Chino

scripted meta-applications for Linux audio

David Adler

Linux Audio Conference 2013

Modular sessions

Some programs, interconnected via:

- Jack audio
- Jack Midi
- Alsa Midi

Connection graphs

Assume a system with

P	port-types, each having
O_p	output ports and
I_p	input ports.

The number of all possible connection graphs Ω then is

$$\Omega = 2^{\sum_{p=1}^{P} O_p I_p}.$$

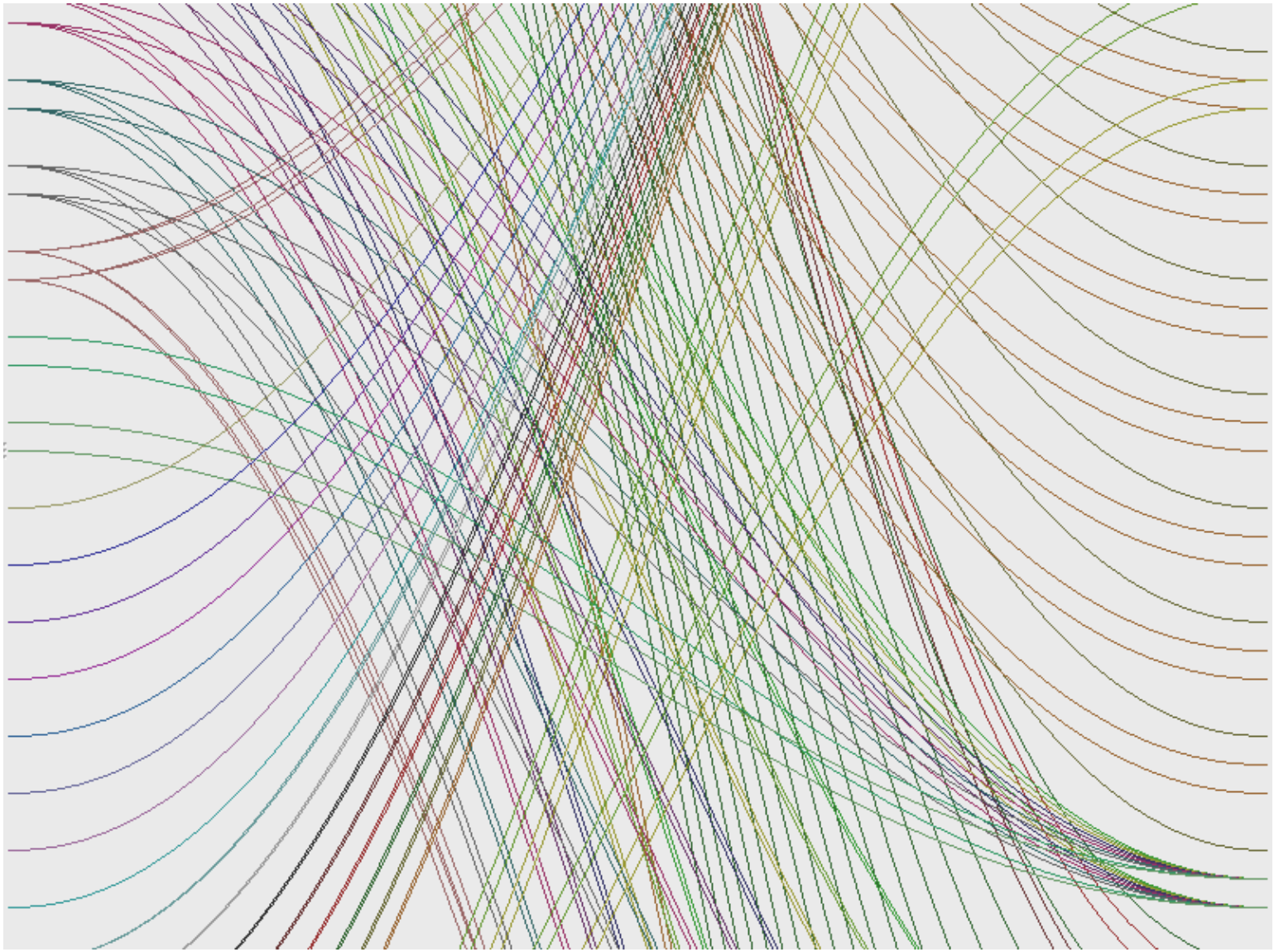
Connection graphs

$$
\begin{aligned}
P &= 2 && \text{two port types (audio and Midi)} \\
O_{audio} &= 60 && \text{audio out} \\
I_{audio} &= 40 && \text{audio in} \\
O_{Midi} &= 10 && \text{Midi out} \\
I_{Midi} &= 24 && \text{Midi in}
\end{aligned}
$$

$$
\begin{aligned}
\Omega &= 2^{60 \times 40 + 10 \times 24} \\
&\approx 5.2383 \times 10^{794}
\end{aligned}
$$

523827811845807022473069412697450810840202138552146390343923852550252328373526611242634
304443288124903155859584171121522190194441642368311822810617101824229763752012138871400
338141475333310613748352834274204391111543458179386779419426670226906155229104912130390
217334114948465567997725583506626760079327284282190636044973095394755719877387745505253
662881688145891886121778965082712737352645959266197863386104570006162250453485561771001
996794496786970643305368320196449988307824326955983391060400307901108132237578916039599
486559805293158850454806147553660894103328655100818563235679005646554430079461326485138
236231102341411657846260093852444901136796765137831897844865351140917190023079525082293
317414748034454180435686945722914901460093887155049566784630708713973253922095471423612
774856523776

Session management...



...is desirable.

History

Once upon a time [...]

