C++ function), 119
 crypto::cipher::ecb_t::operator= (C++ function), 119
 crypto::cipher::gcm_t (C++ class), 119
 crypto::cipher::gcm_t::decrypt (C++ function), 119
 crypto::cipher::gcm_t::encrypt (C++ function), 119
 crypto::cipher::gcm_t::gcm_t (C++ function), 119
 crypto::cipher::gcm_t::operator= (C++ function), 119
 crypto::cipher::round_to_pkcs7_padded (C++ function), 115
 crypto::cipher::tag_size (C++ member), 115
 crypto::cipher_ctx_t (C++ type), 114
 crypto::creds_t (C++ struct), 115
 crypto::creds_t::pkey (C++ member), 115
 crypto::creds_t::x509 (C++ member), 115
 crypto::md_ctx_t (C++ type), 114
 crypto::pkey_ctx_t (C++ type), 114
 crypto::pkey_t (C++ type), 114
 crypto::sha256_t (C++ type), 114
 crypto::x509_store_ctx_t (C++ type), 114
 crypto::x509_store_t (C++ type), 114
 crypto::x509_t (C++ type), 114

D

debug (C++ member), 127
 display_cursor (C++ member), 122
 dyn (C++ type), 174

E

egl (C++ type), 171
 error (C++ member), 127

F

`fatal` (C++ member), 127
`file_handler` (C++ type), 121

H

`http` (C++ type), 123

I

`info` (C++ member), 127
`input` (C++ type), 124
`input::touch_port_t` (C++ struct), 124
`input::touch_port_t::client_offsetX` (C++ member), 126
`input::touch_port_t::client_offsetY` (C++ member), 126
`input::touch_port_t::env_height` (C++ member), 126
`input::touch_port_t::env_width` (C++ member), 126
`input::touch_port_t::scalar_inv` (C++ member), 126

L

`launch_ui` (C++ function), 121
`launch_ui_with_path` (C++ function), 121
`lifetime` (C++ type), 121
`log_flush` (C++ function), 127

M

`MAIL` (C macro), 122
`mail` (C++ type), 122
`mail::audio_packets` (C++ member), 123
`mail::broadcast_shutdown` (C++ member), 123
`mail::gamepad_feedback` (C++ member), 123
`mail::hdr` (C++ member), 123
`mail::idr` (C++ member), 123
`mail::invalidate_ref_frames` (C++ member), 123
`mail::man` (C++ member), 123
`mail::shutdown` (C++ member), 123
`mail::switch_display` (C++ member), 123
`mail::touch_port` (C++ member), 123
`mail::video_packets` (C++ member), 123
`main` (C++ function), 128
`mime_types` (C++ member), 113
`move_by_copy_util` (C++ type), 128
`move_by_copy_util::cmove` (C++ function), 129
`move_by_copy_util::const_cmove` (C++ function), 129
`move_by_copy_util::MoveByCopy` (C++ class), 129
`move_by_copy_util::MoveByCopy::_to_move` (C++ member), 131
`move_by_copy_util::MoveByCopy::move_type` (C++ type), 130

`move_by_copy_util::MoveByCopy::MoveByCopy` (C++ function), 130
`move_by_copy_util::MoveByCopy::operator`
`move_type` (C++ function), 130
`move_by_copy_util::MoveByCopy::operator=`
(C++ function), 130

N

`net` (C++ type), 131
`net::af_e` (C++ enum), 132
`net::af_e::BOTH` (C++ enumerator), 132
`net::af_e::IPV4` (C++ enumerator), 132
`net::host_t` (C++ type), 131
`net::net_e` (C++ enum), 132
`net::net_e::LAN` (C++ enumerator), 132
`net::net_e::PC` (C++ enumerator), 132
`net::net_e::WAN` (C++ enumerator), 132
`net::packet_t` (C++ type), 131
`net::peer_t` (C++ type), 131
`nvhttp` (C++ type), 132
`nvhttp::GFE_VERSION` (C++ member), 133
`nvhttp::PORT_HTTP` (C++ member), 133
`nvhttp::PORT_HTTPS` (C++ member), 133
`nvhttp::VERSION` (C++ member), 133

