Sunshine

ReenigneArcher

Aug 09, 2022

ABOUT

CHAPTER ONE

OVERVIEW

LizardByte has the full documentation hosted on Read the Docs.

1.1 About

Sunshine is a Game stream host for Moonlight. Sunshine is a self hosted, low latency, cloud gaming solution with support for AMD, Intel, and Nvidia gpus. It is an open source implementation of NVIDIA's GameStream, as used by the NVIDIA Shield. Connect to Sunshine from any Moonlight client, available for nearly any device imaginable.

These are the advantages of Sunshine over GeForce Experience.

- FOSS (Free and Open Source Software)
- Multi-platform
  - Linux
  - macOS
  - Windows
- Pair over web ui
- Supports AMD, Intel, and Nvidia GPUs for encoding
- Supports software encoding
- Supports streaming to multiple clients
- Web UI for configuration

1.2 Integrations

1.3 Support

Our support methods are listed in our LizardByte Docs.

1.4 Downloads

CHAPTER

TWO

INSTALLATION

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, and Windows. Binaries can be found in the latest release.

Tip: Some third party packages also exist. See *Third Party Packages*.

2.2 Docker

Todo: Docker images of Sunshine are planned to be included in the future. They will be available on Dockerhub.io and ghcr.io.

2.3 Linux

Follow the instructions for your preferred package type below.

2.3.1 AppImage

According to AppImageLint the AppImage can run on the following distros.

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

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

1. Download `sunshine-appimage.zip` and extract the contents to your home directory.

2.3.2 AUR Package

1. Open terminal and run the following code.

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

2.3.3 Debian Package

1. Download `sunshine.deb` and run the following code.

   ```
   sudo apt install -f ./sunshine.deb
   ```

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

2.3.4 Flatpak Package

Todo: This package needs to have CUDA added.

1. Install Flatpak as required.
2. Download `sunshine.flatpak` and run the following code.

   System level (recommended)

   ```
   flatpak install --system sunshine.flatpak
   ```

   User level

   ```
   flatpak install --user sunshine.flatpak
   ```

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.

2.4 macOS

pkg

Warning: The *pkg* does not include runtime dependencies and should be considered experimental.

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

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 ~/Downlaods/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.

2.5 Windows

Installed option:

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

Standalone option:

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

Todo: This is a planned feature. Currently no Dockerfile or image exists for Sunshine.

CHAPTER

THREE

DOCKER

3.1 Using docker run

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

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

To update the container it must be removed and recreated:

```
# Stop the container
docker stop sunshine
# Remove the container
docker rm sunshine
# Pull the latest update
docker pull lizardbyte/sunshine
# Run the container with the same parameters as before
docker run -d ...
```

3.2 Using docker-compose

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

```
version: '3'
services:
  sunshine:
    image: lizardbyte/sunshine
    container_name: sunshine
    restart: unless-stopped
    volumes:
```

(continues on next page)

(continued from previous page)

```
      - <path to data>:/config
    environment:
      - PUID=<uid>
      - PGID=<gid>
      - TZ=<timezone>
    ports:
      - 47984-47990:47984-47990/tcp
      - 48010:48010
      - 47998-48000:47998-48000/udp
```

Create and start the container (run the command from the same folder as your `docker-compose.yml` file):

```
docker-compose up -d
```

To update the container:

```
# Pull the latest update
docker-compose pull
# Update and restart the container
docker-compose up -d
```

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

3.3.1 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)
```

CHAPTER

FOUR

THIRD PARTY PACKAGES

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

4.1 Chocolatey

4.2 Scoop

4.3 Winget

4.4 Legacy GitHub Repo

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

CHAPTER

FIVE

USAGE

1. See the *setup* section for your specific OS.
2. Run `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.

   Tip: If using the Linux AppImage, replace `sunshine` with `./sunshine.AppImage`

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".

   Caution: If running for the first time, make sure to note the username and password Sunshine showed to you, since you cannot get back later!

   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.

   Attention: Application list is not fully supported on macOS

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

Sunshine will be available on port 47990 by default.

Danger: Do not expose port 47990, or the web ui, to the internet!

5.2 Arguments

To get a list of available arguments run the following:

```
sunshine --help
```

5.3 Setup

5.3.1 Linux

The deb and rpm packages handle these steps automatically. The AppImage does not, third party packages may not as well.

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.**

   ```
   sudo nano /etc/udev/rules.d/85-sunshine.rules
   ```

   Input the following contents.

   ```
   KERNEL=="uinput", GROUP="input", MODE="0660", OPTIONS+="static_node=uinput"
   ```

   Save the file and exit:

   1. `CTRL+X` to start exit.
   2. `Y` to save modifications.

3. **Optionally, configure autostart service**

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