LADCCA (2003)
Linux **A**udio **D**evelopers **C**onfiguration and **C**onnection **A**PI
`https://savannah.nongnu.org/projects/ladcca`

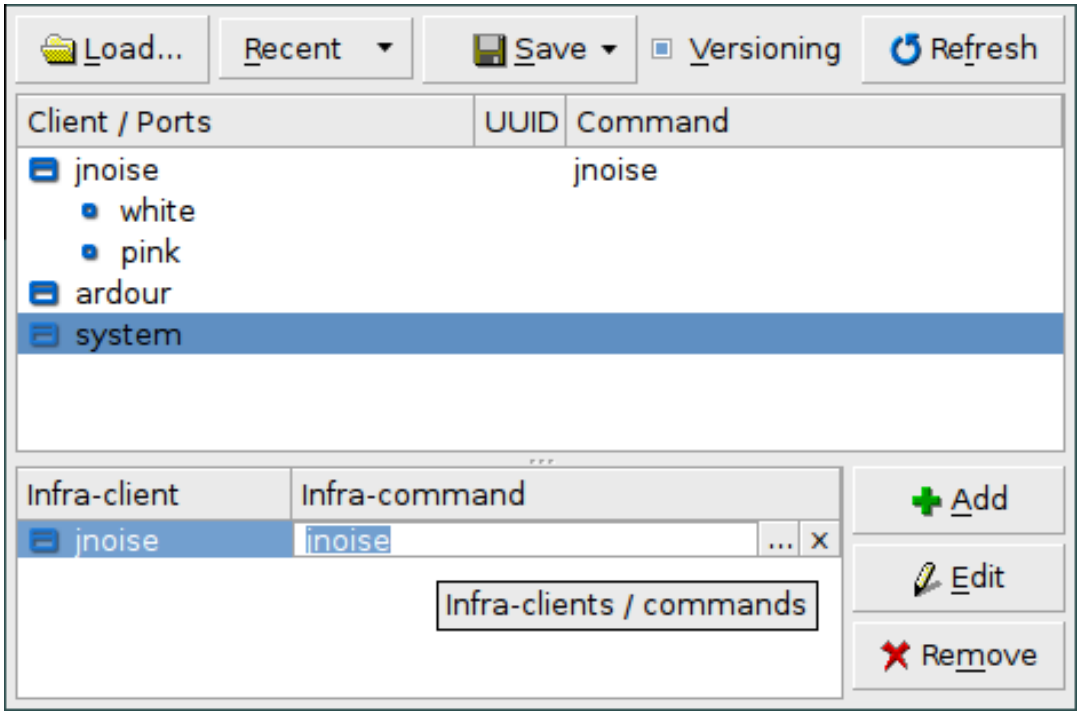
superseded by

LASH (2005)
Linux **A**udio **S**ession **H**andler
`https://savannah.nongnu.org/projects/lash`

which is still alive.

JACK Session

`http://trac.jackaudio.org/wiki/WalkThrough/User/jack_session`



Frontends:

- pyjacksm
- QjackCtl
- and . . .

In QJackCtl, non-compliant applications can be added as “Infra-clients”.

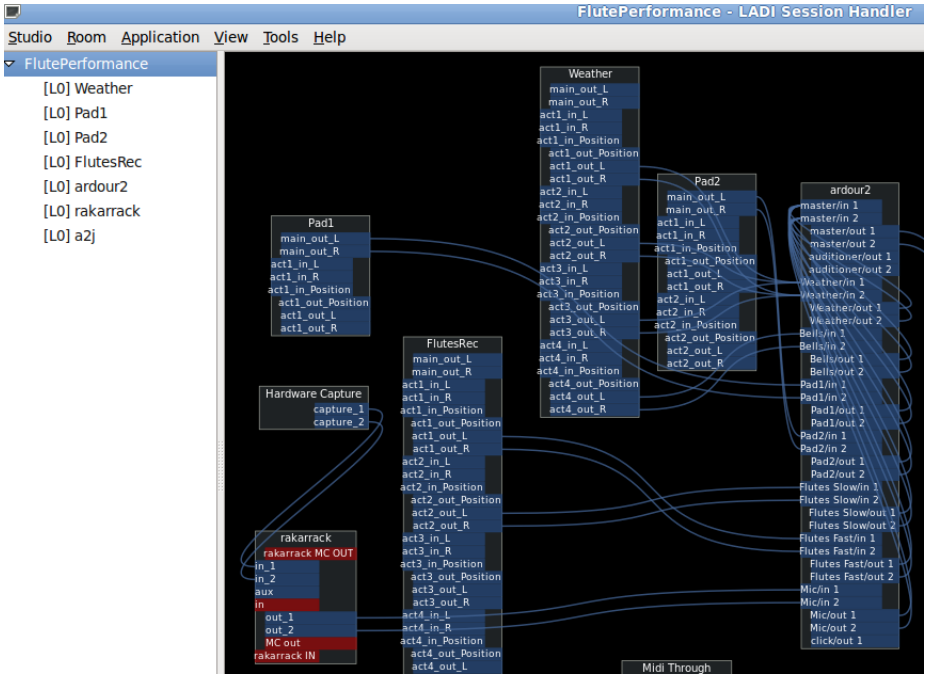
Chino

David Adler

LADISH

Linux **A**udio **D**esktop **I**ntegration **S**ession **H**andler

`http://ladish.org`



LASH's successor, incorporates both LASH and JACK Session.

Frontend:

- patchage

Non-compliant applications can be added as "Level 0-clients".

