RaySession Manual

Table of Contents

for version 0.13.1

Introduction

RaySession is an audio session manager for GNU / Linux. It allows you to start several audio programs in the same session, to save their projects together and thus to avoid multiple operations to return to a given configuration.

To be launched in RaySession, it is very much preferable that these audio programs be compatible with the NSM protocol, and many are already, among others Ardour, Qtractor, Carla, Guitarix, Mamba, Patroneo, ZynAddSubFx …

RaySession assumes that your audio setup is working for audio production, if not, fix this first and don't waste your time trying to use RaySession.

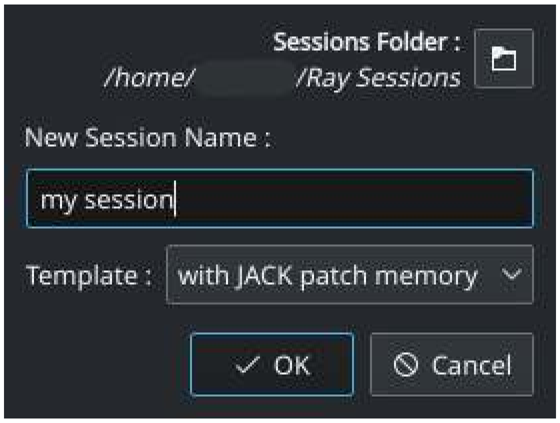
Now let's see how to use it.

Simple use example

For audio production under GNU / linux, it is highly recommended to use the JACK server. Before creating or starting a session, make sure the JACK server is running, RaySession has no direct relation to JACK, however the programs you are going to launch in your session will need it.

Let's take an example where to compose a song, we will need Ardour and Guitarix softwares, make sure that these 2 programs are installed.

To create a new session, click at the top left on New Session (*or Ctrl+N*). A dialog window appears.



Sessions Folder :
/home/[redacted]/Ray Sessions

New Session Name :
my session

Template : with JACK patch memory

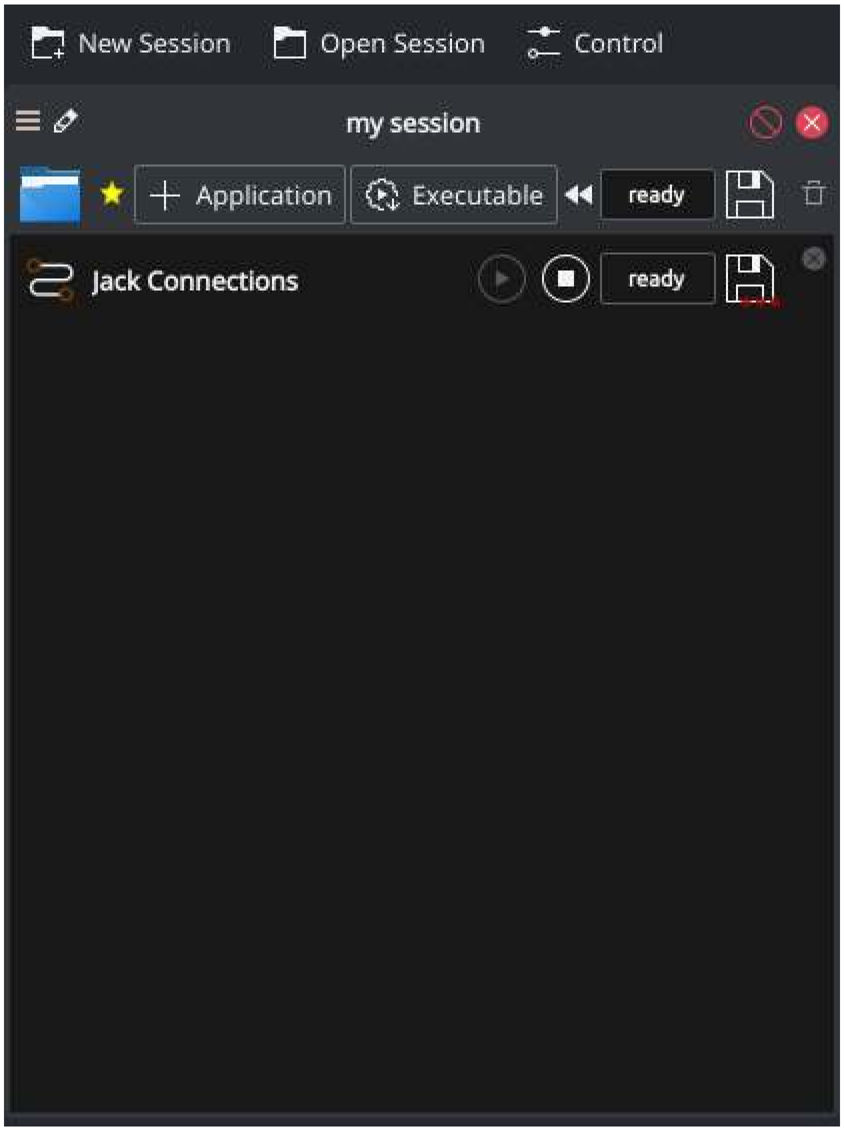
At the top right you can see the root session folder, this is where the sessions will be saved (a RaySession session being a folder containing a file `raysession.xml`).

Enter the name you want for this new session in the field provided. To put your new session in a sub-folder, type the name as follows: `sub-folder/my session` .

The session template multiple choice box lets you choose between

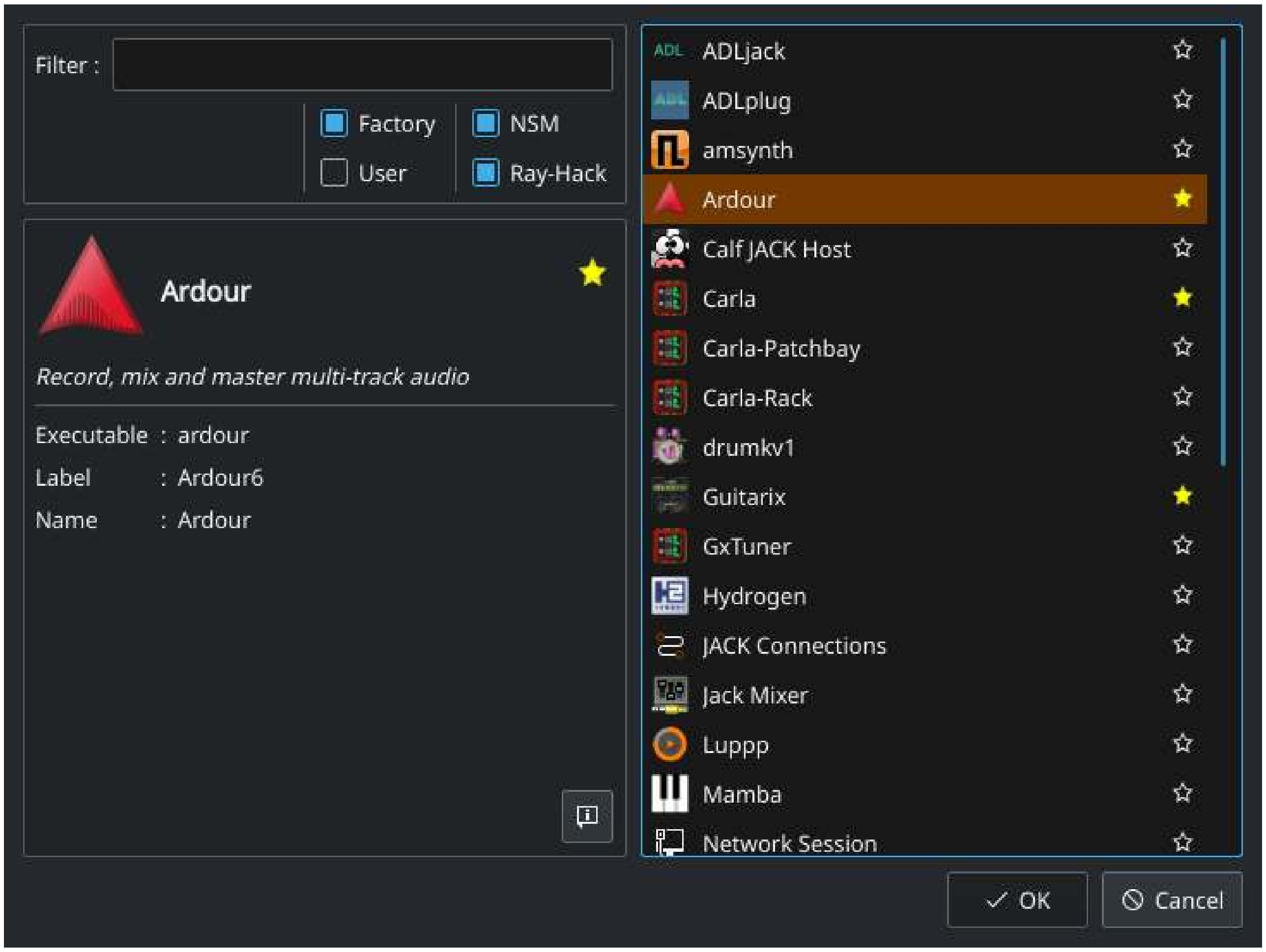
- An empty session template
- A template with memory of JACK connections
- A scripted template with memory of the JACK configuration
- A template with basic session scripts (for advanced users with shell scripting knowledge)
- All the session templates that you have created yourself.

First, keep the template on **With JACK patch memory** , click **Ok** to start your session.



The session part of the window becomes active, at the top is written the name of your session, and your session already contains a client program called **JACK Connections**. This client will save and redo the connections of the JACK patchbay.

To add Ardour to your session, click on the **Application** button (*or Ctrl+A*), the application dialog will appear (more details in the Add application window section). On the right side, find Ardour (Depending on your version of ardour, it may be called Ardour5 or Ardour6) and double-click on it.


Filter :
[✓ Factory] [✓ NSM]
User [✓ Ray-Hack]