```
[Unit]
Description=Sunshine Gamestream Server for Moonlight

[Service]
ExecStart=<see table>

[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

```
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

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

- 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 be replaced by `$(HOME)`
- `env` - Adds or overwrites Environment variables for the commands/applications run by Sunshine
- `"Variable name":"Variable value"`
- `apps` - The list of applications
- Example application:

```
{
"name":"An App",
"cmd":"command to open app",
"prep-cmd":[
          {
                  "do":"some-command",
                  "undo":"undo-that-command"
          }
  ],
"detached":[
  "some-command",
  "another-command"
  ]
}
```

  – `name` - The name of the application/game

    - `output` - The file where the output of the command is stored
    - `detached` - A list of commands to be run and forgotten about
    - `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
            · This should not fail considering it is supposed to undo the `do` commands
            · If it fails, Sunshine is terminated
        * `cmd` - The main application
            · If not specified, a process is started that sleeps indefinitely

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.
- 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.

CHAPTER

SIX

ADVANCED USAGE

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

6.1 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-aur	/usr/share/sunshine/config/
Linux-deb	/usr/local/sunshine/config/
Linux-rpm	/usr/local/sunshine/config/
macOS	/usr/local/sunshine/config/
Windows	./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.

6.2 General

6.2.1 sunshine_name

Description
The name displayed by Moonlight

Default
PC hostname

Example

```
sunshine_name = Sunshine
```

6.2.2 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 Controls

6.3.1 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
```

6.3.2 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
```

6.3.3 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
```

6.3.4 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
```

6.3.5 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
]
```

6.3.6 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
```

6.4 Display

6.4.1 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
```

(continues on next page)

(continued from previous page)

```
# 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
```

Todo: macOS

Windows

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

6.4.2 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
```

6.4.3 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]
```

6.4.4 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,
]
```

6.4.5 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.

Default

`enabled`

Examples

Windows

```
dwmflush = enabled
```

6.5 Audio

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␣
```

(continues on next page)

(continued from previous page)

```
↪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 = {0.0.0.00000000}.{FD47D9CC-4218-4135-9CE2-0C195C87405B}
```

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*!

Default

Todo: Unknown

Example

```
virtual_sink = {0.0.0.00000000}.{8edba70c-1125-467c-b89c-15da389bc1d4}
```

6.6 Network

6.6.1 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
```

6.6.2 port

Description
Set the family of ports used by Sunshine.

Default
`47989`

Example

```
port = 47989
```

6.6.3 pkey

Description
The private key. This must be 2048 bits.

Default

Todo: Unknown

Example

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

6.6.4 cert

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

Default

Todo: Unknown

Example

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

6.6.5 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
```

6.6.6 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
```

6.6.7 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
```

6.6.8 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 Encoding

6.7.1 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
```

6.7.2 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
```

6.7.3 qp

Description

Quantitization 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
```

6.7.4 min_threads

Description

Minimum number of threads used by ffmpeg to encode the video.

> **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
```

6.7.5 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
```

6.7.6 encoder

Description
Force a specific encoder.

Choices

Value	Description
nvenc	For Nvidia graphics cards
amdvce	For AMD graphics cards
software	Encoding occurs on the CPU

Default
Sunshine will use the first encoder that is available.

Example

```
encoder = nvenc
```

6.7.7 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
```

6.7.8 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
```

6.7.9 nv_preset

Description
The encoder preset to use.

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

Choices

Value	Description
default	let ffmpeg decide
hp	high performance
hq	high quality
slow	high quality, 2 passes
medium	high quality, 1 pass
fast	high performance, 1 pass
bd	
ll	low latency
llhq	low latency, high quality
llhp	low latency, high performance
lossless	lossless
losslesshp	lossless, high performance

Default
`llhq`

Example

```
nv_preset = llhq
```

6.7.10 nv_rc

Description

The encoder rate control.

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

Note: Moonlight does not currently support variable bitrate, although it can still be selected here.

Choices

Value	Description
auto	let ffmpeg decide
constqp	constant QP mode
cbr	constant bitrate
cbr_hq	constant bitrate, high quality
cbr_ld_hq	constant bitrate, low delay, high quality
vbr	variable bitrate
vbr_hq	variable bitrate, high quality

Default

`auto`

Example