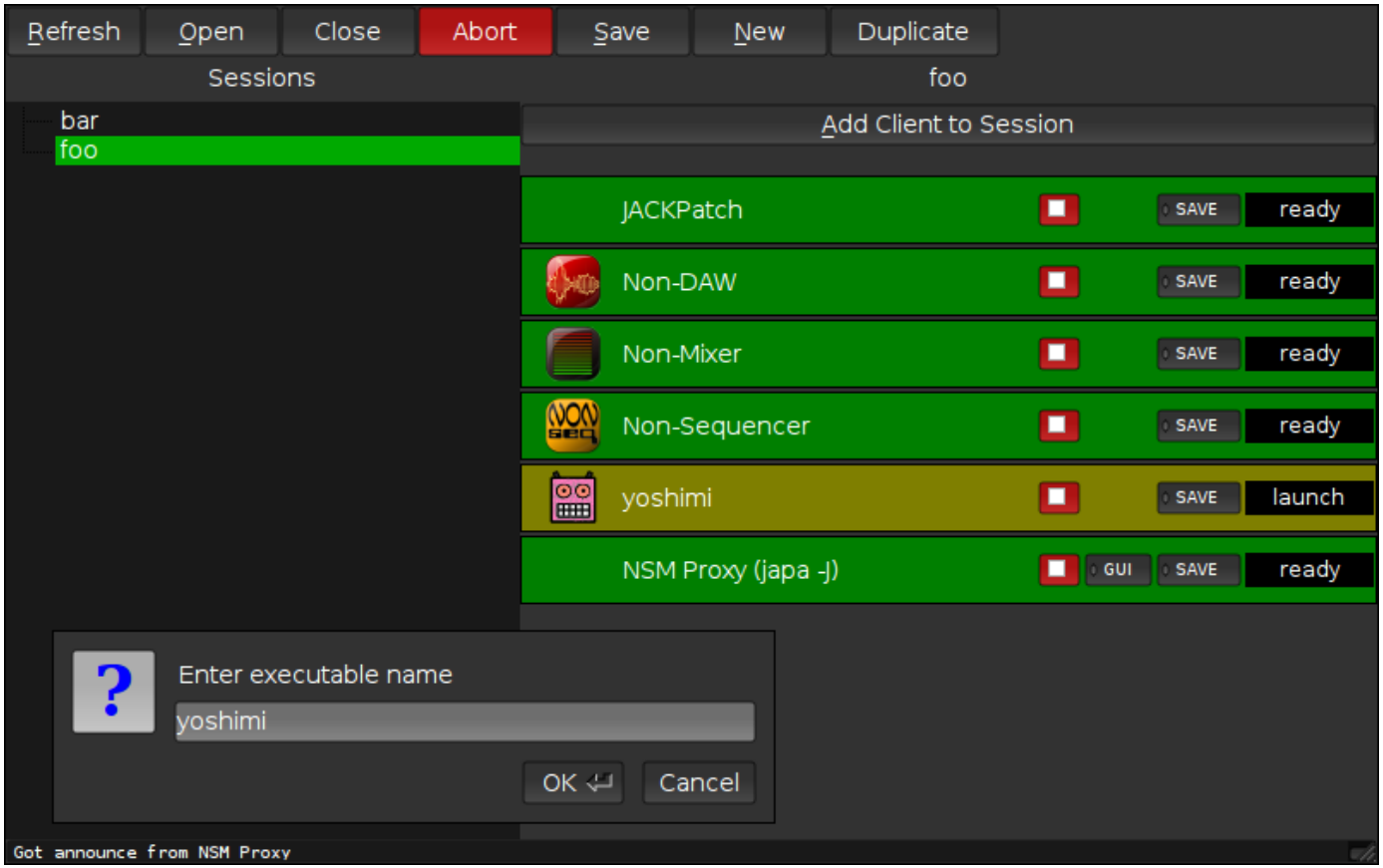
Chino

David Adler

NSM

Non Session Manager

`http://non.tuxfamily.org/wiki/Non%20Session%20Manager`



Applications not supporting the protocol can be added as “NSM Proxy-clients”.

Chino

David Adler

Chino vs. the others

	others	Chino
any connection graph	yes	no
any application	no[1]	yes[2]
central saving point	yes	no
preparation required	no	yes
GUI	yes	no

[1] LADISH, Qjackctl and NSM allow to include non-compliant applications.

[2] If the application is sufficiently scriptable.