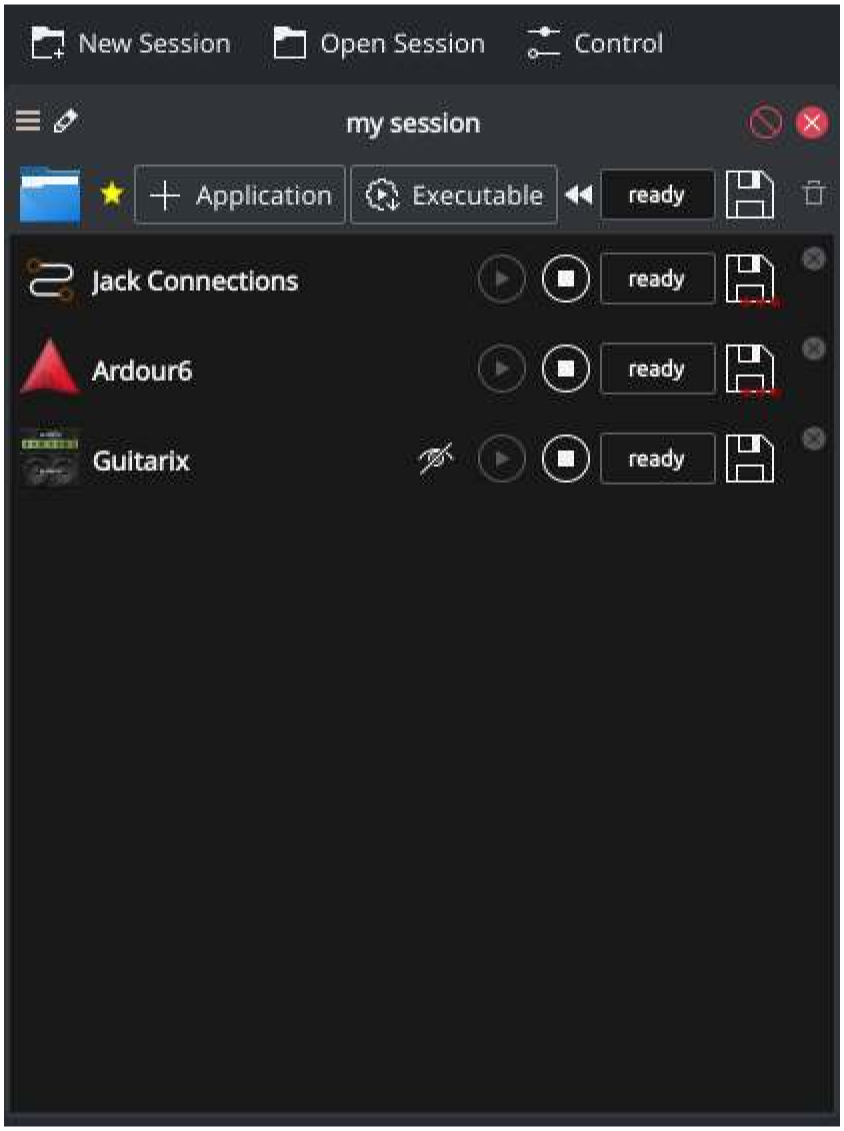
Ardour

Record, mix and master multi-track audio

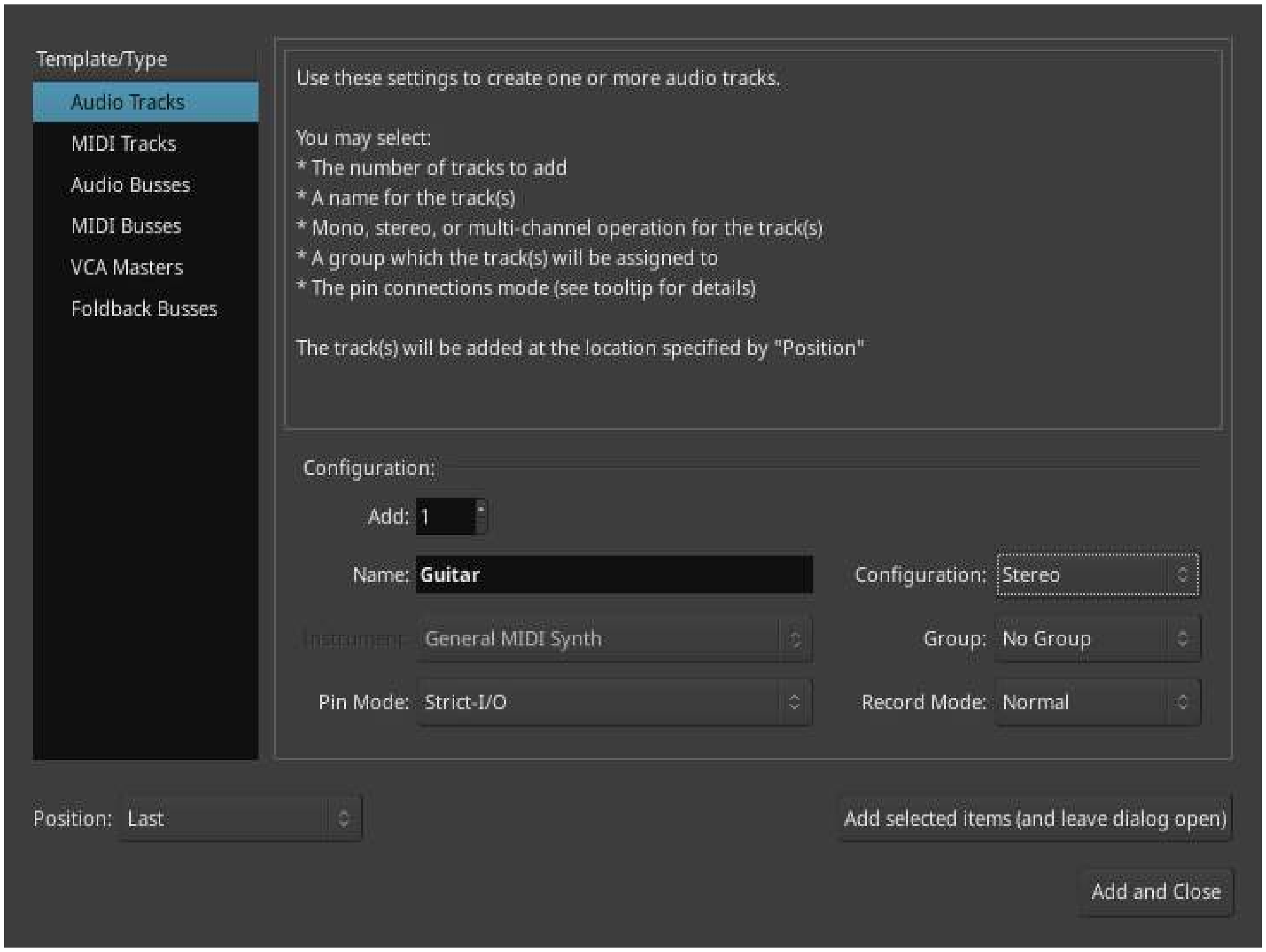
Executable : ardour
Label : Ardour6
Name : Ardour

An Ardour client appears below JACK Connections and Ardour starts up (normally directly with an empty Ardour session).

Do the same with Guitarix.



In Ardour, add a track (Menu: Track → Add a track), in the Ardour window that has appeared, name the track **Guitar** and put the multiple choice box configuration on *Stereo*.



Use these settings to create one or more audio tracks.

You may select:
* The number of tracks to add
* A name for the track(s)
* Mono, stereo, or multi-channel operation for the track(s)
* A group which the track(s) will be assigned to
* The pin connections mode (see tooltip for details)

The track(s) will be added at the location specified by "Position"

Configuration:

Add: 1

Name: Guitar

Configuration: Stereo

Instrument: General MIDI Synth

Group: No Group

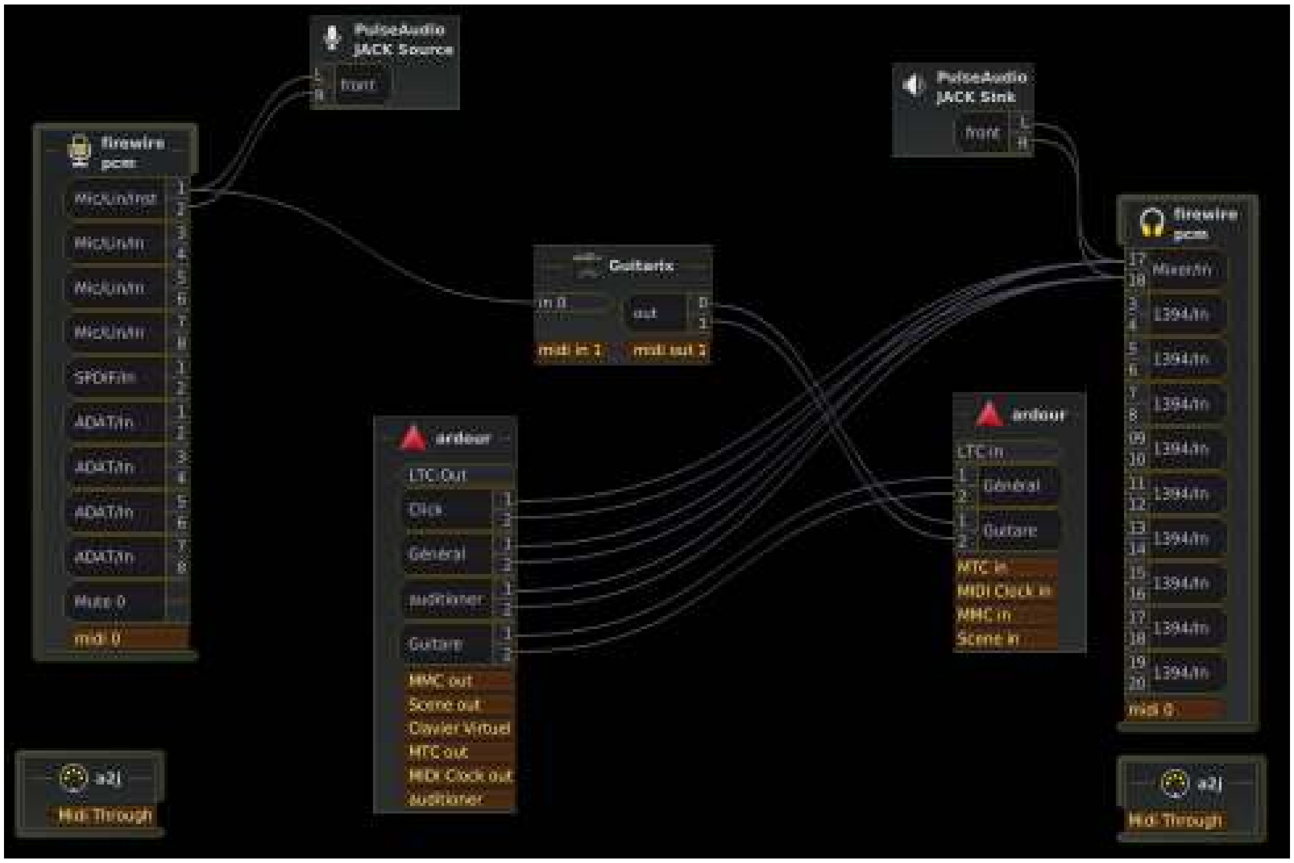
Pin Mode: Strict-I/O

Record Mode: Normal

Position: Last

Now use JACK's Patchbay which occupies the right side of the RaySession window.

Connect the Guitarix input to a hardware input and the Guitarix outputs to the inputs of this new Ardour track. Make sure your Guitar track inputs are not connected to the hardware inputs.



Here you have a configuration where you can directly record the sound of your guitar processed by Guitarix in Ardour. If you don’t have a guitar, all you have to do is sing out of tune into a mic or tap a cushion, this is just an example.

Now save the current session by clicking the floppy disk button to the top right of the session part(*or Ctrl+S*). It is highly recommended because it is very practical to assign a global keyboard shortcut of your system to the save of the current session. This will depend on your desktop environment, but just assign the *Ctrl+Meta+S* shortcut to the command `ray_control save` (Meta is the Windows key), so you won’t have to return to the RaySession window to save the session.

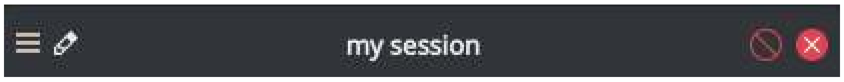
Now close the session by clicking on the red cross at the top right of the session part (*or Ctrl+W*).

Once the session is closed, click on **Open Session** (*or Ctrl+O*), double-click on the session you just created to re-open it.

You will find the programs and their projects as well as the JACK connections as they were when you closed the session, and everything therefore works without any further manipulation. One of the advantages of modularity in this specific case is that once you have finished the guitar records, you can stop Guitarix so as not to overload the processor unnecessarily, and it will still be easy to restart it if necessary.

Overview of session tools

Top row

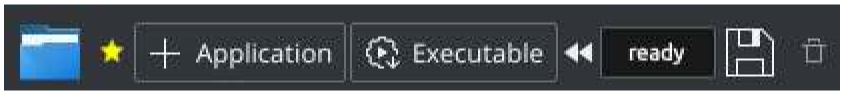


From left to right:

- the menu button gives you access to
  - **save the current session as a template**
    The created session template will then appear in the multiple choice of session templates in the New Session dialog window. Be careful, however, all the files of the session will be saved in the template, so you should not do this if the session contains a lot of audio files. firstly, the copy will be long, secondly, you run the risk of unnecessarily multi-copying files which will take up a lot of space.
  - **Duplicate the current session**
    This is the equivalent of the well-known "Save As ...", except that RaySession has to stop and restart most programs to switch between sessions. Avoid duplicating a session with a lot of audio files, it could take a long time, but fortunately such an operation can be stopped.
  - **Rename the session**
    It will then be necessary to stop all the clients.
    Alternatively, you can rename a session by duplicating it and then deleting the folder from the initial session.
    You can also rename a session by renaming its folder, but BE CAREFUL, this session must not be loaded!
- the pencil-shaped button gives you access to the session notes.
  Write here the information you need, the physical settings, the lyrics of a song, the recipe for granny's cassoulet ... however do not write a novel in 3 volumes, other tools are much more suitable, and notes are limited to 65,000 characters for technical reasons. The pencil is green when notes exist, it is orange when the notes window is open, otherwise it is transparent.
- the name of the loaded session (here **my session**)
- the **Abort session** button which allows you to close the session without saving it

- the **Close session** button ,which saves and closes the current session.
  Note that you do not need to close the current session to start another. Some clients are able to switch from one session to another and it may take a lot less time than closing everything and restarting everything.