P

`platf` (C++ type), 175
`print_help` (C++ function), 127
`proc` (C++ type), 134
`proc::cmd_t` (C++ type), 134
`proc::ctx_t` (C++ struct), 134
`proc::ctx_t::auto_detach` (C++ member), 134
`proc::ctx_t::cmd` (C++ member), 134
`proc::ctx_t::detached` (C++ member), 134
`proc::ctx_t::elevated` (C++ member), 134
`proc::ctx_t::exit_timeout` (C++ member), 134
`proc::ctx_t::id` (C++ member), 134
`proc::ctx_t::image_path` (C++ member), 134
`proc::ctx_t::name` (C++ member), 134
`proc::ctx_t::output` (C++ member), 135
`proc::ctx_t::prep_cmds` (C++ member), 135
`proc::ctx_t::wait_all` (C++ member), 135
`proc::ctx_t::working_dir` (C++ member), 135
`proc::file_t` (C++ type), 134
`proc::proc_t` (C++ class), 135
`proc::proc_t::_app` (C++ member), 135
`proc::proc_t::_app_id` (C++ member), 135
`proc::proc_t::_app_launch_time` (C++ member), 135
`proc::proc_t::_app_prep_begin` (C++ member), 135
`proc::proc_t::_app_prep_it` (C++ member), 136
`proc::proc_t::_apps` (C++ member), 136
`proc::proc_t::_env` (C++ member), 136

proc::proc_t::_pipe (C++ member), 136
 proc::proc_t::_process (C++ member), 136
 proc::proc_t::_process_group (C++ member), 136
 proc::proc_t::~~proc_t (C++ function), 135
 proc::proc_t::execute (C++ function), 135
 proc::proc_t::get_app_image (C++ function), 135
 proc::proc_t::get_apps (C++ function), 135
 proc::proc_t::get_last_run_app_name (C++ function), 135
 proc::proc_t::operator= (C++ function), 135
 proc::proc_t::placebo (C++ member), 136
 proc::proc_t::proc_t (C++ function), 135
 proc::proc_t::running (C++ function), 135
 proc::proc_t::terminate (C++ function), 135

R

round_robin_util (C++ type), 136
 round_robin_util::it_wrap_t (C++ class), 137
 round_robin_util::it_wrap_t::_this (C++ function), 138
 round_robin_util::it_wrap_t::const_pointer (C++ type), 137
 round_robin_util::it_wrap_t::const_reference (C++ type), 137
 round_robin_util::it_wrap_t::diff_t (C++ type), 137
 round_robin_util::it_wrap_t::difference_type (C++ type), 137
 round_robin_util::it_wrap_t::iterator (C++ type), 137
 round_robin_util::it_wrap_t::iterator_category (C++ type), 137
 round_robin_util::it_wrap_t::operator!= (C++ function), 138
 round_robin_util::it_wrap_t::operator* (C++ function), 138
 round_robin_util::it_wrap_t::operator+ (C++ function), 138
 round_robin_util::it_wrap_t::operator++ (C++ function), 138
 round_robin_util::it_wrap_t::operator+= (C++ function), 138
 round_robin_util::it_wrap_t::operator== (C++ function), 138
 round_robin_util::it_wrap_t::operator- (C++ function), 138
 round_robin_util::it_wrap_t::operator-= (C++ function), 138
 round_robin_util::it_wrap_t::operator-- (C++ function), 138
 round_robin_util::it_wrap_t::operator-> (C++ function), 138
 round_robin_util::it_wrap_t::operator> (C++ function), 138
 round_robin_util::it_wrap_t::operator>= (C++ function), 138
 round_robin_util::it_wrap_t::operator< (C++ function), 138
 round_robin_util::it_wrap_t::operator<= (C++ function), 138
 round_robin_util::it_wrap_t::pointer (C++ type), 137
 round_robin_util::it_wrap_t::reference (C++ type), 137
 round_robin_util::it_wrap_t::value_type (C++ type), 137
 round_robin_util::make_round_robin (C++ function), 137
 round_robin_util::round_robin_t (C++ class), 138
 round_robin_util::round_robin_t::_begin (C++ member), 140
 round_robin_util::round_robin_t::_end (C++ member), 140
 round_robin_util::round_robin_t::_pos (C++ member), 140
 round_robin_util::round_robin_t::dec (C++ function), 140
 round_robin_util::round_robin_t::eq (C++ function), 140
 round_robin_util::round_robin_t::get (C++ function), 140
 round_robin_util::round_robin_t::inc (C++ function), 140
 round_robin_util::round_robin_t::iterator (C++ type), 140
 round_robin_util::round_robin_t::pointer (C++ type), 140
 round_robin_util::round_robin_t::round_robin_t (C++ function), 140
 rtsp_stream (C++ type), 140
 rtsp_stream::launch_session_t (C++ struct), 141
 rtsp_stream::launch_session_t::appid (C++ member), 141
 rtsp_stream::launch_session_t::av_ping_payload (C++ member), 141
 rtsp_stream::launch_session_t::control_connect_data (C++ member), 141
 rtsp_stream::launch_session_t::enable_hdr (C++ member), 141
 rtsp_stream::launch_session_t::enable_sops (C++ member), 141
 rtsp_stream::launch_session_t::fps (C++ member), 141
 rtsp_stream::launch_session_t::gcm_key (C++ member), 141
 rtsp_stream::launch_session_t::gcmmap (C++ member), 141