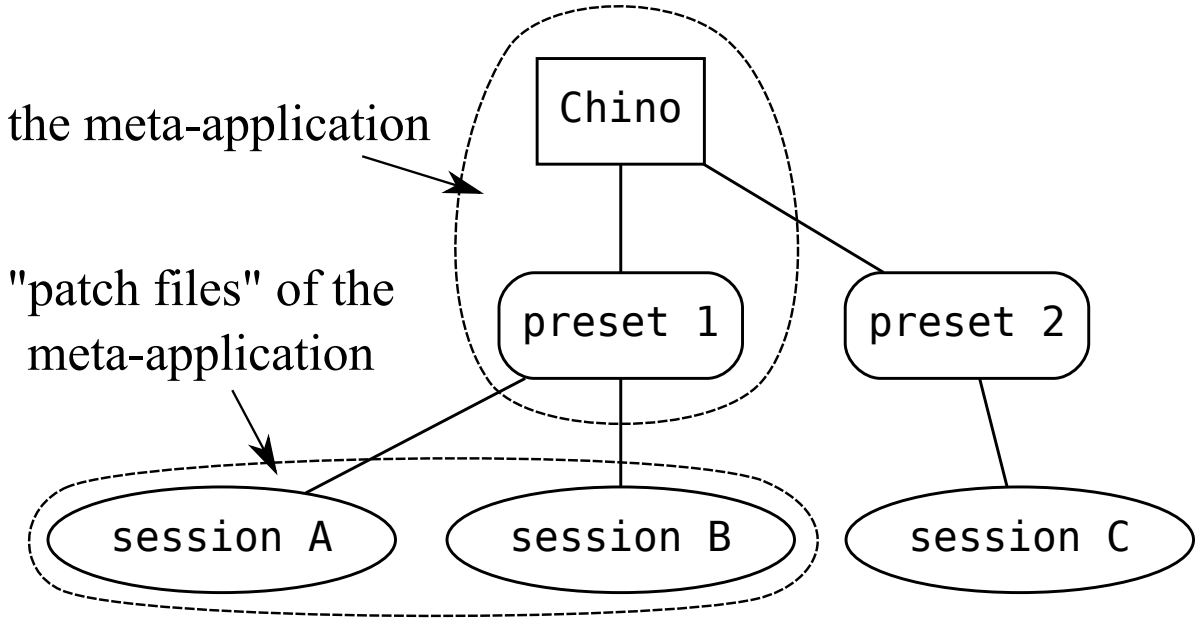
Concepts

Presets and sessions

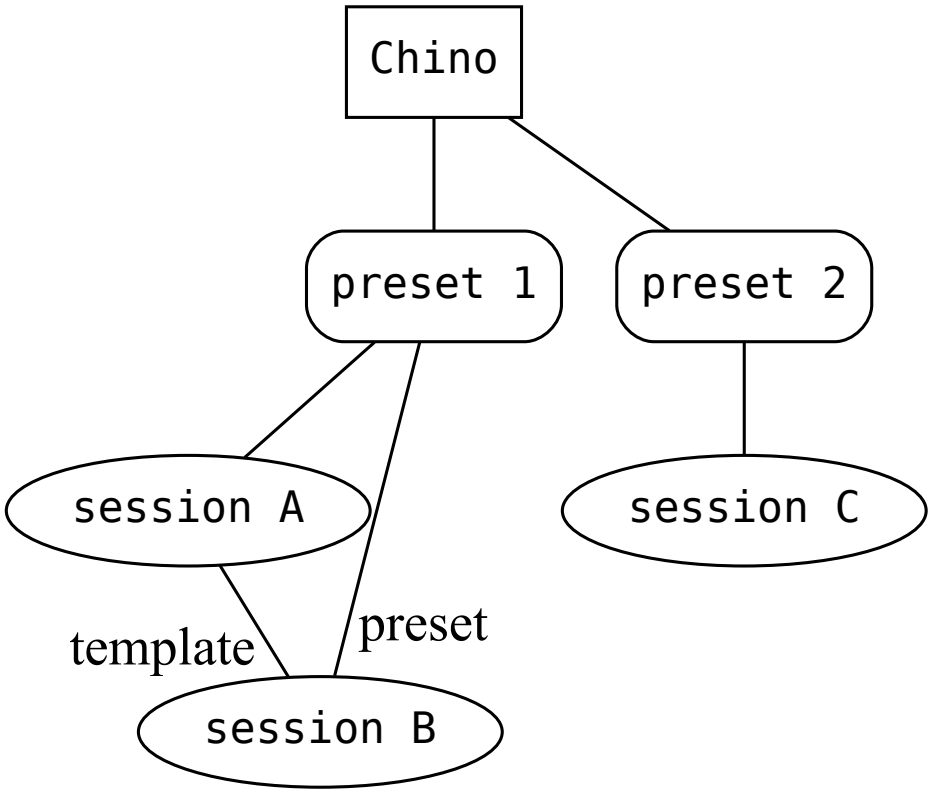
Chino does nothing,

the preset does it all,

the preset is just another session.



Templates, inheritance of files



Applications

A preset defines a number of *applications*.

An application—in Chino—consists of

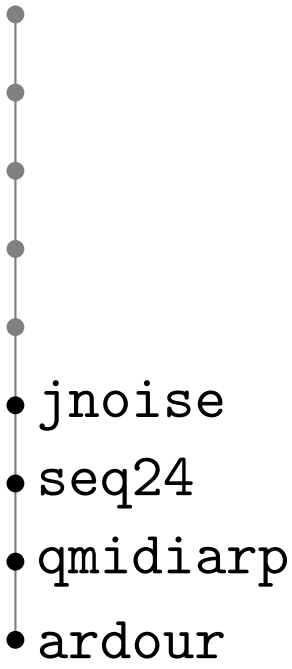
- a program
- the way it's used, defined by
  - *application files* – patch files, configuration files, dirs, etc.
  - an *application library* – a textfile (Bash) defining how the program gets started and interconnected.

Methods

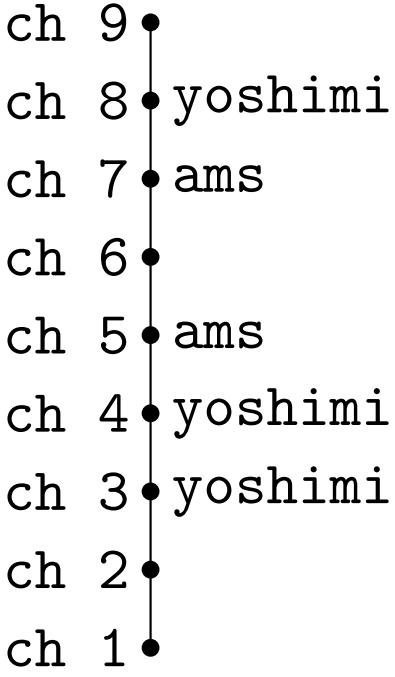
Categories of similar applications, defined via *method libraries*.

Method types:

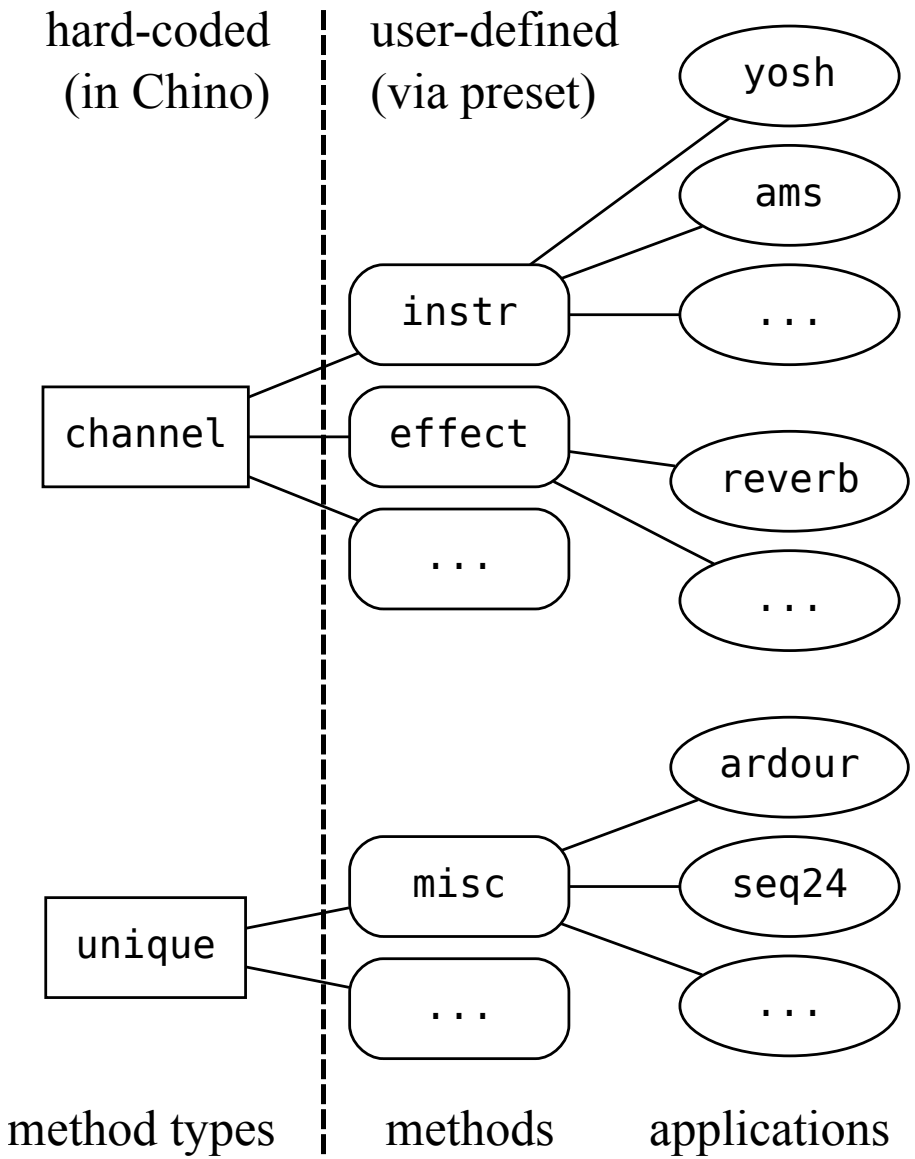
unique methods



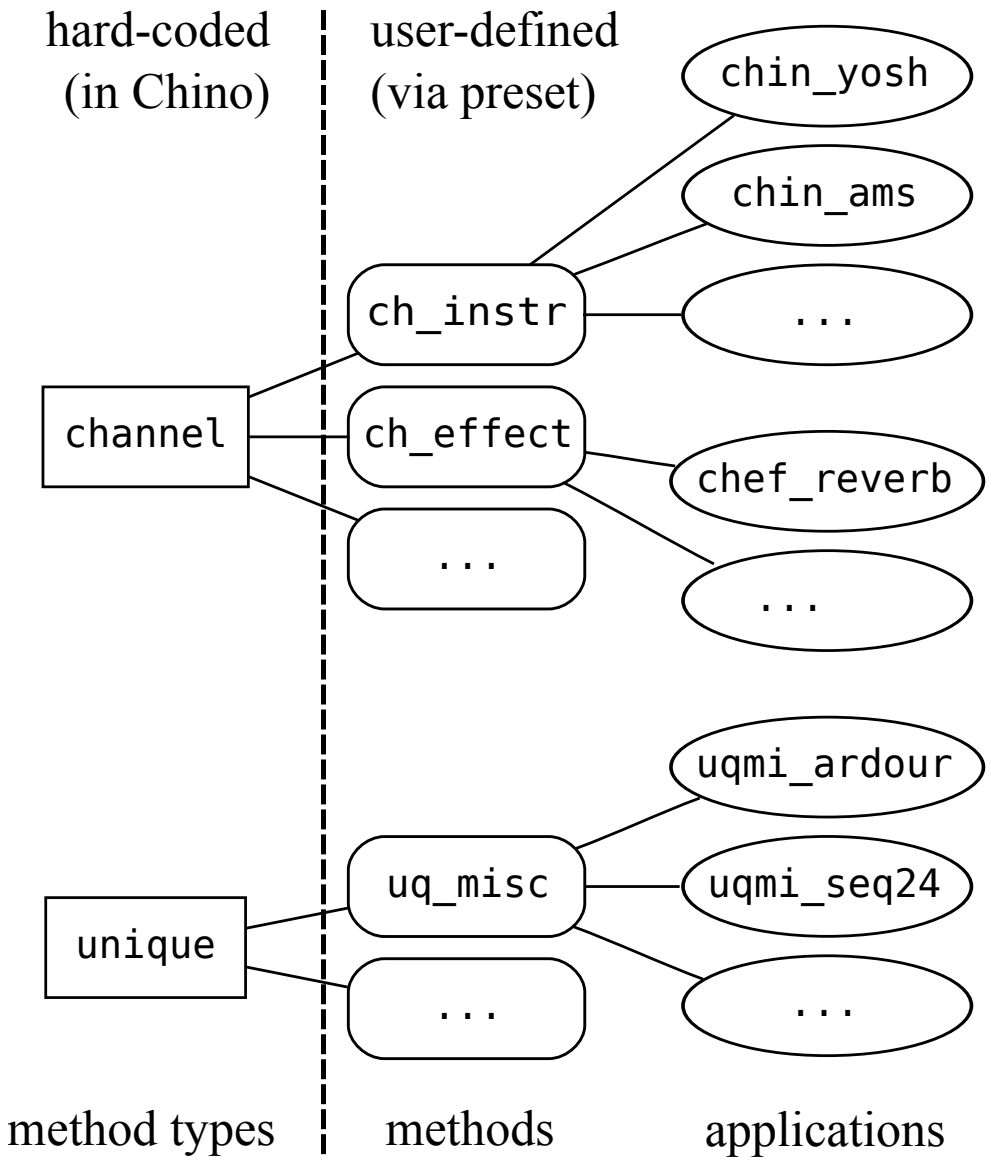
channel methods



Method types, methods and applications



Method IDs and application IDs



Application files

are kept in `<application_ID>` directories below the session's base directory.

Presets must contain all application files.

Sessions will contain the ones they (ever) require(d).

Libraries

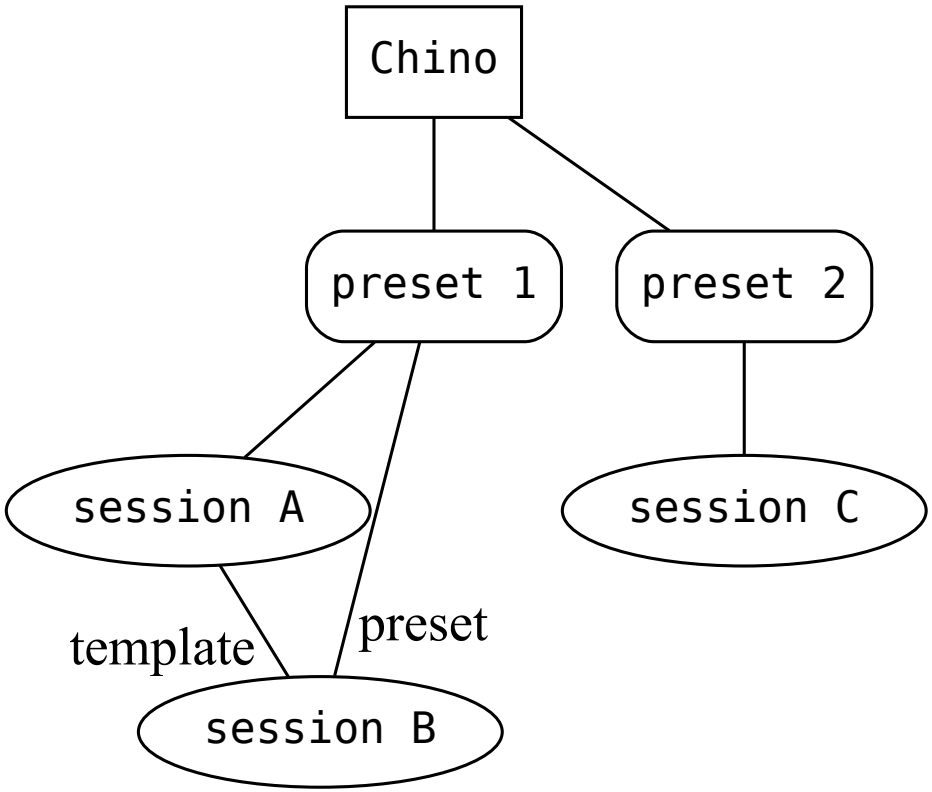
are kept in a `libs` directory below the session's base directory.