Bottom row



From left to right:

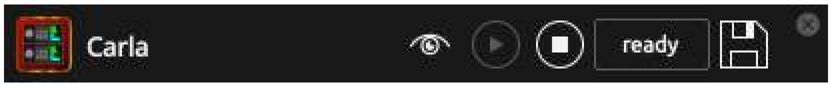
- the folder-shaped button to open the session folder with your file manager
- the yellow star-shaped button that pulls down a menu containing your favorite applications if there are any
- the **Application** button which allows you to add to the session a factory application template or that you have created yourself. This is the recommended method for adding a client. see Add application window.
- the **Executable** button which allows a program to be added to the session from its executable. You will need it if you want to add a program for which there is no template. see Add executable window.
- the reverse button to return to a previous state of the session. This requires having the program `git` installed, else this button will not appears.
  See Snapshots for more details.
- the server status indicator.
  Server states can actually be very stealthy, but they are displayed for a long enough time that you can see them. The server status can be:
  - **off**: no session loaded
  - **ready**: the session is running
  - **launch**: launch of the session's programs
  - **copy**: a copy is in progress, for a session duplication or to save the session as a template
  - **close**: the session is closing

  - **snapshot:** A snapshot of the session is being taken, so you can revert to the current session state.
    see Snapshots.
  - **wait**: The server waits for you to close yourself non-saveable programs
  - **script**: a script is activated

  An information or progress window is displayed if you click this status indicator if it is on **copy** , **snapshot**, or **wait**.

- the **Save Session** button
- the trash, here you will find the clients that you have deleted. You can then restore them in the session or permanently delete all the files they created in the session folder.

Overview of a client



A client contains from left to right:

- The client icon that you can click to bring up a menu with the following actions
  - **Save as application template**
    The created template will then appear in the Add application window. This then allows you to directly launch a client with the desired configuration (Ardour with such tracks, Hydrogen with such drumkit…). Be careful, this copies all the client's files so avoid doing this if the client contains a lot of audio files.
  - **Rename**
    Change the client name located to the right of its icon, it is a purely visual name that can help you organize yourself.
  - **return to a previous state**
    Returns only the client to a previous session state, see Snapshots. However, you will not be able to go back to a state prior to a session renaming, so you must go back the entire session.
  - **Properties**
    Displays the client properties window

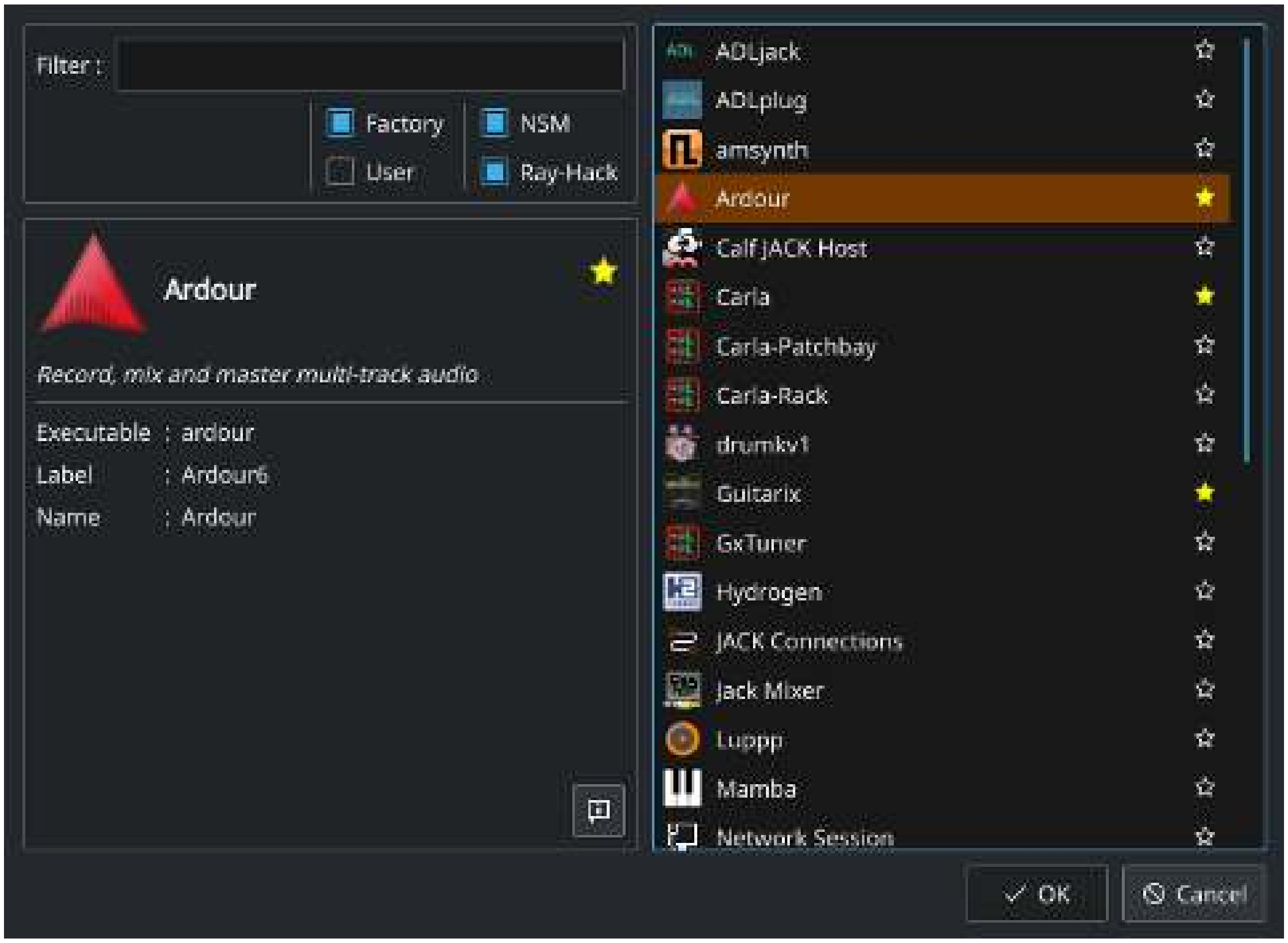
  This menu is also accessible by right-clicking anywhere on the client.

- The name of the client (here Carla), which can be easily changed by right-clicking → Rename
- depending on the type and capacity of the client you can see here
  - an eye (possibly crossed out), this means that the client is NSM compatible and is capable to show or hide its window by clicking on the eye.
  - a **Hack** button, it means that the client is not NSM compatible, or at least that it is not launched with this protocol. Clicking on **Hack** allows to change the way it is launched by opening the client properties window on the Ray-Hack tab.
- The Start button which is grayed out if the client is already started.
- the Stop button which is grayed out if the client is not started.
  If you stop the client and it is still not stopped after a while, the button turns red and you can click on it to kill the client. But stay relaxed, and only use it if it really seems completely inert, it could cause problems, even if nobody will send you to jail.
- the state of the client which can be
  - **stopped**: the client is stopped
  - **ready**: the client is running and everything is ok
  - **open**: the program is opening its project, please wait a little bit.
  - **close**: the program is closing
  - **launch**: if it stays on the launched state, it means
    - if it is a Ray-Hack client, that it does not have a configuration file
    - if it is started as an NSM client, if it is not NSM compatible, and therefore any save is in vain. It may be practical to launch certain programs in this way, such as a patchbay (Catia) or a utility whose state you do not need to save (Qrest).
  - **switch**: the client changes projects during a session switch
- the floppy disk button that allows you to save the client.
  If over this floppy you see
  - three red dots: the client contains unsaved changes

  - a green V: the client does not contain unsaved changes
  - an orange exclamation mark: It is not an NSM client, and it is impossible to save its project, you will have to do it yourself
- the red cross which allows you to send the client to the trash

Add application window

The window for adding application is launched by clicking on the **Application** button (*or Ctrl+A*).


Filter :
[✓ Factory] [✓ NSM] User [✓ Ray-Hack]

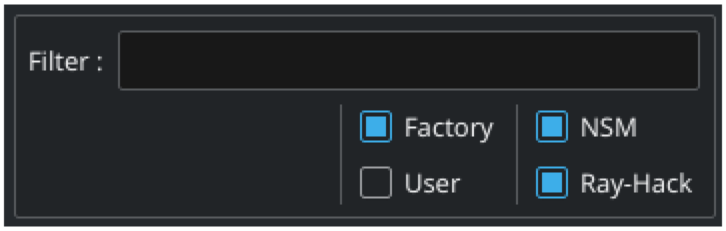
Ardour
Record, mix and master multi-track audio

Executable : ardour
Label : Ardour6
Name : Ardour

ADLjack, ADLplug, amsynth, Ardour, Calf JACK Host, Carla, Carla-Patchbay, Carla-Rack, drumkv1, Guitarix, GxTuner, Hydrogen, JACK Connections, Jack Mixer, Luppp, Mamba, Network Session

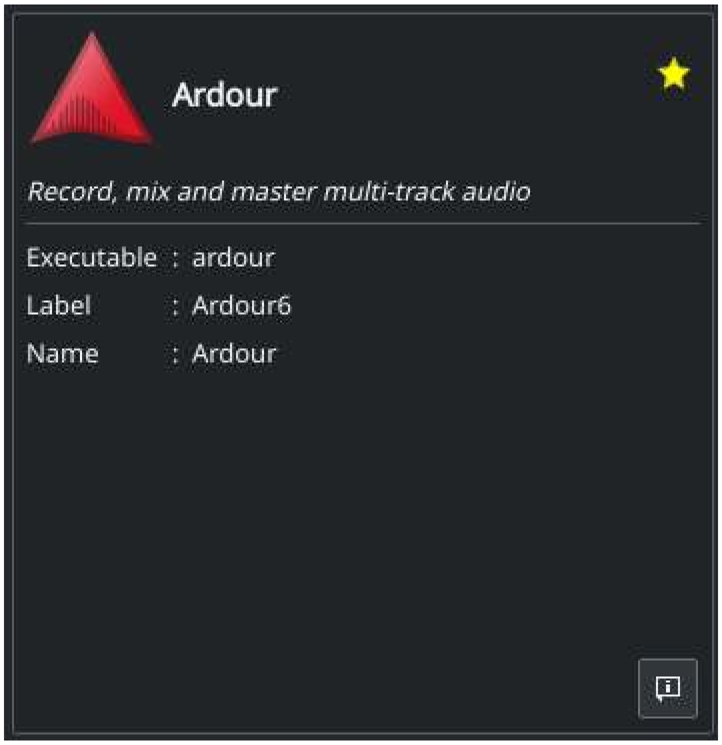
The list of available applications is on the right. If the software you want to add is not present here, see Add a program not provided.

Top left is the filter block



- the filter field allows you to enter a character string, only templates containing this character string in their name will appear.
- the **Factory** checkbox displays the templates integrated into RaySession or provided by your distribution
- **user** displays the templates created by the user by doing **Save as application template**
- **NSM** displays the NSM compatible clients, or launched as such
- **Ray-Hack** displays clients launched without NSM protocol

Bottom left the information block on the selected template on the right



Ardour

Record, mix and master multi-track audio

Executable : ardour
Label : Ardour6
Name : Ardour

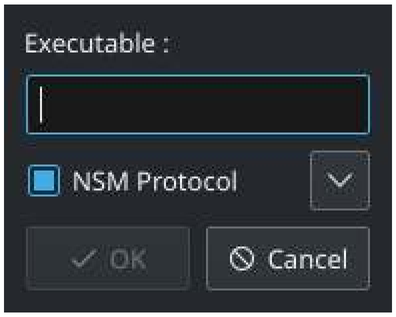
- at the top right of this block, a star, click on it to add it to favorites or remove it from favorites
- If it is a user template, a **User** button allows you to delete this template

- the button at the bottom right of this block allows you to access all the properties of the template, as in the [client_properties], except that nothing is editable.

