
Sunshine

ReenigneArcher

Jan 09, 2023

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	Installation	3
2.1	Binaries	3
2.2	Docker	3
2.3	Linux	3
2.4	macOS	6
2.5	Windows	7
3	Docker	9
3.1	Build your own containers	9
3.2	Where used	9
3.3	Port and Volume mappings	9
3.4	Supported Architectures	11
4	Third Party Packages	13
4.1	Chocolatey	13
4.2	nixpkgs	13
4.3	Scoop	13
4.4	Solus	13
4.5	Winget	13
4.6	Legacy GitHub Repo	14
5	Usage	15
5.1	Network	16
5.2	Arguments	16
5.3	Setup	16
5.4	Shortcuts	19
5.5	Application List	19
5.6	Considerations	20
5.7	Tutorials	20
6	Advanced Usage	21
6.1	Performance Tips	21
6.2	Configuration	21

6.3	General	22
6.4	Controls	23
6.5	Display	25
6.6	Audio	28
6.7	Network	29
6.8	Encoding	32
6.9	Advanced	41
7	GameStream	43
7.1	Migration	43
7.2	Limitations	43
8	General	45
8.1	Forgotten Credentials	45
8.2	Web UI Access	45
8.3	Nvidia issues	45
9	Linux	47
9.1	KMS Streaming fails	47
10	macOS	49
10.1	Dynamic session lookup failed	49
11	Windows	51
11.1	No gamepad detected	51
12	Build	53
12.1	Building Locally	53
12.2	Remote Build	53
13	Linux	55
13.1	Requirements	55
13.2	npm dependencies	58
13.3	Build	58
14	macOS	59
14.1	Requirements	59
14.2	npm dependencies	59
14.3	Build	60
15	Windows	61
15.1	Requirements	61
15.2	npm dependencies	61
15.3	Build	61
16	Contributing	63
17	Localization	65
17.1	CrowdIn	66
17.2	Extraction	66
18	Testing	69
18.1	Clang Format	69
18.2	Sphinx	69
18.3	Unit Testing	69

19 Legal	71
19.1 Commercial Use	71

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: 11.7+ Linux/Debian: 11 (bullseye) Linux/Fedora: 36+ 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

INSTALLATION

The recommended method for running Sunshine is to use the *binaries* bundled with the *latest release*.

Attention: Additional setup is required after installation. See *Setup*.

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*.

2.2 Docker

Docker images are available on [Dockerhub.io](https://hub.docker.com/r/sunshine-slim) and ghcr.io.

See *Docker* for additional information.

2.3 Linux

Follow the instructions for your preferred package type below.

2.3.1 AppImage

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

- [×] Debian oldstable (buster)
- [✓] Debian stable (bullseye)
- [✓] Debian testing (bookworm)
- [✓] Debian unstable (sid)
- [✓] Ubuntu kinetic
- [✓] Ubuntu jammy

- [✓] Ubuntu focal
- [×] Ubuntu bionic
- [×] Ubuntu xenial
- [×] Ubuntu trusty
- [×] CentOS 7

1. Download `sunshine.AppImage` to your home directory.
2. Open terminal and run the following code.

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

Start:

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

Uninstall:

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

2.3.2 AUR Package

1. Open terminal and run the following code.

```
git clone https://aur.archlinux.org/sunshine.git
cd sunshine
makepkg -fi
```

Uninstall:

```
pacman -R sunshine
```

2.3.3 Debian Package

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

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

Note: The `{ubuntu-version}` is the version of ubuntu we built the package on. If you are not using Ubuntu and have an issue with one package, you can try another.

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

Uninstall:

```
sudo apt remove sunshine
```

2.3.4 Flatpak Package

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
```

2.3.5 RPM Package

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.rpm` and run the following code.

```
sudo dnf install ./sunshine.rpm
```

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

Uninstall:

```
sudo dnf remove sunshine
```

2.4 macOS

Sunshine on macOS is experimental. Gamepads do not work. Other features may not work as expected.

2.4.1 pkg

Warning: The *pkg* does not include runtime dependencies.

1. Download the `sunshine.pkg` file and install it as normal.

Uninstall:

```
cd /etc/sunshine/assets  
uninstall_pkg.sh
```

2.4.2 Portfile

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 the Portfile to `~/Downloads` and run the following code.

```
mkdir -p ~/ports/multimedia/sunshine  
mv ~/Downloads/Portfile ~/ports/multimedia/sunshine/  
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.