rtsp_stream::launch_session_t::height (C++ member), 141
 rtsp_stream::launch_session_t::host_audio (C++ member), 141
 rtsp_stream::launch_session_t::id (C++ member), 141
 rtsp_stream::launch_session_t::iv (C++ member), 141
 rtsp_stream::launch_session_t::rtsp_cipher (C++ member), 141
 rtsp_stream::launch_session_t::rtsp_iv_counters (C++ member), 142
 rtsp_stream::launch_session_t::rtsp_url_schemes (C++ member), 142
 rtsp_stream::launch_session_t::surround_info (C++ member), 142
 rtsp_stream::launch_session_t::unique_id (C++ member), 142
 rtsp_stream::launch_session_t::width (C++ member), 142
 rtsp_stream::RTSP_SETUP_PORT (C++ member), 141

S

safe (C++ type), 154
 safe::alarm_raw_t (C++ class), 155
 safe::alarm_raw_t::_cv (C++ member), 156
 safe::alarm_raw_t::_lock (C++ member), 156
 safe::alarm_raw_t::_rang (C++ member), 156
 safe::alarm_raw_t::_status (C++ member), 156
 safe::alarm_raw_t::reset (C++ function), 156
 safe::alarm_raw_t::ring (C++ function), 156
 safe::alarm_raw_t::status (C++ function), 156
 safe::alarm_raw_t::status_t (C++ type), 155
 safe::alarm_raw_t::wait (C++ function), 156
 safe::alarm_raw_t::wait_for (C++ function), 156
 safe::alarm_raw_t::wait_until (C++ function), 156
 safe::alarm_t (C++ type), 155
 safe::cleanup (C++ function), 155
 safe::event_t (C++ class), 156
 safe::event_t::_continue (C++ member), 157
 safe::event_t::_cv (C++ member), 157
 safe::event_t::_lock (C++ member), 157
 safe::event_t::_status (C++ member), 157
 safe::event_t::peek (C++ function), 157
 safe::event_t::pop (C++ function), 157
 safe::event_t::raise (C++ function), 157
 safe::event_t::reset (C++ function), 157
 safe::event_t::running (C++ function), 157
 safe::event_t::status_t (C++ type), 156
 safe::event_t::stop (C++ function), 157
 safe::event_t::view (C++ function), 157
 safe::lock (C++ function), 155
 safe::mail_raw_t (C++ class), 157
 safe::mail_raw_t::cleanup (C++ function), 158
 safe::mail_raw_t::event (C++ function), 158
 safe::mail_raw_t::event_t (C++ type), 158
 safe::mail_raw_t::id_to_post (C++ member), 158
 safe::mail_raw_t::mutex (C++ member), 158
 safe::mail_raw_t::queue (C++ function), 158
 safe::mail_raw_t::queue_t (C++ type), 158
 safe::mail_t (C++ type), 155
 safe::make_alarm (C++ function), 155
 safe::make_shared (C++ function), 155
 safe::post_t (C++ class), 158
 safe::post_t::~~post_t (C++ function), 160
 safe::post_t::mail (C++ member), 160
 safe::post_t::post_t (C++ function), 160
 safe::queue_t (C++ class), 160
 safe::queue_t::_continue (C++ member), 160
 safe::queue_t::_cv (C++ member), 160
 safe::queue_t::_lock (C++ member), 160
 safe::queue_t::_max_elements (C++ member), 160
 safe::queue_t::_queue (C++ member), 160
 safe::queue_t::peek (C++ function), 160
 safe::queue_t::pop (C++ function), 160
 safe::queue_t::queue_t (C++ function), 160
 safe::queue_t::raise (C++ function), 160
 safe::queue_t::running (C++ function), 160
 safe::queue_t::status_t (C++ type), 160
 safe::queue_t::stop (C++ function), 160
 safe::queue_t::unsafe (C++ function), 160
 safe::shared_t (C++ class), 161
 safe::shared_t::_construct (C++ member), 161
 safe::shared_t::_count (C++ member), 161
 safe::shared_t::_destruct (C++ member), 161
 safe::shared_t::_lock (C++ member), 161
 safe::shared_t::_object_buf (C++ member), 161
 safe::shared_t::construct_f (C++ type), 161
 safe::shared_t::destruct_f (C++ type), 161
 safe::shared_t::element_type (C++ type), 161
 safe::shared_t::ptr_t (C++ struct), 161
 safe::shared_t::ptr_t::~~ptr_t (C++ function), 162
 safe::shared_t::ptr_t::get (C++ function), 161
 safe::shared_t::ptr_t::operator bool (C++ function), 161
 safe::shared_t::ptr_t::operator= (C++ function), 161
 safe::shared_t::ptr_t::operator-> (C++ function), 161
 safe::shared_t::ptr_t::owner (C++ member), 162
 safe::shared_t::ptr_t::ptr_t (C++ function), 162
 safe::shared_t::ptr_t::release (C++ function), 162
 safe::shared_t::ref (C++ function), 161
 safe::shared_t::shared_t (C++ function), 161
 safe::signal_t (C++ type), 155