Tip: This window is designed for very fast app addition, and behaves like *Alt+F2* on your desktop.
For example, from the main RaySession window, to add Carla type *Ctrl+A* , then `carla` , select the correct template with the Up/Down arrows, then Enter.

Add executable window

The window for adding an executable is launched by clicking on the **Executable** button (*or Ctrl+E*).



You will need to go through this window if you want to add a client that does not appear in the list of the Add application window. This window is very simple, a field to enter the executable, an **NSM Protocol** box, an advanced options button.

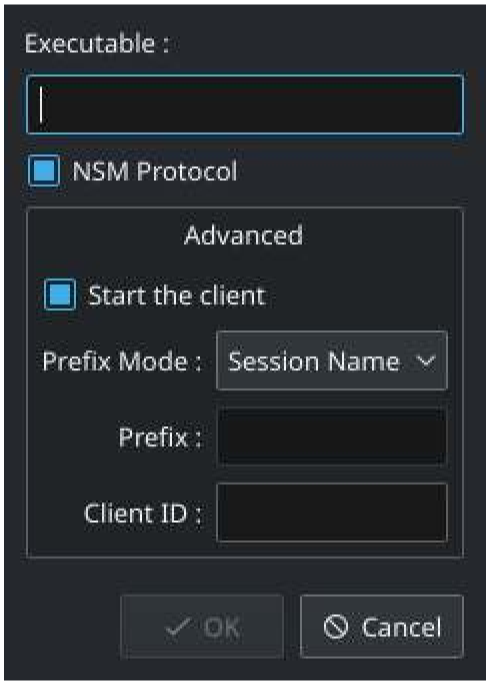
Leave the NSM protocol checked if:

- the program to launch is NSM compatible (if it is not in the list of applications, please let us know!)
- the program to run is a utility for which there is no need to save any project (QRest, Catia…). The state of such a program will remain on **launch** and will never be **ready**, it is irrelevant since they do not have a project to save.

If you leave the **NSM protocol** checked, it will not be possible to use the absolute path of an executable to add it, the executable must be located in a location provided for this purpose (you will not be able to launch `/usr/bin/my_program` , but `my_program`). You cannot enter arguments here, with or without the NSM protocol.

Unchecking **NSM Protocol** is equivalent to launching the program with the Ray-Hack pseudo-protocol.

If you click on the **advanced options** button, an advanced options block appears with



- the **Start the client** checkbox , if you uncheck it the client will be added but not launched
- the multiple choice box **Prefix Mode**, this defines the prefix of the name of the client's files
  - on **Session Name**, the file names will start with the session name, this is the default value
  - on **Client Name**, the file names will begin with the name provided by the client itself, as is the case with New Session Manager
  - on **Custom**, the file names will start with the value you enter in the **Prefix** field just below.
- the **Prefix** field which is only active if **Prefix Mode** is set to Custom.
- the **Client ID** field (client identifier). Enter only alphanumeric characters or '_'. This is useful if you want to catch and launch existing projects in the session with an executable. This is useful if you want to load in the session projects created outside a session. There is no method to make it easier, it depends a lot on the program you are using. RaySession will insult you if you enter a client ID that already exists in the session.

- the **Long JACK naming** checkbox. If this box is checked, clients will be supposed to use `ProgramName.ClientId` as JACK client name, else it will be `ProgramName`, possibly followed by `_N` (where N is a number). This is very useful if you use many instances of the same program, and you want to identify them easily in the patchbay.

Patchbay

Patchbay Manual

The JACK patchbay is displayed by default. You can hide it by clicking on **Control** and then uncheck **Show JACK Patchbay** (*or Ctrl+J*). The patchbay contains all the JACK AUDIO and MIDI ports that you can interconnect.

Obviously, if JACK is not started, this patchbay will be empty.

It is advisable to have the A2J bridge running if you want to work properly with MIDI. You can configure this via **Cadence**, **Studio Control**, or via the command line utility `a2j_control`.

RaySession does not include tools to configure the JACK server, **QJackCtl**, **Studio Control**, **Cadence**, or the command line utility `jack_control` do this job very well. Note that this patchbay also works with PipeWire.

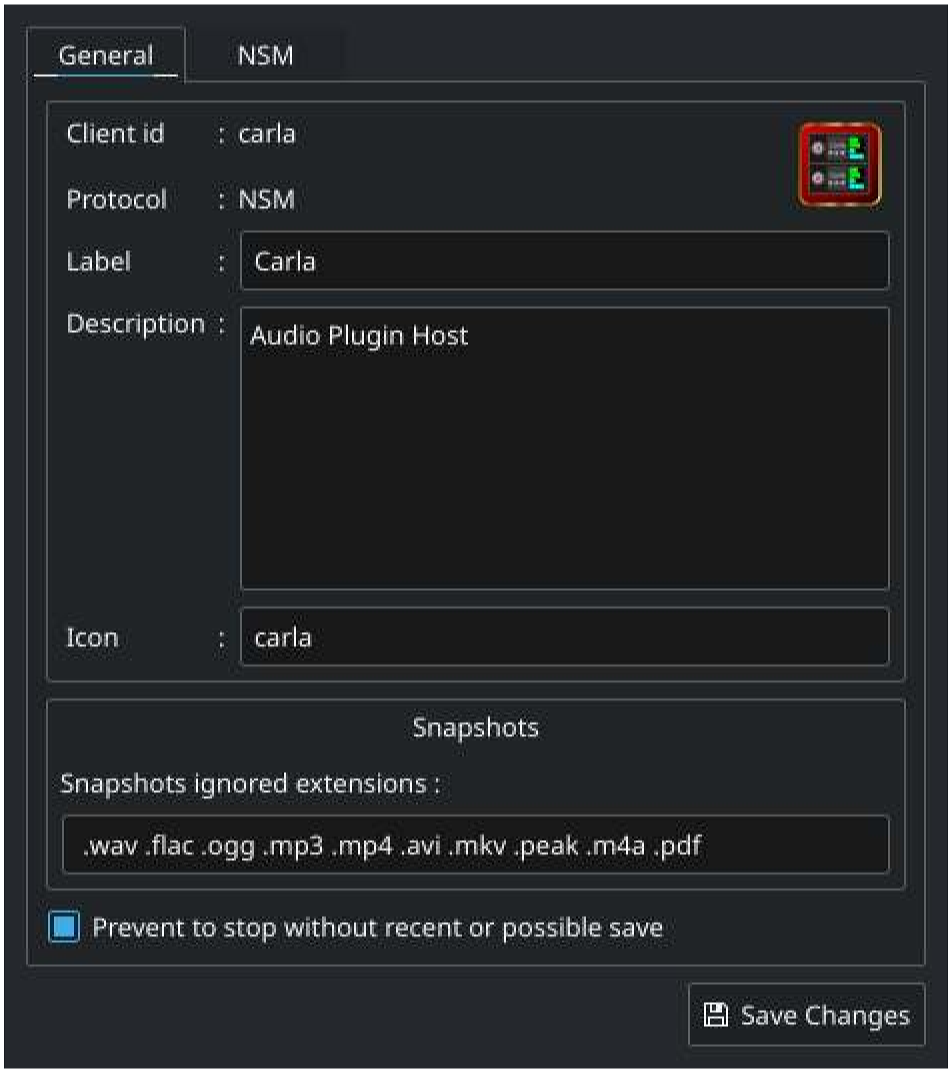
Learn more about the patchbay in the Patchbay Manual.

Client Properties Window

A client's properties window opens from the client menu by clicking Properties.

The client properties window has 2 tabs, a General tab and a tab specific to the protocol used by the client. Depending on the client protocol, the second tab is called NSM, Ray-Hack or Ray-Net.

General tab



Client id : carla
Protocol : NSM
Label : Carla
Description : Audio Plugin Host
Icon : carla

Snapshots
Snapshots ignored extensions :
.wav .flac .ogg .mp3 .mp4 .avi .mkv .peak .m4a .pdf

[✓] Prevent to stop without recent or possible save

The first block of the General tab displays the client ID, protocol, label, description and icon.
If you do not edit them, the label, description and icon are taken from the .desktop file associated with the launched executable, if found.
If you want to know the .desktop file used, type `ray_control client CLIENT_ID get_properties` in a terminal (replacing CLIENT_ID with the client identifier).

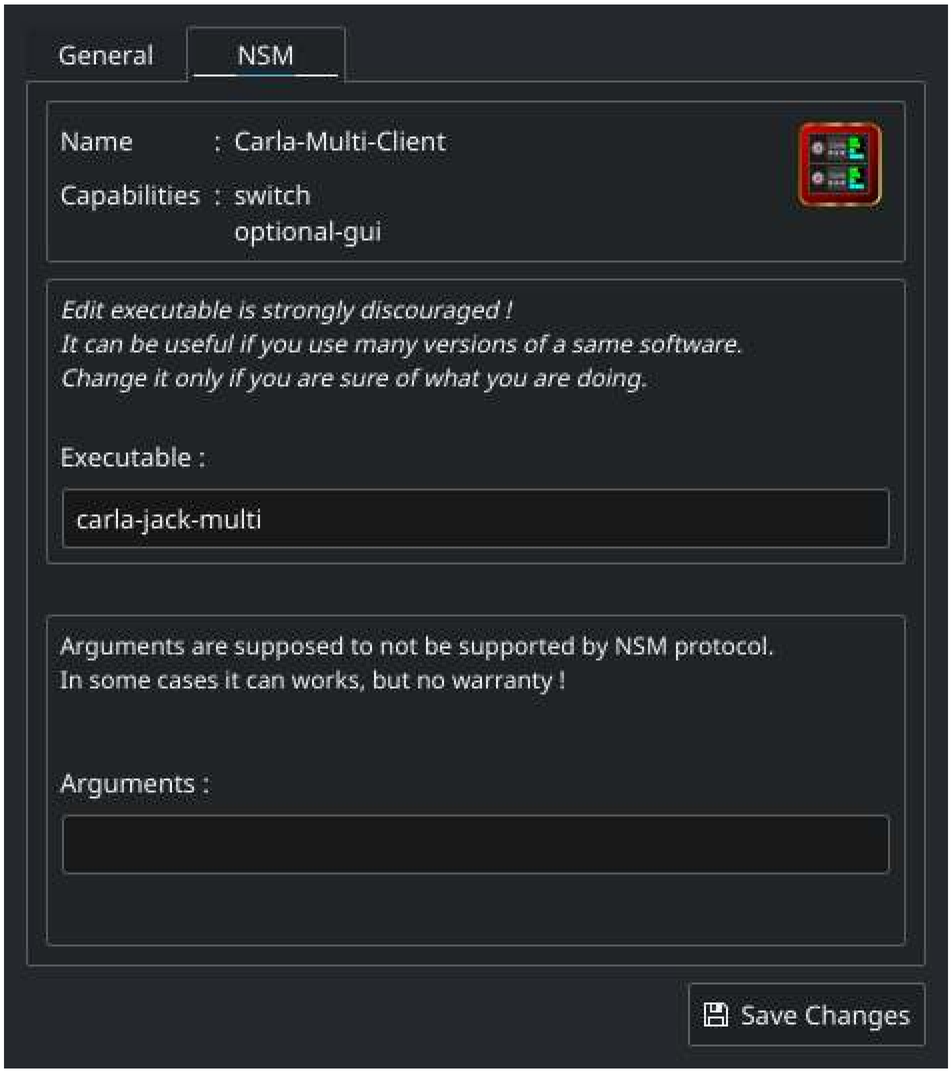
Then comes the block of snapshots, see Snapshots.

The checkbox **Prevent stop without recent or possible save** concerns the window that may appear when you ask a client to stop. If this box is unchecked, then the client will be stopped without a window warning you.
If the box is checked, the window will warn you when

- the client is unsaveable from RaySession
- we know that the client contains unsaved changes
- the client appears not to have been saved for more than a minute

Whether or not to check this box depends only on the importance of your client’s save, it’s up to you to judge. That said, if the warning is annoying, just check **Don’t prevent to stop this client again** in the warning window and **Prevent stop without recent or possible save** will be unchecked.

NSM tab



Name : Carla-Multi-Client
Capabilities : switch
optional-gui

Edit executable is strongly discouraged !
It can be useful if you use many versions of a same software.
Change it only if you are sure of what you are doing.