Uninstall:

```
sudo port uninstall sunshine
```

2.5 Windows

2.5.1 Installer

1. Download and install `sunshine-windows.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.

2.5.2 Standalone

1. Download and extract `sunshine-windows.zip`

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

3.1 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.

```
FROM lizardbyte/sunshine

# install Steam, Wayland, etc.

ENTRYPOINT steam && sunshine
```

3.2 Where used

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

- [Games on Whales](#)

3.3 Port and Volume mappings

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

3.3.1 Using docker run

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

```
docker run -d \
  --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>
```

3.3.2 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>
      - TZ=<timezone>
    ports:
      - "47984-47990:47984-47990/tcp"
      - "48010:48010"
      - "47998-48000:47998-48000/udp"
```

3.3.3 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.

3.4 Supported Architectures

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

The architectures supported by this image are:

Architecture	Available
x86-64	
arm64	

THIRD PARTY PACKAGES

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

4.1 Chocolatey

4.2 nixpkgs

4.3 Scoop

4.4 Solus

4.5 Winget

4.6 Legacy GitHub Repo

Attention: This repo is not maintained. Thank you to Loki for bringing this amazing project to life!

USAGE

1. See the [setup](#) section for your specific OS.
2. If you did not install the service, then start sunshine with the following command, unless a start command is listed in the specified package [installation](#) instructions.

Note: A service is a process that runs in the background. 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.

3. 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.

Add games and applications.

This can be configured in the web ui.

Note: Additionally, apps can be configured manually. *src_assets/<os>/config/apps.json* is an example of a list of applications that are started just before running a stream. This is the directory within the GitHub

repo.

4. In Moonlight, you may need to add the PC manually.
5. When Moonlight request you insert the correct pin on sunshine:
 - 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

5.1 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.

5.2 Arguments

To get a list of available arguments run the following:

```
sunshine --help
```

5.3 Setup

5.3.1 Linux

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

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

1. Add user to group *input*, if this is the first time installing.

```
sudo usermod -a -G input $USER
```

2. Create *udev* rules.

```
echo 'KERNEL=="uinput", GROUP="input", MODE="0660", OPTIONS+="static_node=uinput
→ "' | \
sudo tee /etc/udev/rules.d/85-sunshine-input.rules
```

3. Optionally, configure autostart service
 - filename: `~/.config/systemd/user/sunshine.service`
 - contents:

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

[Service]
ExecStart=<see table>
#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. Additional Setup for KMS

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.

Enable

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

Disable (for Xorg/X11)

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

5. Reboot

```
sudo reboot now
```

5.3.2 macOS

Sunshine can only access microphones on macOS due to system limitations. To stream system audio use [Soundflower](#) or [BlackHole](#) and select their sink as audio device in *sunshine.conf*.

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

Caution: Gamepads are not currently supported.

Configure autostart service

MacPorts

```
sudo port load Sunshine
```

5.3.3 Windows

For gamepad support, install [ViGEmBus](#)

Sunshine firewall

Add rule

```
cd /d "C:\Program Files\Sunshine\scripts"  
add-firewall-rule.bat
```

Remove rule

```
cd /d "C:\Program Files\Sunshine\scripts"  
remove-firewall-rule.bat
```

Sunshine service

Enable

```
cd /d "C:\Program Files\Sunshine\scripts"  
install-service.bat
```

Disable

```
cd /d "C:\Program Files\Sunshine\scripts"  
uninstall-service.bat
```


5.4 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/F13 - Switch to different monitor for Streaming

5.5 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 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

- `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.

5.6 Considerations

- 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.
- In addition to the apps listed, one app “Desktop” is hardcoded into Sunshine. It does not start an application, instead it simply starts a stream.
- For the Linux flatpak you must prepend commands with `flatpak-spawn --host`.

5.7 Tutorials

Tutorial videos are available [here](#).

Community!

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

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

6.1.1 AMD

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

6.1.2 Nvidia

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
```

To manually configure sunshine you may edit the `conf` file in a text editor. Use the examples as reference.

Hint: Some settings are not available within the web ui.

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 log_path

Description

The path where the sunshine log is stored.

Default

sunshine.log

Example

```
log_path = sunshine.log
```

6.4 Controls

6.4.1 gamepad

Description

The type of gamepad to emulate on the host.

Caution: Applies to Windows only.

Choices

Value	Description
x360	xbox 360 controller
ds4	dualshock controller (PS4)

Default

x360

Example

```
gamepad = x360
```

6.4.2 back_button_timeout

Description

If, after the timeout, the back/select button is still pressed down, Home/Guide button press is emulated.

On Nvidia Shield, the home and power button are not passed to Moonlight.