sink (C++ member), 127
 stream (C++ type), 142
 stream::AUDIO_STREAM_PORT (C++ member), 143
 stream::config_t (C++ struct), 143
 stream::config_t::audio (C++ member), 143
 stream::config_t::audioQosType (C++ member), 143
 stream::config_t::controlProtocolType (C++ member), 143
 stream::config_t::encryptionFlagsEnabled (C++ member), 143
 stream::config_t::gcmmap (C++ member), 143
 stream::config_t::minRequiredFecPackets (C++ member), 143
 stream::config_t::mlFeatureFlags (C++ member), 143
 stream::config_t::monitor (C++ member), 144
 stream::config_t::packetSize (C++ member), 144
 stream::config_t::videoQosType (C++ member), 144
 stream::CONTROL_PORT (C++ member), 143
 stream::session (C++ type), 144
 stream::session::state_e (C++ enum), 144
 stream::session::state_e::RUNNING (C++ enumerator), 144
 stream::session::state_e::STARTING (C++ enumerator), 144
 stream::session::state_e::STOPPED (C++ enumerator), 144
 stream::session::state_e::STOPPING (C++ enumerator), 144
 stream::VIDEO_STREAM_PORT (C++ member), 143
 sync_util (C++ type), 144
 sync_util::sync_t (C++ class), 145
 sync_util::sync_t::_lock (C++ member), 146
 sync_util::sync_t::lock (C++ function), 145
 sync_util::sync_t::mutex_t (C++ type), 145
 sync_util::sync_t::operator* (C++ function), 145
 sync_util::sync_t::operator= (C++ function), 145
 sync_util::sync_t::operator-> (C++ function), 145
 sync_util::sync_t::raw (C++ member), 146
 sync_util::sync_t::sync_t (C++ function), 145
 sync_util::sync_t::value_t (C++ type), 145
 system_tray (C++ type), 146
 system_tray::end_tray (C++ function), 146
 system_tray::run_tray (C++ function), 146
 system_tray::system_tray (C++ function), 146
 system_tray::tray_donate_github_cb (C++ function), 146
 system_tray::tray_donate_mee6_cb (C++ function), 146
 system_tray::tray_donate_patreon_cb (C++ function), 146

system_tray::tray_donate_paypal_cb (C++ function), 146
 system_tray::tray_open_ui_cb (C++ function), 146
 system_tray::tray_quit_cb (C++ function), 146
 system_tray::update_tray_pausing (C++ function), 147
 system_tray::update_tray_playing (C++ function), 147
 system_tray::update_tray_require_pin (C++ function), 147
 system_tray::update_tray_stopped (C++ function), 147