Presets must contain all libraries.

Sessions may optionally hold libraries.

A "root-library", called `<session_name>-listlib`, holds a list of all allowed methods and applications.

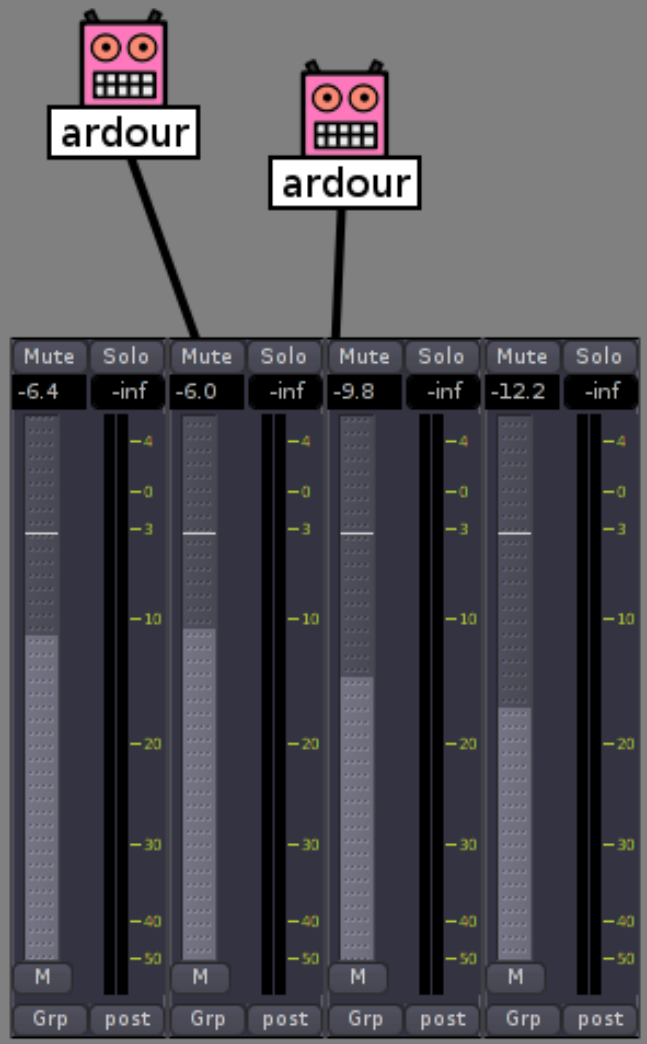
Inheritance of libraries