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

Default

2000

Example

```
back_button_timeout = 2000
```

6.4.3 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
```

6.4.4 key_repeat_frequency

Description

How often keys repeat every second.

Tip: This configurable option supports decimals.

Default

Todo: Unknown

Example

```
key_repeat_frequency = 24.9
```

6.4.5 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

None

Example

```
keybindings = [  
    0x10, 0xA0,  
    0x11, 0xA2,  
    0x12, 0xA4,  
    0x4A, 0x4B  
]
```

6.4.6 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

None

Example

```
key_rightalt_to_key_win = enabled
```

6.5 Display

6.5.1 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
```

Default

Sunshine will select the default video card.

Examples

Linux

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

Sunshine

Todo: macOS

Windows

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

6.5.2 output_name

Description

Select the display number you want to stream.

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

Linux

```
xrandr --listmonitors
```

Example output: `0: +HDMI-1 1920/518x1200/324+0+0 HDMI-1`

You need to use the value before the colon in the output, e.g. `0`.

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
```


6.5.3 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

Todo: Unknown

Example

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

6.5.4 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

Todo: Unknown

Example

```
resolutions = [  
    352x240,  
    480x360,  
    858x480,  
    1280x720,  
    1920x1080,  
    2560x1080,  
    3440x1440,  
    1920x1200,  
    3860x2160,  
    3840x1600,  
]
```

6.5.5 dwmflush

Description

Invoke DwmFlush() to sync screen capture to the Windows presentation interval.

Caution: Applies to Windows only. Alleviates visual stuttering during mouse movement. If enabled, this feature will automatically deactivate if the client framerate exceeds the host monitor's current refresh rate.

Note: If you disable this option, you may see video stuttering during mouse movement in certain scenarios. It is recommended to leave enabled when possible.

Default

enabled

Example

```
dwmflush = enabled
```

6.6 Audio

6.6.1 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
```

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 = {0.0.0.000000000}. {FD47D9CC-4218-4135-9CE2-0C195C87405B}
```

6.6.2 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](#)!

Default

Todo: Unknown

Example

```
virtual_sink = {0.0.0.000000000}. {8edba70c-1125-467c-b89c-15da389bc1d4}
```

6.7 Network

6.7.1 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
```

6.7.2 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 are only allowed on select Moonlight clients.

Default

47989

Example

```
port = 47989
```

6.7.3 pkey

Description

The private key. This must be 2048 bits.

Default

Todo: Unknown

Example

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

6.7.4 cert

Description

The certificate. Must be signed with a 2048 bit key.

Default

Todo: Unknown

Example

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

6.7.5 origin_pin_allowed

Description

The origin of the remote endpoint address that is not denied for HTTP method /pin.

Choices

Value	Description
pc	Only localhost may access /pin
lan	Only LAN devices may access /pin
wan	Anyone may access /pin

Default

pc

Example

```
origin_pin_allowed = pc
```

6.7.6 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
```

6.7.7 upnp

Description

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

Choices

Value	Description
on	enable UPnP
off	disable UPnP

Default

off

Example

```
upnp = on
```

6.7.8 ping_timeout

Description

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

Default

10000

Example

```
ping_timeout = 10000
```

6.8 Encoding

6.8.1 channels

Description

This will generate distinct video streams, unlike simply broadcasting to multiple Clients.

When multicasting, it could be useful to have different configurations for each connected Client.

For instance:

- Clients connected through WAN and LAN have different bitrate constraints.
- Decoders may require different settings for color.

Warning: CPU usage increases for each distinct video stream generated.

Default

1

Example

```
channels = 1
```

6.8.2 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
```

6.8.3 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
```

6.8.4 min_threads

Description

Minimum number of threads used for software 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

1

Example

```
min_threads = 1
```

6.8.5 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
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
```

6.8.6 encoder

Description

Force a specific encoder.

Choices

Value	Description
nvenc	For Nvidia 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
```


6.8.7 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	
superfast	
faster	
fast	
medium	
slow	
slow	
slower	
veryslow	slowest

Default

superfast

Example

```
sw_preset = superfast
```

6.8.8 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
```

6.8.9 nv_preset

Description

The encoder preset to use.

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

Choices

Value	Description
p1	fastest (lowest quality)
p2	faster (lower quality)
p3	fast (low quality)
p4	medium (default)
p5	slow (good quality)
p6	slower (better quality)
p7	slowest (best quality)

Default

p4

Example

```
nv_preset = p4
```

6.8.10 nv_tune

Description

The encoder tuning profile.