Executable :

carla-jack-multi

Arguments are supposed to not be supported by NSM protocol.
In some cases it can works, but no warranty !

Arguments :

The **Name** of the client here is provided by the client himself.
The **capabilities** are those which are transmitted by the client at its start-up. If the client has not yet been started, this field is therefore empty.

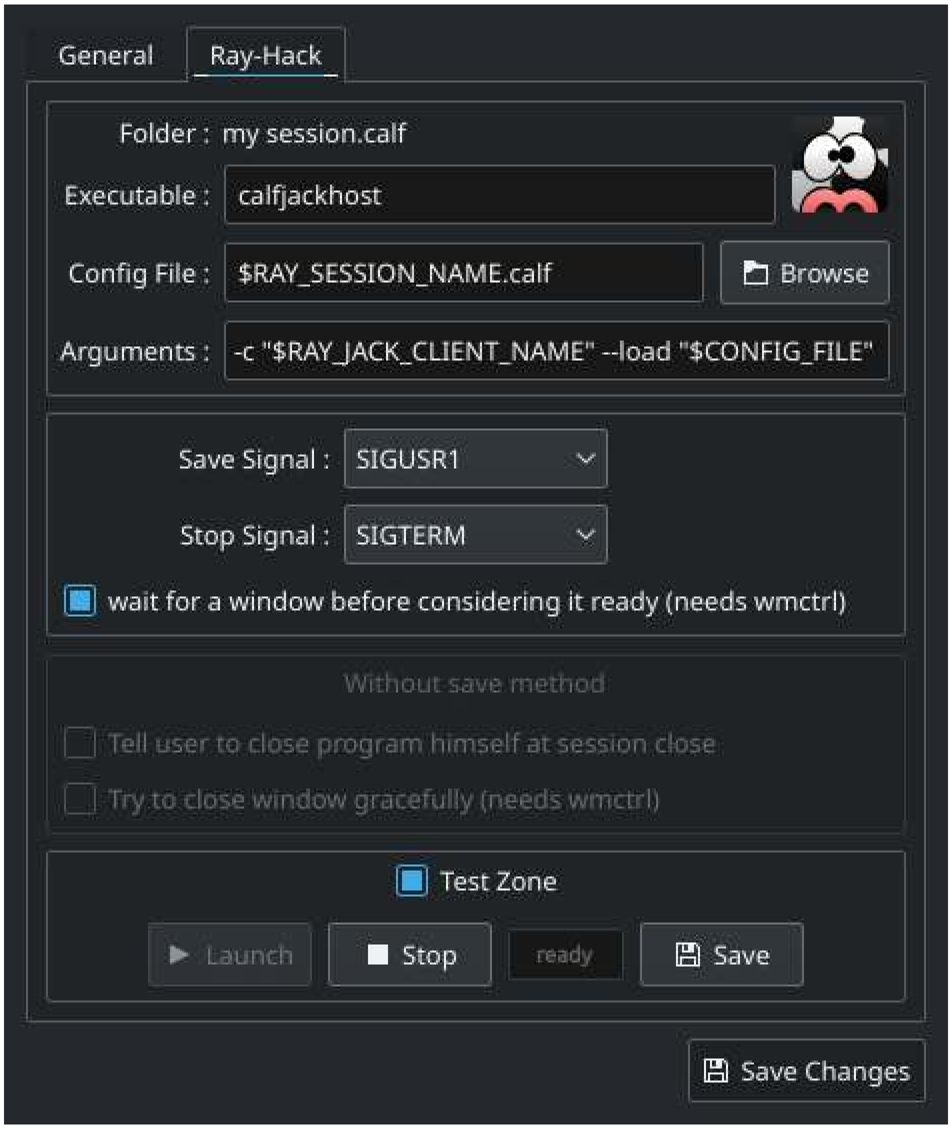
Editing the executable allows you to change the command that launches the client. Only change it to another executable capable of loading the existing client’s project. This is useful for example if you have two versions of Ardour, one running with the command `ardour` , the other with `Ardour6` , and you want to change which version to use.

Editing the arguments is strongly discouraged, and is especially not suitable for loading a file as an argument.

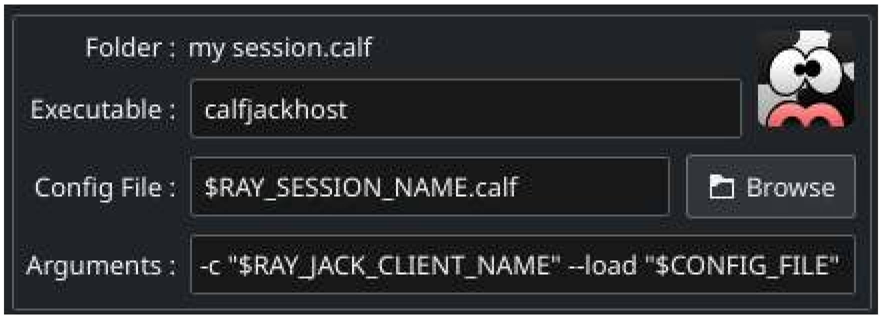
Ray-Hack tab

If the client is a Ray-Hack type, here many fields are available to you. This is not necessarily good news, the idea is to be able to load a program into the session that is not (yet) compatible with NSM. If properly implemented in the client, the NSM protocol will always be much more comfortable to use and more reliable than this hack. That said, if we can expect the NSM implementation in all audio programs, this is not the case for other programs which can still be useful in the session.

The Ray-Hack pseudo-protocol uses the attributes of proxies (nsm-proxy or ray-proxy), except that the client is launched directly in the session.


Folder : my session.calf
Executable : calfjackhost
Config File : $RAY_SESSION_NAME.calf
Arguments : -c "$RAY_JACK_CLIENT_NAME" --load "$CONFIG_FILE"
Save Signal : SIGUSR1
Stop Signal : SIGTERM
[✓] wait for a window before considering it ready (needs wmctrl)
Without save method
[ ] Tell user to close program himself at session close
[ ] Try to close window gracefully (needs wmctrl)
[✓] Test Zone

Launch block



Folder : my session.calf

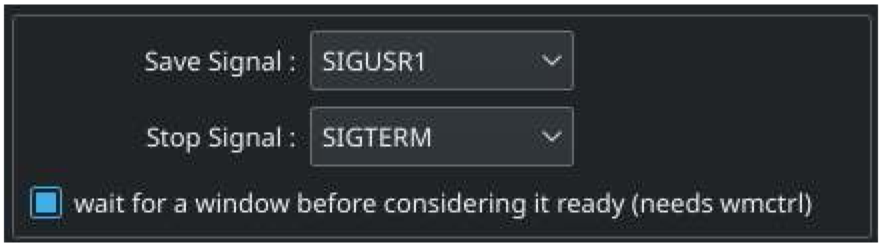
Executable : calfjackhost

Config File : $RAY_SESSION_NAME.calf

Arguments : -c "$RAY_JACK_CLIENT_NAME" --load "$CONFIG_FILE"

- the **Folder** is the folder name of this client in the session folder. The program is launched from this folder.
- the **Executable** is the command that starts the program
- The **Config file** will be the project file that we want to open with this program. It is more than highly recommended to reference a file in the client folder.
  The variable `$RAY_SESSION_NAME` will be automatically replaced by the name of the session.
  If this field is empty, the client status will always remain **launch** and will never be **ready**. In some cases, therefore, it may be useful to type anything here rather than nothing.
- The **Browse** button opens a dialog box to find the project file and fill in the **Configuration file** field
- The **Arguments** field includes the arguments passed to the Executable command the arguments are split as they would be in a terminal, don't forget the " or ' if necessary.
  For example to reproduce `my_command my_argument_1 "my argument 2"` enter `my_command` in the **Executable** field and `my_command my_argument_1 "my argument 2 "` in the **Arguments** field.

Signals block



Save Signal : SIGUSR1

Stop Signal : SIGTERM

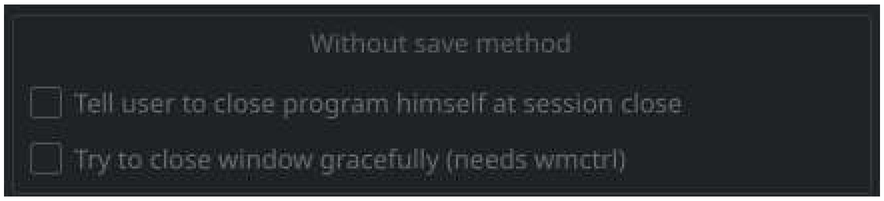
☑ wait for a window before considering it ready (needs wmctrl)

- **Save Signal** can be only rarely used. It can be SIGUSR1 for programs compatible with the old LASH protocol. Otherwise leave it on *None*, if there is no save method, we cannot invent it.

- **Stop Signal** will usually be SIGTERM. Only change it if this signal does not close the program correctly.
- If **Wait for a window before considered it ready** box is checked, then the client status will only change to **ready** when a window appears.
  If `wmctrl` is not installed, or the window manager does not seem to be compatible with it, then the client status will be **ready** half a second after it is launched.

With `ray_control` you can assign signals other than those offered in the multiple choice boxes. For example `ray_control client CLIENT_ID set_properties save_sig:22` will define the SIGTTOU signal for the client CLIENT_ID save.
Type `kill -l` to see the available signals and their numbers.

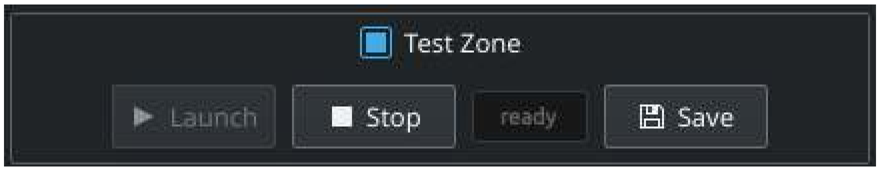
Non-saveable management block


Without save method
Tell user to close program himself at session close
Try to close window gracefully (needs wmctrl)

This block is active only if the **Save Signal** is set to *None*.

- if **Tell user to close program himself at session close** is checked, the client will be considered as not saveable and an orange exclamation mark will appear in front of its save icon. When closing the session, RaySession will wait for you to close the program yourself because it is impossible to know if it contains unsaved changes.
- If **Try to close window gracefully** is checked, then at session close, RaySession will try to close the window as if you were closing the program window. This is very useful when the program reacts by closing if there are no unsaved changes and displaying a close confirmation window in the opposite case (most programs react in this way). If `wmctrl` is not installed or the window manager does not seem to be compatible, you will have to close the program yourself in any case.