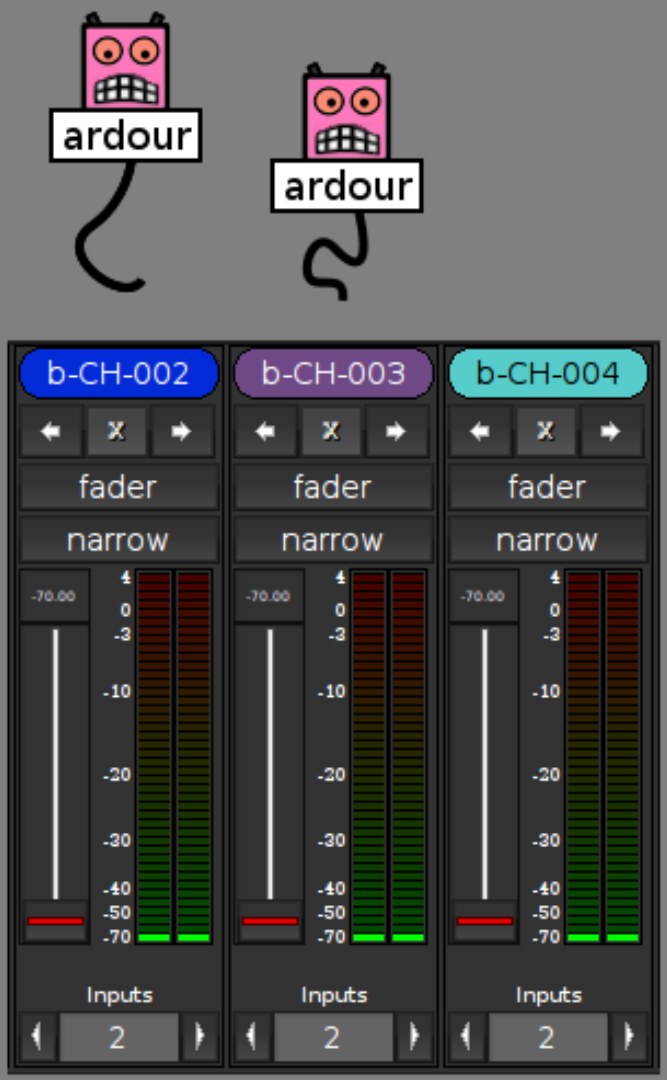
Dependencies

$$\text{program} + \text{use case} \Rightarrow \text{dependencies}$$

Dependencies



Dependencies

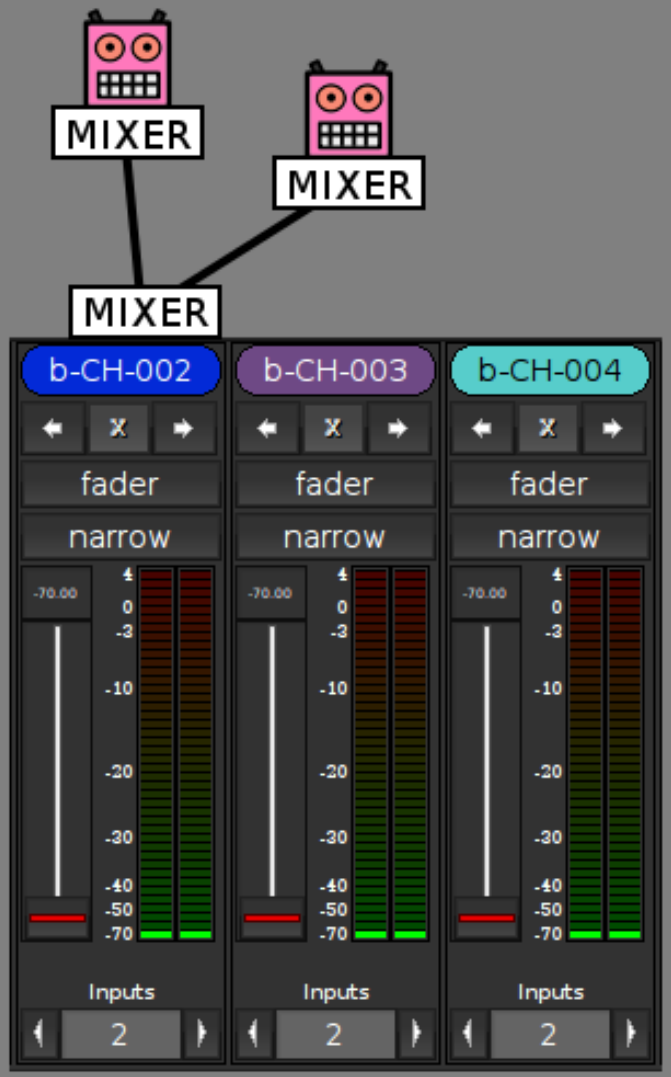


Chino

David Adler

Dependencies – port-groups

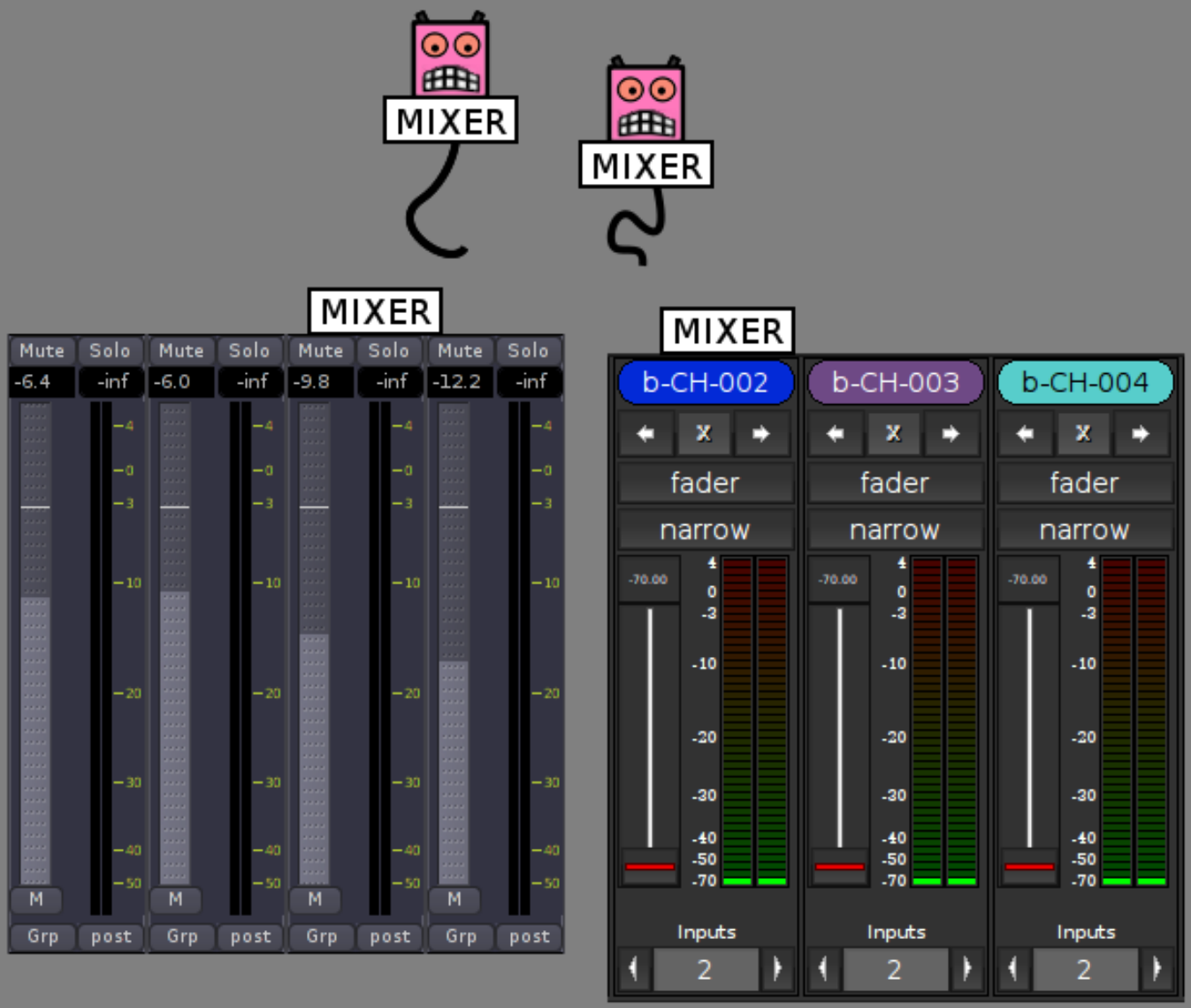