T

task_pool (C++ member), 122
 task_pool_util (C++ type), 147
 task_pool_util::_Impl (C++ class), 147
 task_pool_util::_Impl::_func (C++ member), 149
 task_pool_util::_Impl::_Impl (C++ function), 149
 task_pool_util::_Impl::run (C++ function), 149
 task_pool_util::_ImplBase (C++ class), 149
 task_pool_util::_ImplBase::~_ImplBase (C++ function), 149
 task_pool_util::_ImplBase::run (C++ function), 149
 task_pool_util::TaskPool (C++ class), 149
 task_pool_util::TaskPool::__task (C++ type), 150
 task_pool_util::TaskPool::__time_point (C++ type), 150
 task_pool_util::TaskPool::_task_mutex (C++ member), 151
 task_pool_util::TaskPool::_tasks (C++ member), 151
 task_pool_util::TaskPool::_timer_tasks (C++ member), 151
 task_pool_util::TaskPool::cancel (C++ function), 150
 task_pool_util::TaskPool::delay (C++ function), 150
 task_pool_util::TaskPool::next (C++ function), 150
 task_pool_util::TaskPool::operator= (C++ function), 150
 task_pool_util::TaskPool::pop (C++ function), 150
 task_pool_util::TaskPool::push (C++ function), 150
 task_pool_util::TaskPool::pushDelayed (C++ function), 150
 task_pool_util::TaskPool::ready (C++ function), 150
 task_pool_util::TaskPool::task_id_t (C++ type), 150

task_pool_util::TaskPool::TaskPool (C++ *function*), 151
 task_pool_util::TaskPool::timer_task_t (C++ *class*), 151
 task_pool_util::TaskPool::timer_task_t::future (C++ *member*), 152
 task_pool_util::TaskPool::timer_task_t::task_id (C++ *member*), 152
 task_pool_util::TaskPool::timer_task_t::timer_task_t (C++ *function*), 151
 task_pool_util::TaskPool::toRunnable (C++ *function*), 151
 text_sink (C++ *type*), 127
 thread_pool_util (C++ *type*), 152
 thread_pool_util::ThreadPool (C++ *class*), 152
 thread_pool_util::ThreadPool::__task (C++ *type*), 154
 thread_pool_util::ThreadPool::_continue (C++ *member*), 154
 thread_pool_util::ThreadPool::_cv (C++ *member*), 154
 thread_pool_util::ThreadPool::_lock (C++ *member*), 154
 thread_pool_util::ThreadPool::_main (C++ *function*), 154
 thread_pool_util::ThreadPool::_thread (C++ *member*), 154
 thread_pool_util::ThreadPool::~ThreadPool (C++ *function*), 154
 thread_pool_util::ThreadPool::join (C++ *function*), 154
 thread_pool_util::ThreadPool::push (C++ *function*), 154
 thread_pool_util::ThreadPool::pushDelayed (C++ *function*), 154
 thread_pool_util::ThreadPool::start (C++ *function*), 154
 thread_pool_util::ThreadPool::stop (C++ *function*), 154
 thread_pool_util::ThreadPool::ThreadPool (C++ *function*), 154

U

upnp (C++ *type*), 162
 uuid_util (C++ *type*), 163
 uuid_util::uuid_t (C++ *union*), 163
 uuid_util::uuid_t::b16 (C++ *member*), 164
 uuid_util::uuid_t::b32 (C++ *member*), 164
 uuid_util::uuid_t::b64 (C++ *member*), 164
 uuid_util::uuid_t::b8 (C++ *member*), 164
 uuid_util::uuid_t::generate (C++ *function*), 164
 uuid_util::uuid_t::operator== (C++ *function*), 164
 uuid_util::uuid_t::operator> (C++ *function*), 164