Test area



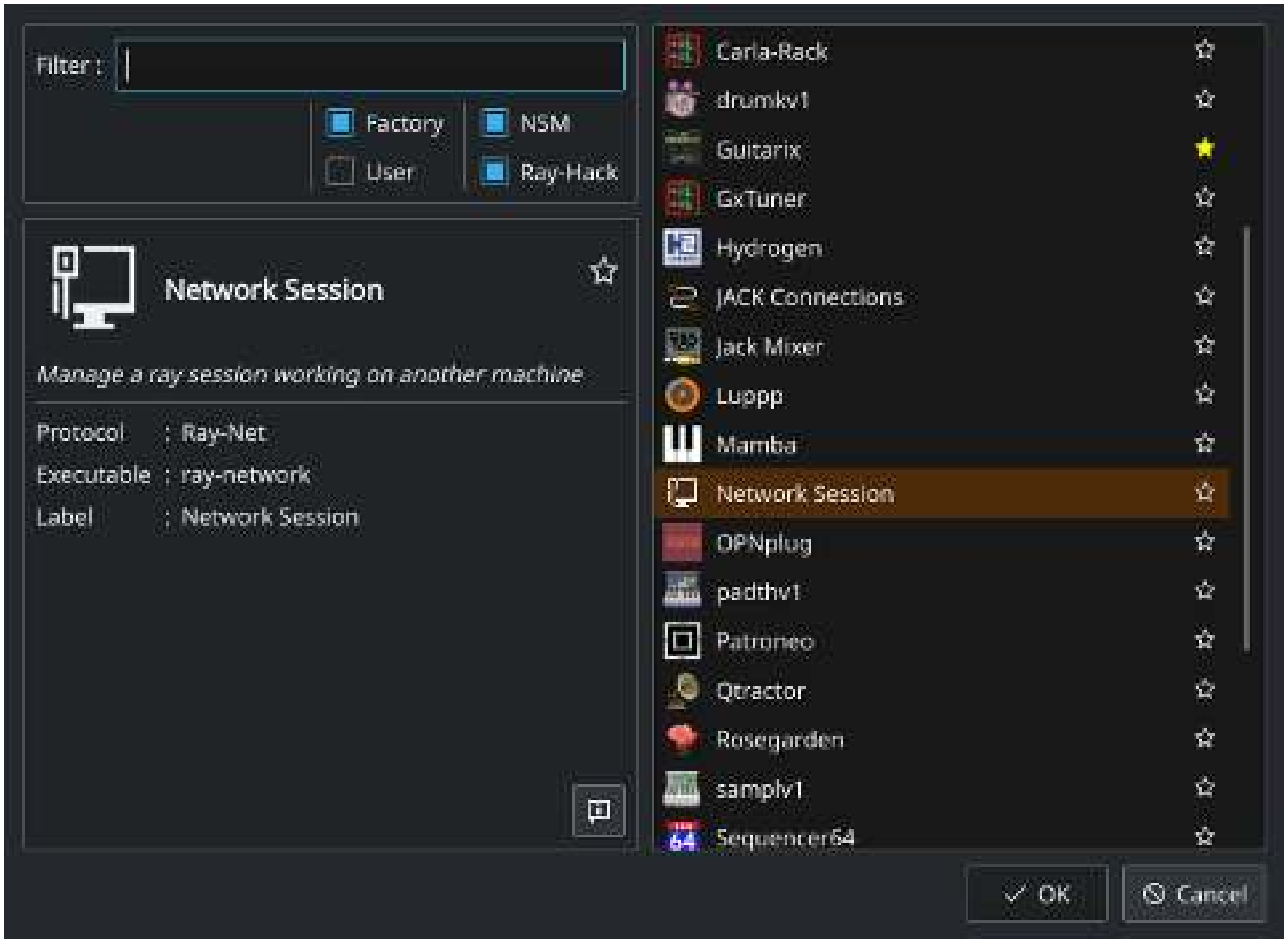
The test area allows you to test the start, stop, and save settings set in this window without having to **Save the changes**.

Network Sessions

A network session allows you to launch another session on another machine at the same time as your session. This can be particularly useful if you are using net-jack to unload your machine from part of the DSP, if you have greedy effects running on another machine for example.

Network sessions operate on a master-to-slave basis. A session is master and can have several slave sessions which are themselves masters of other slaves, but such a scenario seems completely out of the ordinary. Organize yourself simply: one master, one or more slave(s).

To launch a network session (therefore a slave), launch the **Network Session** application template from the applications window and follow the instructions.



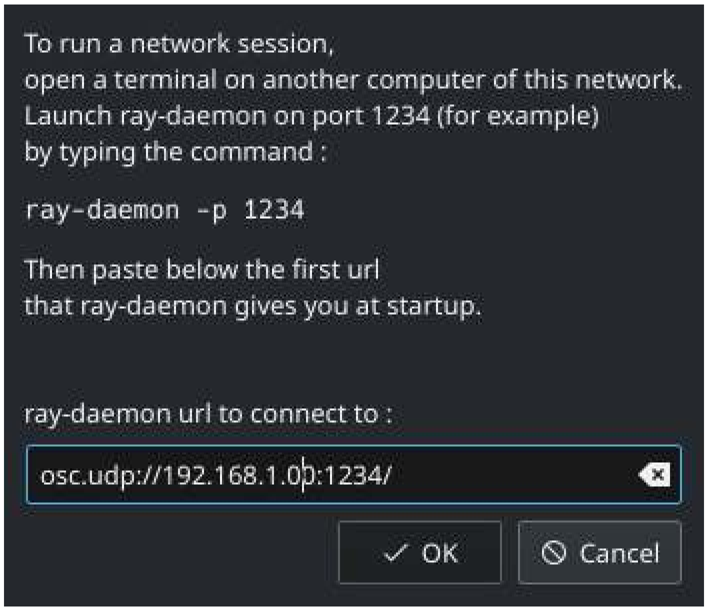
Manage a ray session working on another machine

Protocol : Ray-Net
Executable : ray-network
Label : Network Session

You will have to start a daemon on the remote machine with the command `ray-daemon -p 1234` (`1234` is an example, put the port you want). This daemon displays something like this in the terminal:

```
[ray-daemon]URL : osc.udp://192.168.1.00:1234/`
[ray-daemon]      osc.udp://nom-de-machine:1234/`
```

```
[ray-daemon]ROOT: /home/utilisateur/Ray Sessions reseau
```


To run a network session,
open a terminal on another computer of this network.
Launch ray-daemon on port 1234 (for example)
by typing the command :

```
ray-daemon -p 1234
```

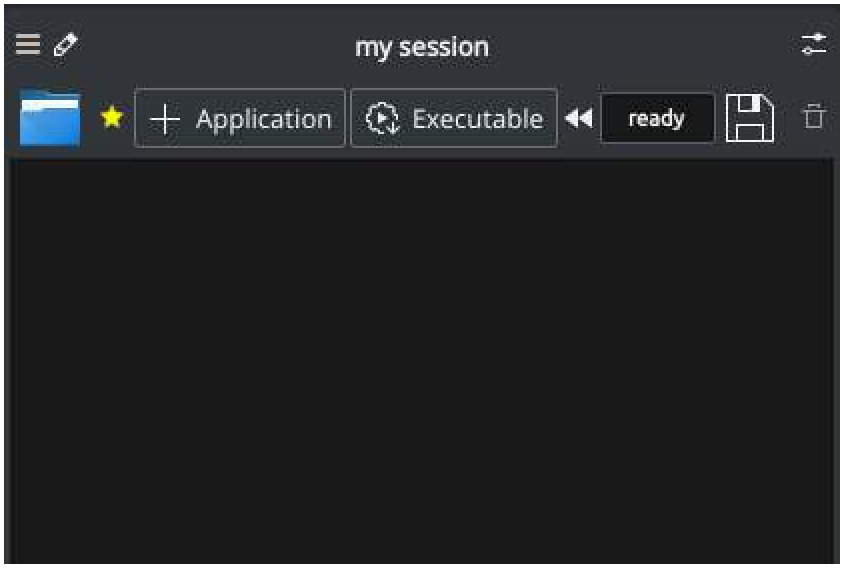
Then paste below the first url
that ray-daemon gives you at startup.

ray-daemon url to connect to :

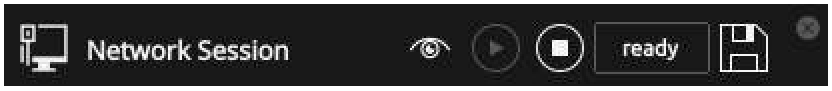
osc.udp://192.168.1.00:1234/

You will need to copy one of the two URLs into the network session invitation window. The first (which begins with osc.udp://192.168.) must work for sure, the second will work only if the name of the slave machine is correctly entered in the file `/etc/hosts` of the master machine. However, entering the name of the slave machine in `/etc/hosts` and using the second URL is preferable, because the address in 192.168. will move if you connect your slave machine differently (wired, wifi), or if you reinstall the distribution.

You now have 2 RaySession windows on your master machine, one controls the master session, the other the slave. You will recognize the slave by the fact that it does not have a toolbar (**New Session**, **Open Session**, **Control**), nor **Abort session** and **Close session** buttons.



The slave window is hideable as is the case in many NSM programs.



If you run `raysession -p 1234` on your slave machine, you will have the slave session window in duplicate, one on each machine.

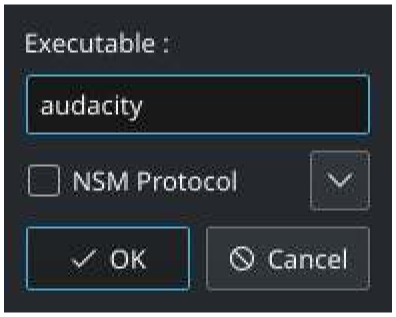
Tip: Put this `ray-daemon -p 1234` in your slave machine startup.

Add a program not provided

If the program you want to add does not manage a project to save, click on **Executable**, enter the name of the executable and click on **Ok**. Otherwise follow this example.

We want to add Audacity to the session here. Audacity is chosen as an example because it is known and generally installed on audio distributions. This is not necessarily a very suitable program for the modularity of an audio session given the way it handles JACK.

Click **Executable** (*or Ctrl+E*). In the Add executable window, Uncheck the **NSM Protocol** box, type `audacity` in the **Executable** field and click **Ok**.


Executable : audacity

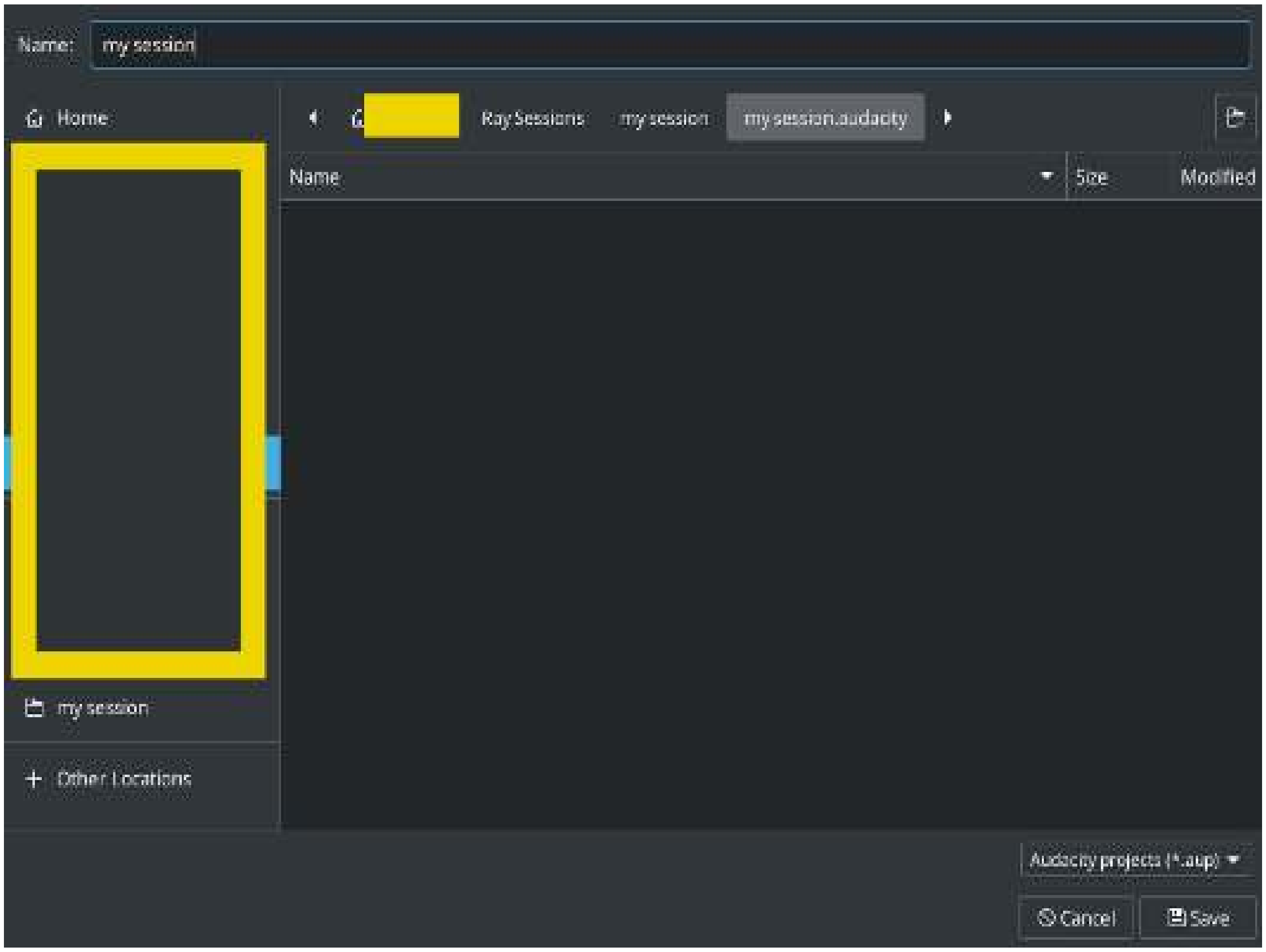
☐ NSM Protocol

A new client is created, its properties window opens on the **Ray-Hack** tab and Audacity is launched.