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

Choices

Value	Description
hq	high quality
ll	low latency
ull	ultra low latency
lossless	lossless

Default

ull

Example

```
nv_tune = ull
```

6.8.11 nv_rc

Description

The encoder rate control.

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

Choices

Value	Description
constqp	constant QP mode
vbr	variable bitrate
cbr	constant bitrate

Default

cbr

Example

```
nv_rc = cbr
```

6.8.12 nv_coder

Description

The entropy encoding to use.

Note: This option only applies when using H264 with nvenc *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

```
nv_coder = auto
```

6.8.13 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
```

6.8.14 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
```

6.8.15 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
```

6.8.16 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
```

6.8.17 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
```

6.8.18 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
```

6.9 Advanced

6.9.1 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
```

6.9.2 file_state

Description

The file where current state of Sunshine is stored.

Default

sunshine_state.json

Example

```
file_state = sunshine_state.json
```

6.9.3 credentials_file

Description

The file where user credentials for the UI are stored.

Default

sunshine_state.json

Example

```
credentials_file = sunshine_state.json
```


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. At the time of writing this GSMS offers the ability to migrate your custom 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 Limitations

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

- HDR support is limited and currently HDR is converted to SDR.
- Automatic game/application list.
- Changing game settings automatically, to optimize streaming.

GENERAL

8.1 Forgotten Credentials

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

```
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 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))
```


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 [ViGEmBus](#).

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

End of Life: TBD

Install Requirements

```
sudo apt update && sudo apt install \  
    build-essential \  
    cmake \  
    libavdevice-dev \  
    libboost-filesystem-dev \  
    libboost-log-dev \  
    libboost-thread-dev \  
    libboost-program-options-dev \  
    libcap-dev \ # KMS  
    libdrm-dev \ # KMS  
    libevdev-dev \  
    libnuma-dev \  
    libopus-dev \  
    libpulse-dev \  
    libssl-dev \  
    libva-dev \  
    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 \  
    nvidia-cuda-dev \ # Cuda, NvFBC  
    nvidia-cuda-toolkit \ # Cuda, NvFBC
```

13.1.2 Fedora 36

End of Life: TBD

Install Repositories

```
sudo dnf update && \
    sudo dnf group install "Development Tools" && \
    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
```

Install Requirements

```
sudo dnf install \
    boost-devel \
    boost-static.x86_64 \
    cmake \
    gcc-c++ \
    libevdev-devel \
    libva-devel \
    libvdpau-devel \
    libX11-devel \ # X11
    libxcb-devel \ # X11
    libXcursor-devel \ # X11
    libXfixes-devel \ # X11
    libXinerama-devel \ # X11
    libXi-devel \ # X11
    libXrandr-devel \ # X11
    libXtst-devel \ # X11
    mesa-libGL-devel \
    nodejs \
    npm \
    numactl-devel \
    openssl-devel \
    opus-devel \
    pulseaudio-libs-devel \
    rpm-build # if you want to build an RPM binary package
```

13.1.3 Ubuntu 20.04

End of Life: April 2030

Install Requirements

```
sudo apt update && sudo apt install \
    build-essential \
    cmake \
    g++-10 \
    libavdevice-dev \
    libboost-filesystem-dev \
    libboost-log-dev \
    libboost-thread-dev \
```

(continues on next page)

(continued from previous page)

```

libboost-program-options-dev \
libcap-dev \ # KMS
libdrm-dev \ # KMS
libevdev-dev \
libnuma-dev \
libopus-dev \
libpulse-dev \
libssl-dev \
libva-dev \
libvdpau-dev \
libwayland-dev \ # Wayland
libx11-dev \ # X11
libxcb-shm0-dev \ # X11
libxcb-xfixes0-dev \ # X11
libxcb1-dev \ # X11
libxf86-dev \ # X11
libxrandr-dev \ # X11
libxtst-dev \ # X11
nodejs \
npm \
wget

```

Update gcc alias

```

update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 --slave /usr/bin/
↪ g++ g++ /usr/bin/g++-10

```

Install CuDA

```

wget https://developer.download.nvidia.com/compute/cuda/11.4.2/local_installers/
↪ cuda_11.4.2_470.57.02_linux.run --progress=bar:force:noscroll -q --show-progress -
↪ 0 ./cuda.run && chmod a+x ./cuda.run
./cuda.run --silent --toolkit --toolkitpath=/usr --no-opengl-libs --no-man-page --
↪ no-drm && rm ./cuda.run

```

13.1.4 Ubuntu 22.04

End of Life: April 2027

Install Requirements

```

sudo apt update && sudo apt install \
build-essential \
cmake \
libavdevice-dev \
libboost-filesystem-dev \
libboost-log-dev \
libboost-thread-dev \
libboost-program-options-dev \
libcap-dev \ # KMS
libdrm-dev \ # KMS

```

(continues on next page)

(continued from previous page)

```
libevdev-dev \
libnuma-dev \
libopus-dev \
libpulse-dev \
libssl-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 \
nvidia-cuda-dev \ # Cuda, NvFBC
nvidia-cuda-toolkit # Cuda, NvFBC
```

13.2 npm dependencies

Install npm dependencies.