V

va (C++ *type*), 171
 verbose (C++ *member*), 127
 video (C++ *type*), 165
 video::config_t (C++ *struct*), 165
 video::config_t::bitrate (C++ *member*), 165
 video::config_t::dynamicRange (C++ *member*), 165
 video::config_t::encoderCscMode (C++ *member*), 165
 video::config_t::framerate (C++ *member*), 165
 video::config_t::height (C++ *member*), 165
 video::config_t::numRefFrames (C++ *member*), 165
 video::config_t::slicesPerFrame (C++ *member*), 166
 video::config_t::videoFormat (C++ *member*), 166
 video::config_t::width (C++ *member*), 166
 video::hdr_info_raw_t (C++ *struct*), 166
 video::hdr_info_raw_t::enabled (C++ *member*), 166
 video::hdr_info_raw_t::hdr_info_raw_t (C++ *function*), 166
 video::hdr_info_raw_t::metadata (C++ *member*), 166
 video::hdr_info_t (C++ *type*), 165
 video::packet_raw_avcodec (C++ *struct*), 166
 video::packet_raw_avcodec::~packet_raw_avcodec (C++ *function*), 167
 video::packet_raw_avcodec::av_packet (C++ *member*), 167
 video::packet_raw_avcodec::data (C++ *function*), 167
 video::packet_raw_avcodec::data_size (C++ *function*), 167
 video::packet_raw_avcodec::frame_index (C++ *function*), 167
 video::packet_raw_avcodec::is_idr (C++ *function*), 167
 video::packet_raw_avcodec::packet_raw_avcodec (C++ *function*), 167
 video::packet_raw_generic (C++ *struct*), 167
 video::packet_raw_generic::data (C++ *function*), 167
 video::packet_raw_generic::data_size (C++ *function*), 167
 video::packet_raw_generic::frame_data (C++ *member*), 169
 video::packet_raw_generic::frame_index (C++ *function*), 167

[video::packet_raw_generic::idr \(C++ member\), 169](#)
[video::packet_raw_generic::index \(C++ member\), 169](#)
[video::packet_raw_generic::is_idr \(C++ function\), 167](#)
[video::packet_raw_generic::packet_raw_generic \(C++ function\), 167](#)
[video::packet_raw_t \(C++ struct\), 169](#)
[video::packet_raw_t::~~packet_raw_t \(C++ function\), 169](#)
[video::packet_raw_t::after_ref_frame_invalidation \(C++ member\), 169](#)
[video::packet_raw_t::channel_data \(C++ member\), 169](#)
[video::packet_raw_t::data \(C++ function\), 169](#)
[video::packet_raw_t::data_size \(C++ function\), 169](#)
[video::packet_raw_t::frame_index \(C++ function\), 169](#)
[video::packet_raw_t::frame_timestamp \(C++ member\), 169](#)
[video::packet_raw_t::is_idr \(C++ function\), 169](#)
[video::packet_raw_t::replace_t \(C++ struct\), 170](#)
[video::packet_raw_t::replace_t::_new \(C++ member\), 170](#)
[video::packet_raw_t::replace_t::old \(C++ member\), 170](#)
[video::packet_raw_t::replace_t::operator= \(C++ function\), 170](#)
[video::packet_raw_t::replace_t::replace_t \(C++ function\), 170](#)
[video::packet_raw_t::replacements \(C++ member\), 169](#)
[video::packet_t \(C++ type\), 165](#)

W

[warning \(C++ member\), 127](#)
[WEB_DIR \(C macro\), 113](#)
[wl \(C++ type\), 171](#)
[wl::monitor_t \(C++ class\), 171](#)
[wl::monitor_t::description \(C++ member\), 173](#)
[wl::monitor_t::listen \(C++ function\), 173](#)
[wl::monitor_t::monitor_t \(C++ function\), 173](#)
[wl::monitor_t::name \(C++ member\), 173](#)
[wl::monitor_t::operator= \(C++ function\), 173](#)
[wl::monitor_t::output \(C++ member\), 173](#)
[wl::monitor_t::viewport \(C++ member\), 173](#)