In Audacity, we will directly save an empty project in the client's folder. The client's folder is located in the session folder and has the name given after **Folder:** at the top of the **Ray-Hack** tab. We will call the project EXACTLY like the current RaySession session. To do this, in Audacity, go to *Menu → File → Save project → Save project*.

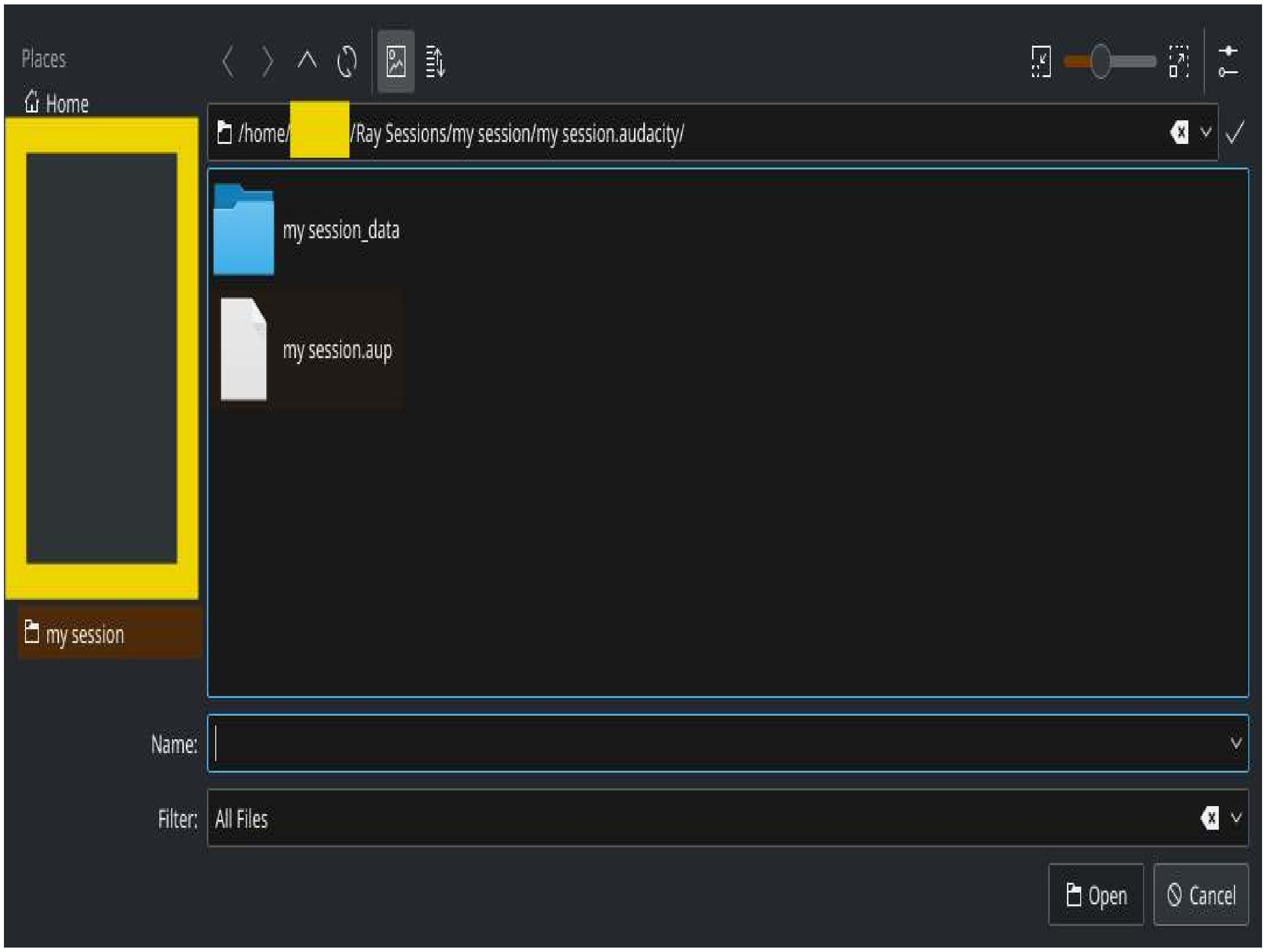
Save empty Audacity project

Click **Validate** at the possible warning window.
In the save files box that opens, you will find the session folder at the bottom left (see Provide bookmarks for session folder), click on it to enter it. Inside this you should see the client's folder as it appears at the top of the Ray-Hack tab, enter this folder. At the top left of the backup box, type the exact name of your session in the **Name:** field then validate.



Name: my session

Close Audacity.
At the top right of the **Ray-Hack** tab of the client properties window, click **Browse**.

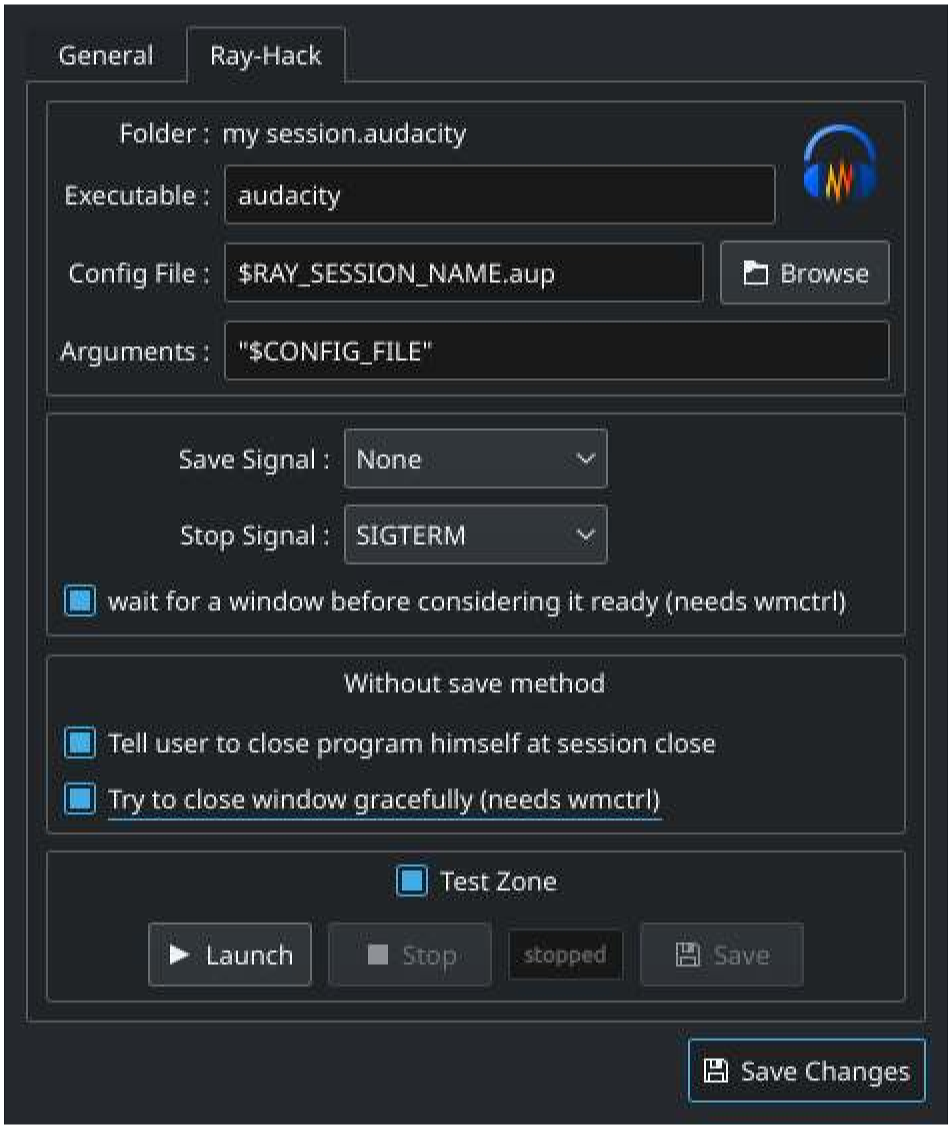


/home/[redacted]/Ray Sessions/my session/my session.audacity/

Filter: All Files

select the Audacity project you just created, its name starts with the session name and ends with .aup.

If all went well, the **Configuration File** field became `$RAY_SESSION_NAME.aup` and the **Arguments** field became `"$CONFIG_FILE"` .
Check the boxes **Wait for a window before being considered ready**, **Ask the user to close the program himself** and **Try to close the window gracefully**. Click in the bottom right corner on **Save Changes**.



Folder : my session.audacity

Executable : audacity

Config File : $RAY_SESSION_NAME.aup

Arguments : "$CONFIG_FILE"

Save Signal : None

Stop Signal : SIGTERM

☑ wait for a window before considering it ready (needs wmctrl)

Without save method

☑ Tell user to close program himself at session close

☑ Try to close window gracefully (needs wmctrl)

☑ Test Zone

Launch the Audacity client and verify that the Audacity window has the name of the session.
Click on the Audacity client icon, in the drop-down menu choose **Save as an application template**, and enter `Audacity` the field of the dialog box that has appeared. Now when you want to launch Audacity in the session, all you have to do is launch the Audacity template from the Add application window.

Note that the client's save button is behind an orange exclamation point, this means that RaySession is not able to save its project and that you will have to do it yourself.

Depending on what program you want to add to the session, it might not always be that easy. Some programs will require an argument that precedes the configuration file, in this case type `my_program --help` or `man my_program` to know how to load a project when the program starts, and adapt this in the **Arguments** field.

Import an NSM session

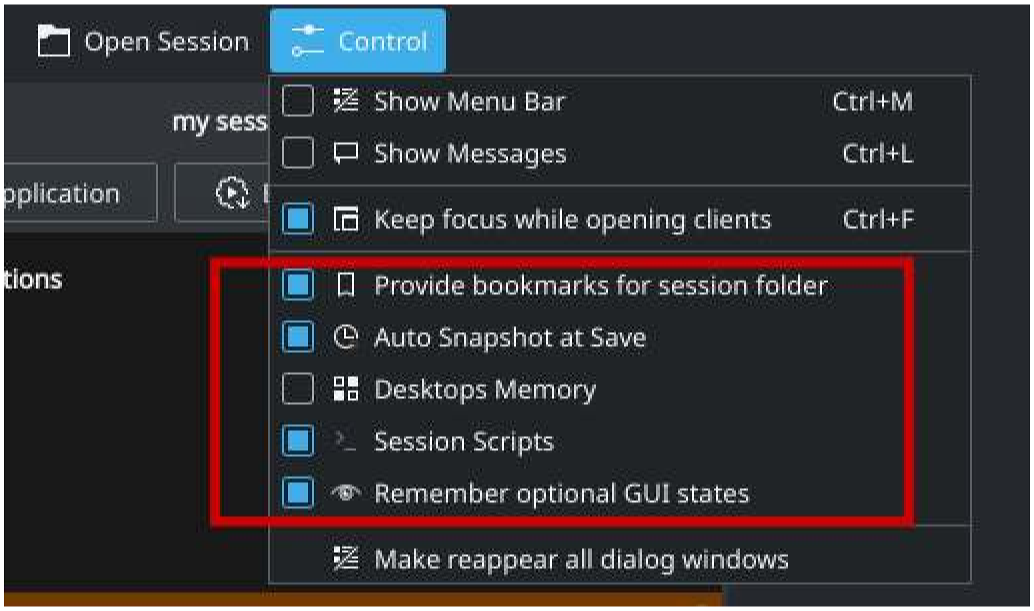
To import a session created with Non Session Manager or New Session Manager, move or copy the session folder to the RaySession root sessions folder (default ~/Ray Sessions). Then click **Open session**, your session should appear in the list of sessions, double-click on it.