```
nv_rc = auto
```

6.7.11 nv_coder

Description

The entropy encoding to use.

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

Choices

Value	Description
auto	let ffmpeg decide
cabac	
cavlc	

Default

`auto`

Example

```
nv_coder = auto
```

6.7.12 amd_quality

Description

The encoder preset to use.

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

Choices

Value	Description
default	let ffmpeg decide
speed	fast
balanced	balance performance and speed

Default

`balanced`

Example

```
amd_quality = balanced
```

6.7.13 amd_rc

Description

The encoder rate control.

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

Note: Moonlight does not currently support variable bitrate, although it can still be selected here.

Choices

Value	Description
auto	let ffmpeg decide
constqp	constant QP mode
cbr	constant bitrate
vbr_latency	variable bitrate, latency constrained
vbr_peak	variable bitrate, peak constrained

Default

`auto`

Example

```
amd_rc = auto
```

6.7.14 amd_coder

Description

The entropy encoding to use.

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

Choices

Value	Description
auto	let ffmpeg decide
cabac	
cavlc	

Default

`auto`

Example

```
amd_coder = auto
```

6.7.15 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
```

6.7.16 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
```

6.7.17 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
```

6.8 Advanced

6.8.1 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
```

6.8.2 file_state

Description
The file where current state of Sunshine is stored.

Default
`sunshine_state.json`

Example

```
file_state = sunshine_state.json
```

6.8.3 credentials_file

Description
The file where user credentials for the UI are stored.

Default
`sunshine_state.json`

Example

```
credentials_file = sunshine_state.json
```

CHAPTER SEVEN

GENERAL

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

```
sunshine -creds <new username> <new password>
```

Can't access the web UI?

1. Check firefall rules.

CHAPTER EIGHT

LINUX

If screencasting fails with Wayland, you may need to run the following to force screencasting with X11.

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

CHAPTER

NINE

MACOS

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
> ```

Uninstall:

- pkg

  ```
  sudo chmod +x /opt/local/etc/sunshine/assets/uninstall_pkg.sh
  sudo /opt/local/etc/sunshine/assets/uninstall_pkg.sh
  ```

- Portfile

  ```
  sudo port uninstall Sunshine
  ```

CHAPTER

TEN

WINDOWS

No gamepad is detected.

1. Verify that you’ve installed ViGEmBus.

CHAPTER ELEVEN

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.

11.1 Building Locally

11.1.1 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
```

11.1.2 Compile

See the section specific to your OS.

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

11.2 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

CHAPTER

TWELVE

LINUX

12.1 Requirements

Danger: Installing these dependencies may break your distribution. It is recommended to build in a virtual machine or to use the *Dockerfile builds* located in the *./scripts* directory.

12.1.1 Debian Bullseye

End of Life: TBD

Install Requirements

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

12.1.2 Fedora 35

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 \
    ffmpeg-devel \
    gcc-c++ \
    libevdev-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 \
    openssl-devel \
    opus-devel \
    pulseaudio-libs-devel \
    rpm-build \  # if you want to build an RPM binary package
```

12.1.3 Ubuntu 18.04

End of Life: April 2028

Install Repositories

```
sudo apt update && sudo apt install \
    software-properties-common \
&& add-apt-repository ppa:savoury1/graphics && \
add-apt-repository ppa:savoury1/multimedia && \
add-apt-repository ppa:savoury1/ffmpeg4 && \
add-apt-repository ppa:savoury1/boost-defaults-1.71 && \
add-apt-repository ppa:ubuntu-toolchain-r/test && \
```

Install Requirements

```
sudo apt install \
    build-essential \
    cmake \
```

(continues on next page)

(continued from previous page)

```
    gcc-10 \
    git \
    g++-10 \
    libavdevice-dev \
    libboost-filesystem1.71-dev \
    libboost-log1.71-dev \
    libboost-regex1.71-dev \
    libboost-thread1.71-dev \
    libcap-dev \  # KMS
    libdrm-dev \  # KMS
    libevdev-dev \
    libpulse-dev \
    libopus-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
    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 -
↪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
```

Install CMake

```
wget https://cmake.org/files/v3.22/cmake-3.22.2-linux-x86_64.sh
mkdir /opt/cmake
sh /cmake-3.22.2-linux-x86_64.sh --prefix=/opt/cmake --skip-license
ln -s /opt/cmake/bin/cmake /usr/local/bin/cmake
cmake --version
```

12.1.4 Ubuntu 20.04

End of Life: April 2030

Install Requirements