```
npm install
```

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 npm9 pkgconfig
```

14.1.2 Homebrew

Install Requirements

```
brew install boost cmake node opus  
# if there are issues with an SSL header that is not found:  
cd /usr/local/include  
ln -s ../opt/openssl/include/openssl .
```

14.2 npm dependencies

Install npm dependencies.

```
npm install
```

14.3 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-openssl mingw-w64-x86_64-opus mingw-w64-x86_64-toolchain
```

15.2 npm dependencies

Install nodejs and npm. Downloads available [here](#).

Install npm dependencies.

```
npm install
```

15.3 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)

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

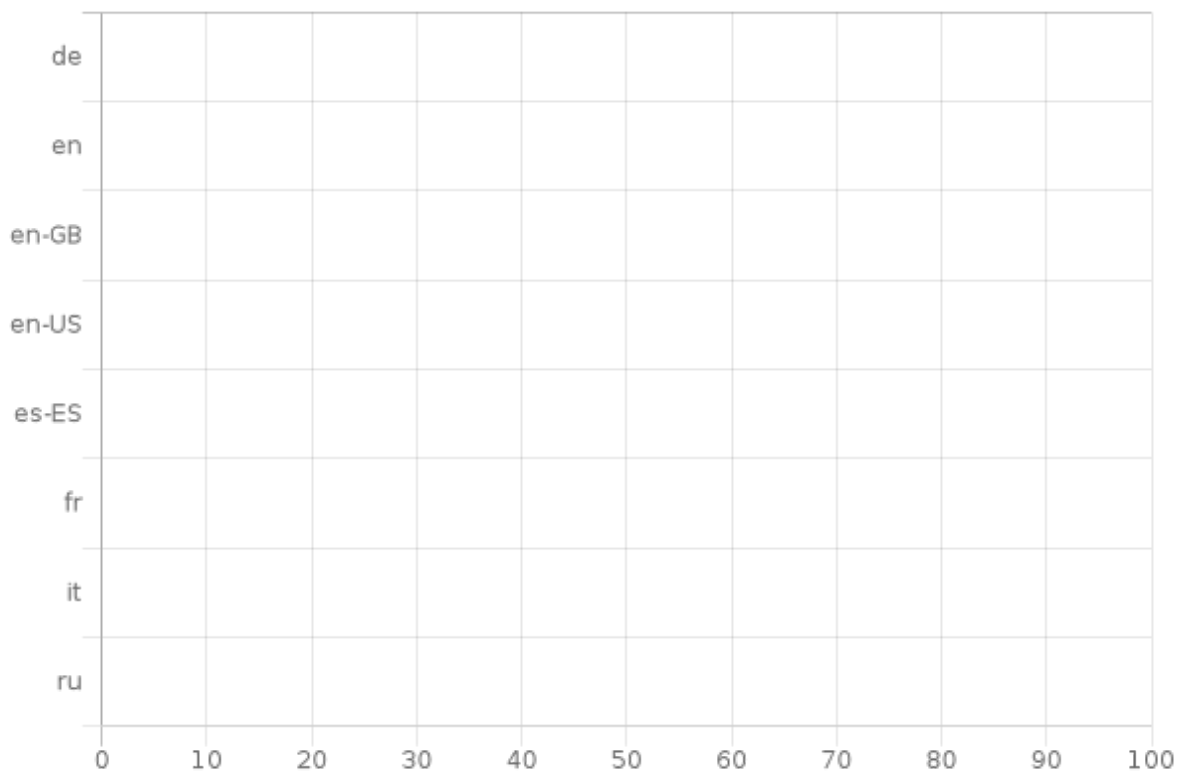

CONTRIBUTING

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

LOCALIZATION

Sunshine is being localized into various languages. The default language is *en* (English) and is highlighted green.

Graph



17.1 CrowdIn

The translations occur on [CrowdIn](#). Feel free to contribute to localization there. Only elements of the API are planned to be translated.

Attention: The rest API has not yet been implemented.

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>
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 documentation 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.

The config file for Sphinx is *docs/source/conf.py*. 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
```

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.