RaySession will not rewrite clients added or deleted to the `session.nsm` file, as long as you open an NSM session with RaySession you must continue with RaySession.

If ever you want to use another session manager, you will find in the menu **File** → **Utilities** → **Convert the session to NSM file format**. Obviously this leads to some changes, such as prefix mode or the way the JACK clients will be named, but nothing that prevents the session from working.

The daemon’s options

Daemon options are services which can be activated and deactivated via the **Control** button at the top right of the main window, or via the options menu in the menu bar.



- [ ] Show Menu Bar Ctrl+M
- [ ] Show Messages Ctrl+L
- [x] Keep focus while opening clients Ctrl+F
- [x] Provide bookmarks for session folder
- [x] Auto Snapshot at Save
- [ ] Desktops Memory
- [x] Session Scripts
- [x] Remember optional GUI states

Here are the details of the different options:

Provide bookmarks for session folder

In audio production, creating audio or midi file with one program and load it into another is an usual case. This option offers something purely practical: a shortcut to the current session folder in your file manager and in the dialog boxes provided for fetching or saving files. It simply avoids wasting time browsing through your personal folder tree to find a file that you have put in your session folder, since that is where it belongs.
Of course, this shortcut is deleted when the session is unloaded.
Technically, shortcuts are created for GTK2, GTK3, QT4, QT5, KDE and FLTK.

Auto Snapshot at Save

This option is far from being trivial, it allows you to take a snapshot of the session after each session save. This means that in case of a technical or artistic error you will be able to find the session in the state it was in at the moment of the snapshot. This option requires that you have the `git` program installed. See Snapshots for more details.

Desktops Memory

If this option is activated, RaySession will save (or attempt to save) the number of the virtual desktop on which the client windows were located when the session was saved. So when you restart the session or the clients, the windows will be redispatched to the desktops on which they appeared. This option requires you to have the program `wmctrl` installed to work, and probably will not work with Wayland.

Session scripts

Disable this option to not activate any session script, and thus open, save or close a session completely ignoring the scripts associated with these actions. These scripts are used by sessions with JACK configuration memory. See Session scripts for more details.

Remember optional GUI states

This option only concerns NSM clients capable of showing/hiding their graphical interface. Without this option, some of them will always start hidden, others will remember if they were visible when they were last saved. With this option enabled, the graphical interfaces will be displayed when the session is ready if they were visible during the last save or if the client has never been launched.

Snapshots

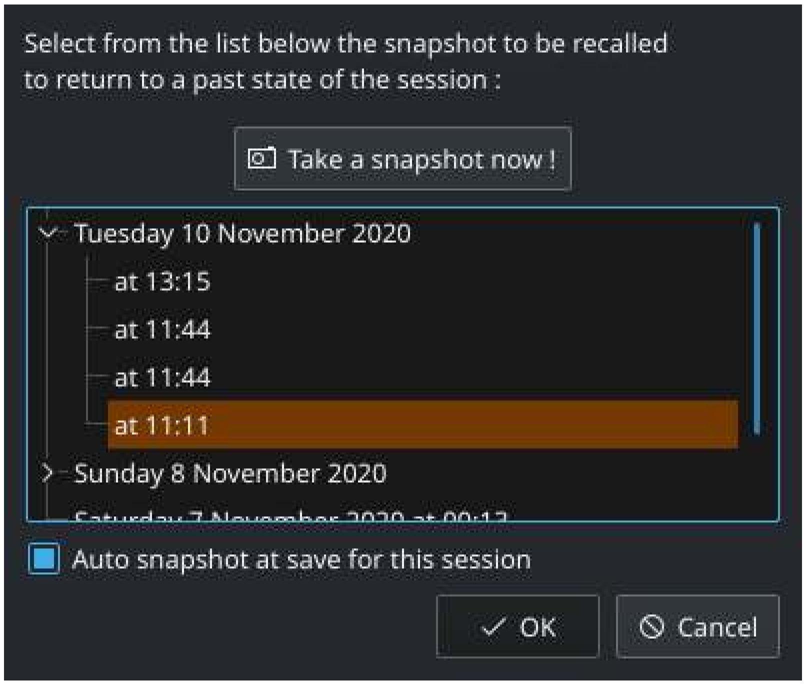
Snapshots require you to have the program `git` installed, if you don't have `git` , the reverse button does not appear and it is not possible to take or return to a snapshot.

A snapshot stores files and their contents at a specific time. Large files and files with certain extensions such as audio and video files are ignored, otherwise the snapshot process will take too long and the size of the session folders will needlessly double. This is actually not very annoying, on the contrary, since your recent audio files remain present when you go back to a previous snapshot.
If despite everything the snapshot process turns out to be long, a window appears and you can safely cancel the current snapshot. If you cancel it, the automatic snapshot will no longer take place for this session.

The interest of the snapshots lies in the fact of being able to return to the previous moment of the session, before having had this brilliant artistic idea which turned out to be null and void, before having attempted a recutting of the samples with the microcoscope which finally killed all forms of musicality, before a program crashes for some reason unknown to the police…

Don't worry, going back to a snapshot won't stop you from getting back to where you were.

To revert the session to a snapshot, click the reverse button located to the right of the **Executable** button.


Select from the list below the snapshot to be recalled
to return to a past state of the session :

- Tuesday 10 November 2020
  - at 13:15
  - at 11:44
  - at 11:44
  - at 11:11
- Sunday 8 November 2020

[✓] Auto snapshot at save for this session

Select the snapshot you want to revert to and click **Ok**. A new snapshot is taken, the session closes, the desired snapshot is recalled and the session reopens.

It is also possible to return only a client to a previous state of the session by right-clicking on the client,then **Return to a previous state**. If you want you can edit for each client the files ignored by the snapshots in the [client_properties].

With the **Automatic snapshot after save** option, a snapshot is taken immediately after each backup of the session, unless there is no change since the previous snapshot. To take a snapshot at another time, click on the reverse icon to the right of the **Executable** button and on **Take a snapshot now**, this has the advantage of being able to name the snapshot and thus having a more meaningful time mark than the date and time of the snapshot.

Session scripts

Session scripts allow you to program personalized actions when opening, saving and closing the session. They are used in particular for sessions with JACK configuration memory.
Knowledge of shell scripting is required to edit these scripts, but anyone can use them.

Session scripts are located in a folder `ray-scripts` located either in a session folder or in a parent folder.
For example, for a session being in:
`~/Ray Sessions/with_foo_script/my session`

the session scripts folder may be

```
~/Ray Sessions/with_foo_script/my session/ray-scripts
~/Ray Sessions/with_foo_script/ray-scripts
~/Ray Sessions/ray-scripts
~/ray-scripts
```

The advantage of such behavior is to be able to script a set of sessions without having to copy the scripts there, but above all to deliver an unscripted session when it is transferred to someone else for collective work.
Only the script folder closest to the session in the tree will be considered. Thus, a `ray-scripts` empty folder in a session will disable scripts for that session.

To edit the scripts, start by creating a session with the template with the basic scripts, this is a template session with scripts that does not include any particular action. Go to the folder `ray-scripts` in the session folder, you will find the files `load.sh`, `save.sh` and `close.sh`. In each of these scripts, `ray_control run_step` corresponds to the normal action performed (depending on the script: load, save or close the session). If one of these three scripts is of no use to you, delete it, it will save time not to go through that script.

The script files must imperatively be executable to work.

You will probably need the command line utility `ray_control` to perform actions relating to a particular client. type `ray_control --help` to know all its possibilities, see also Control RaySession from the command line.

JACK memory configuration session template uses session scripts, but we can also imagine many possible actions according to your needs and desires, for example:

- define a specific order for launching clients when the session is opened (an example is provided in the source code)
- make a backup copy of the session on an external hard drive each time you close it
- send a *Ctrl+S* shortcut to non-saveable client windows when saving the session (an example is provided in the source code)
- Turn on the red light at the entrance to the studio when opening, turn it off when closing
- Start the coffee machine at the end of the session (stupid example, go and press the button on the coffee maker, anyway you will have to change the filter!)
- Make many, many, many mistakes that will crash your session, be careful of course!

JACK configuration memory

It is possible thanks to session scripts to automatically recall the JACK configuration specific to a session before loading it. This behavior may remind some of the operation of LADISH studios, much better done, at least that’s what is hoped.

In which cases to use it

This can be useful:

- If you need to use a specific audio interface for the session
- If you are working on multiple projects with different sample rates (such session at 44100 Hz, such session at 48000 Hz).
  This will prevent you from having to reconfigure, stop and restart JACK yourself, or even avoid forgetting to do so and being insulted by certain programs.
- If you want to avoid loading a very DSP-intensive session (for example in the mixing phase) with a too small buffer (128 for example).
  Note that on most audio interfaces it is possible to change the buffer size hotly (without restarting JACK).

Usage

The Session scripts option must be enabled (This option is enabled by default).

To use the JACK configuration memory, create a new session from the **With JACK configuration memory** template. It is in fact a scripted session (see Session scripts) which launches a script supplied with RaySession, but which is completely external to it, so RaySession still has no direct relation to JACK.

Read the information window on this subject then validate. JACK restarts then your session starts.

Working principle

Each time the session is saved, the JACK configuration is saved in the session, in the `jack_parameters` file.
Before opening the session, JACK is restarted if the configuration of the session is different from the current configuration of JACK.
After closing the session, JACK is restarted if necessary with the current configuration before opening.

The configuration of the PulseAudio → JACK bridges is also saved and restored with the JACK configuration.

If you open this session after having copied it to another computer, the JACK configuration will not be recalled but will be overwritten when saving. Only the sampling frequency of the session will be used.

Special cases

To open a session without reloading its configuration from JACK:

- disable the **Session scripts** option
- open the session

To change the JACK configuration of a session:

- Start JACK with the desired configuration
- Disable the Session scripts option
- Open the session
- Re-enable the Session scripts option
- Save the session

To make an old session sensitive to the configuration of JACK

- copy the folder `ray-scripts` of a session with memory from the JACK configuration to the session folder
- Activate the Session scripts option
- Open the session

or

- move the session to a sub-folder containing the good one `ray-scripts` folder
- Open the session

Under the hood

RaySession is really just a GUI for ray-daemon. When you launch RaySession, the GUI launches and connects to the daemon, and it stops the daemon when closed. The graphical interface and the daemon communicate with each other by OSC (Open Sound Control) messages, as is the case between the daemon and the NSM clients. Thus, you can connect several graphical interfaces to a daemon, even remotely. Type `raysession --help` to see how.

It is not forbidden to have several daemon instances launched simultaneously, so if you launch RaySession while an instance is already launched, it will launch a new daemon. However, this way of working being unusual, the use of a single daemon is favored. So, if a daemon is running and it has no GUI attached, raysession will connect to that daemon by default.

Control RaySession from the command line

the command `ray_control` lets you do just about anything you can do with the GUI, and a little more. type `ray_control --help-all` to know all the possibilities.

In case there are multiple daemons started (see Under the hood), `ray_control` will only consider the one that was started first, unless you specify its OSC port with the `--port` option or `RAY_CONTROL_PORT` environment variable.

One might think that there is no point in using `ray_control` since the command `oscsend` allows to send an OSC message to the daemon, but it is false.
Firstly, because `oscsend` allows you to send messages but not to obtain information in a simple way (which are the active clients? What is the executable of such and such client? ...), secondly, because the command `ray_control` will end when the requested action is taken, for example `ray_control open_session "my session"` will end when the session is loaded.

Remember to assign `ray_control save` to a global shortcut of your desktop environment (*Ctrl+Meta+S*), this will save you a lot of time!

Frequently Asked Questions

- **Is it still worth running Ardour (or another NSM compatible DAW) directly rather than in RaySession?**

  Except for a really tiny project, no. If you are using Ardour, always run it from RaySession, firstly, the automatic snapshot after save can be of unexpected help to you, secondly, you are not immune to needing another program even if you did not plan it.

- **Can I launch an Ardour session launched normally into a RaySession session ?**

You will find in the menu **File → Utililies → Convert an Ardour session to a Ray session**, it creates and launch a RaySession session from an ardour session. It is adviced to backup your ardour session first. Obviously, Ardour must not be open with this session during the execution of this script, but you already knew that.