```
sudo apt update && sudo apt install \
    build-essential \
    cmake \
    git \
    g++-10 \
    libavdevice-dev \
    libboost-filesystem-dev \
    libboost-log-dev \
    libboost-thread-dev \
    libcap-dev \  # KMS
    libdrm-dev \  # KMS
    libevdev-dev \
    libpulse-dev \
    libopus-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
    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 -
↪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
```

12.1.5 Ubuntu 21.10

End of Life: July 2022

Install Requirements

```
sudo apt update && sudo apt install \
    build-essential \
    cmake \
    git \
```

(continues on next page)

(continued from previous page)

```
    libavdevice-dev \
    libboost-filesystem-dev \
    libboost-log-dev \
    libboost-thread-dev \
    libcap-dev \  # KMS
    libdrm-dev \  # KMS
    libevdev-dev \
    libpulse-dev \
    libopus-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
    nvidia-cuda-dev \  # Cuda, NvFBC
    nvidia-cuda-toolkit \  # Cuda, NvFBC
```

12.1.6 Ubuntu 22.04

End of Life: April 2027

Todo: Create Ubuntu 22.04 Dockerfile and complete this documentation.

12.2 Build

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

Debian based OSes

```
cmake -DCMAKE_C_COMPILER=gcc-10 -DCMAKE_CXX_COMPILER=g++-10 ..
```

Red Hat based Oses

```
cmake -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++ ..
```

Finally

```
make -j ${nproc}
cpack -G DEB  # optionally, create a deb package
cpack -G RPM  # optionally, create a rpm package
```

12.3 Dockerfile Builds

You may wish to simply build sunshine from source, without bloating your OS with development files. There are scripts located in the `./scripts` directory that will create docker images that have the necessary packages. As a result, removing the development files after you're done is a single command away. These scripts use docker under the hood, as such, they can only be used to compile the Linux version

Todo: Publish the Dockerfiles to Dockerhub and ghcr.

Requirements
Install Docker

Instructions

1. *Clone*. Sunshine.
2. Select the desired Dockerfile from the `./scripts` directory.

   Available Files:

   ```
   Dockerfile-debian
   Dockerfile-fedora_33  # end of life
   Dockerfile-fedora_35
   Dockerfile-ubuntu_18_04
   Dockerfile-ubuntu_20_04
   Dockerfile-ubuntu_21_04  # end of life
   Dockerfile-ubuntu_21_10
   ```

3. Execute

   ```
   cd scripts  # move to the scripts directory
   ./build-container.sh -f Dockerfile-<name>  # create the container (replace the "
   ↪<name>")
   ./build-sunshine.sh -p -s ..  # compile and build sunshine
   ```

4. Updating

   ```
   git pull  # pull the latest changes from github
   ./build-sunshine.sh -p -s ..  # compile and build sunshine
   ```

5. Optionally, delete the container .. code-block:: bash

   ./build-container.sh -c delete

6. Install the resulting package

   Debian

   ```
   sudo apt install -f sunshine-build/sunshine.deb
   ```

   Red Hat

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

CHAPTER

THIRTEEN

MACOS

13.1 Requirements

macOS Big Sur and Xcode 12.5+

Use either MacPorts or Homebrew

13.1.1 MacPorts

Install Requirements

```
sudo port install cmake boost ffmpeg libopus
```

13.1.2 Homebrew

Install Requirements

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

13.2 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/1.76 ..`

CHAPTER FOURTEEN

WINDOWS

14.1 Requirements

First you need to install MSYS2, then startup "MSYS2 MinGW 64-bit" and install the following packages using:

```
pacman -S mingw-w64-x86_64-binutils mingw-w64-x86_64-openssl mingw-w64-x86_64-cmake␣
↪mingw-w64-x86_64-toolchain mingw-w64-x86_64-opus mingw-w64-x86_64-x265 mingw-w64-x86_
↪64-boost git mingw-w64-x86_64-make cmake make gcc
```

14.2 Build

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

```
cmake -G"Unix Makefiles" ..
cmake -G"MinGW Makefiles" ..  # alternatively

mingw32-make

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

CHAPTER

FIFTEEN

CONTRIBUTING

Tip: If this is your first time contributing to an open source project, it is a good idea to read MDN's Basic etiquette for open source projects first. There are a few best practices to adopt that will help ensure that you and the other project contributors feel valued and safe, and stay productive.

1. Fork the repo on GitHub
2. Create a new branch for the feature you are adding or the issue you are fixing

   Tip: Base the new branch off the *nightly* branch. It will make your life easier when you submit the PR!

3. Make changes, push commits, etc.
4. Files should contain an empty line at the end.
5. Document your code!
6. Test your code!
7. When ready create a PR to this repo on the *nightly* branch.

   Hint: If you accidentally make your PR against a different branch, a bot will comment letting you know it's on the wrong branch. Don't worry. You can edit the PR to change the target branch. There is no reason to close the PR!

   Note: Draft PRs are also welcome as you work through issues. The benefit of creating a draft PR is that an automated build can run in a github runner.

   Attention: Do not expect partially complete PRs to be merged. These topics will be considered before merging.

   - Does the code follows the style guidelines of this project?

     Tip: Look at examples of existing code in the project!

   - Is the code well commented?
   - Were documentation blocks updated for new or modified components?

Note: Developers and maintainers will attempt to assist with challenging issues.

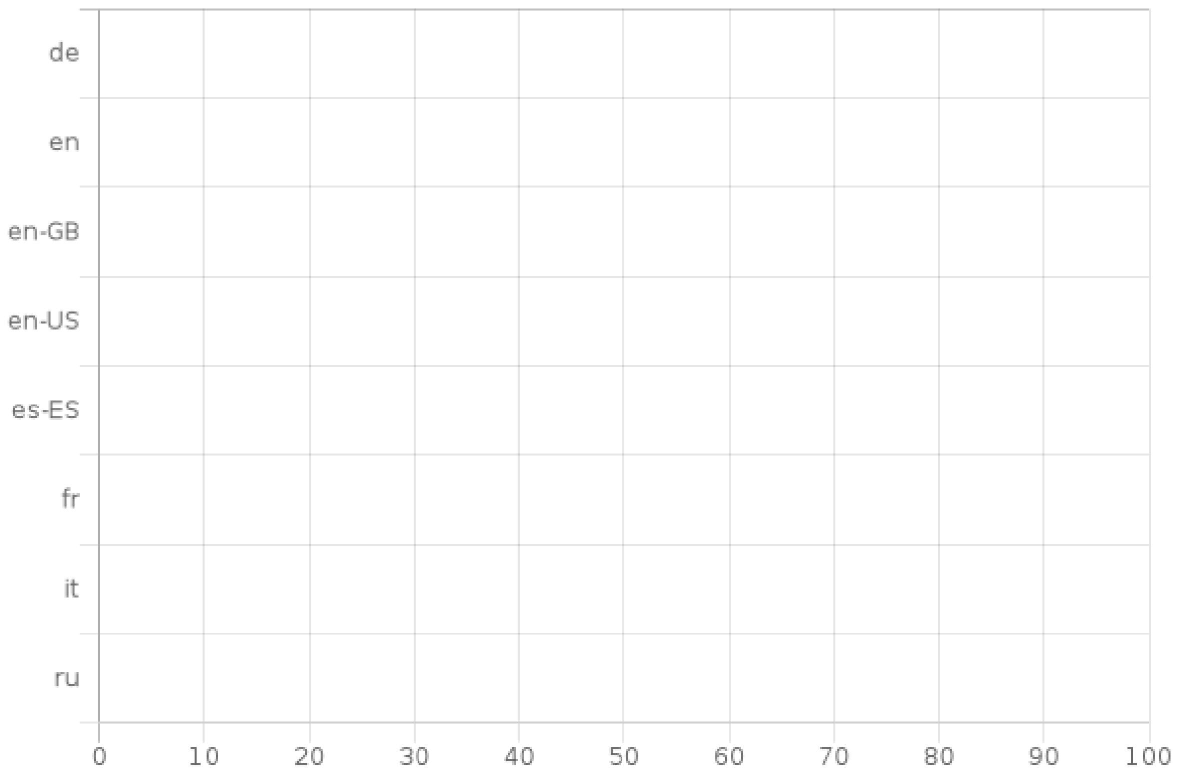
CHAPTER

SIXTEEN

LOCALIZATION

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

Graph



16.1 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.

16.2 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
```

CHAPTER

SEVENTEEN

TESTING

17.1 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/clang.yml*.

Test clang-format locally.

Todo: This documentation needs to be improved.

```
clang-format ...
```

17.2 Sphinx

Sunshine uses Sphinx for documentation building. Sphinx is included in the *./scripts/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
```

17.3 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.