Dependencies – port-groups



Chino

David Adler

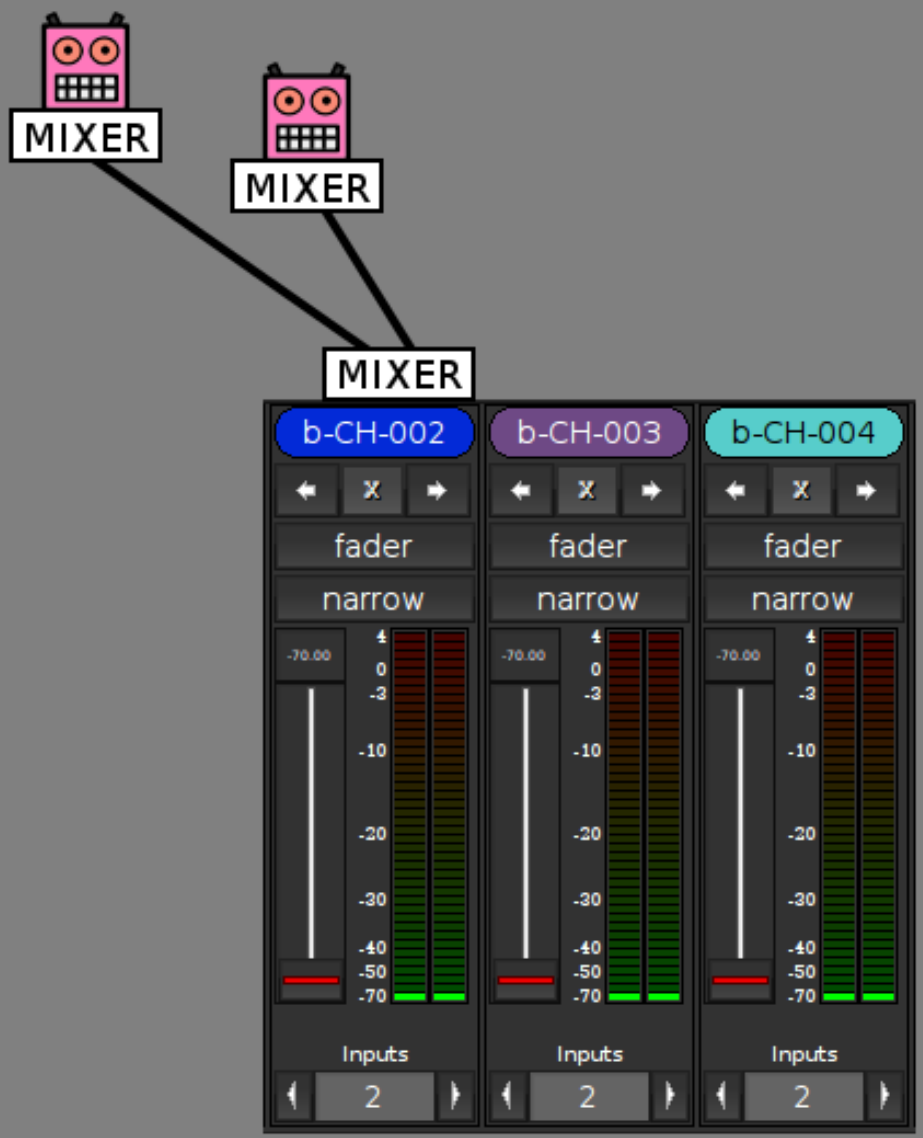
Dependencies – ambiguous provides



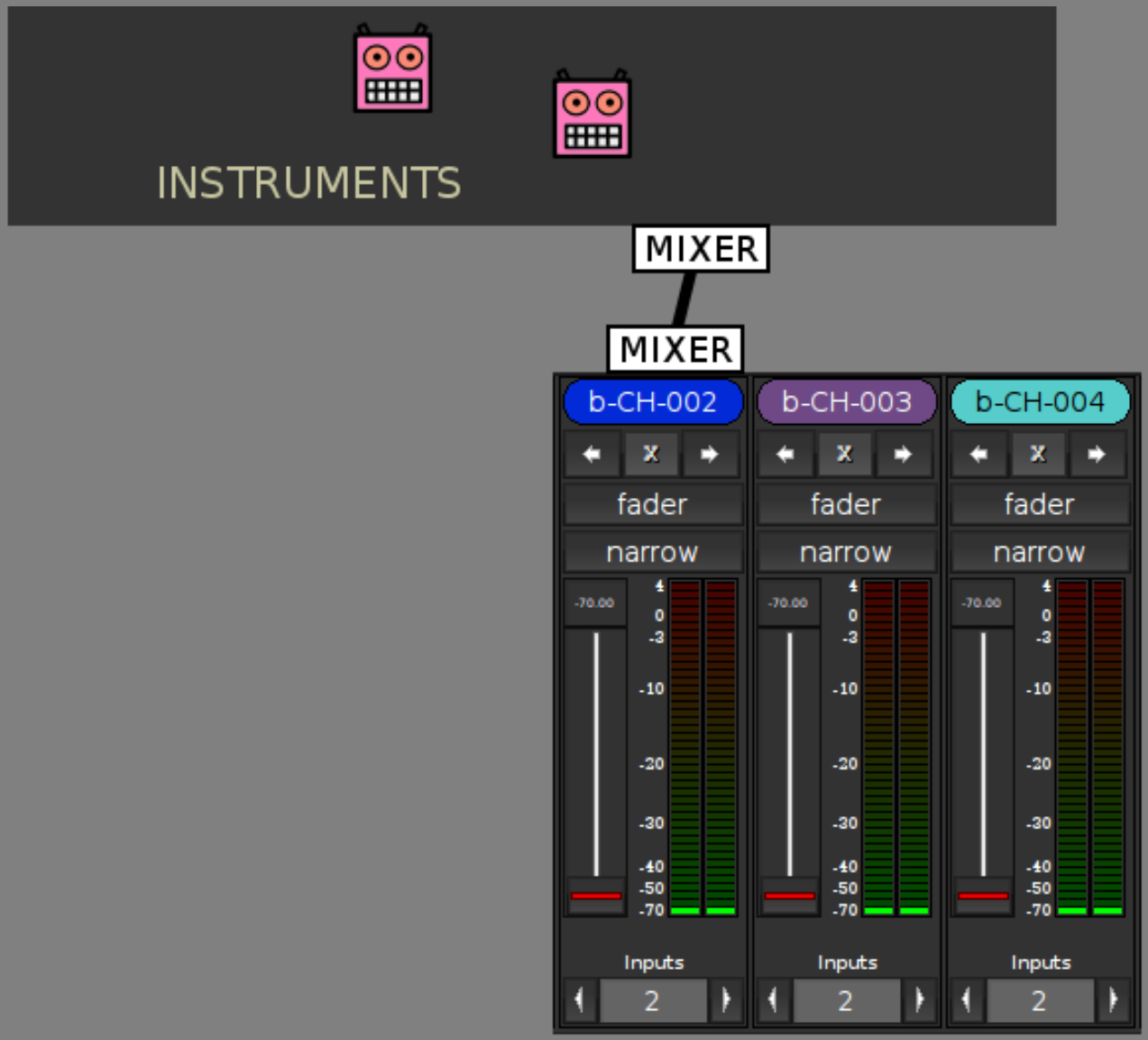
Chino

David Adler

Dependencies – ambiguous provides



Dependencies – methods

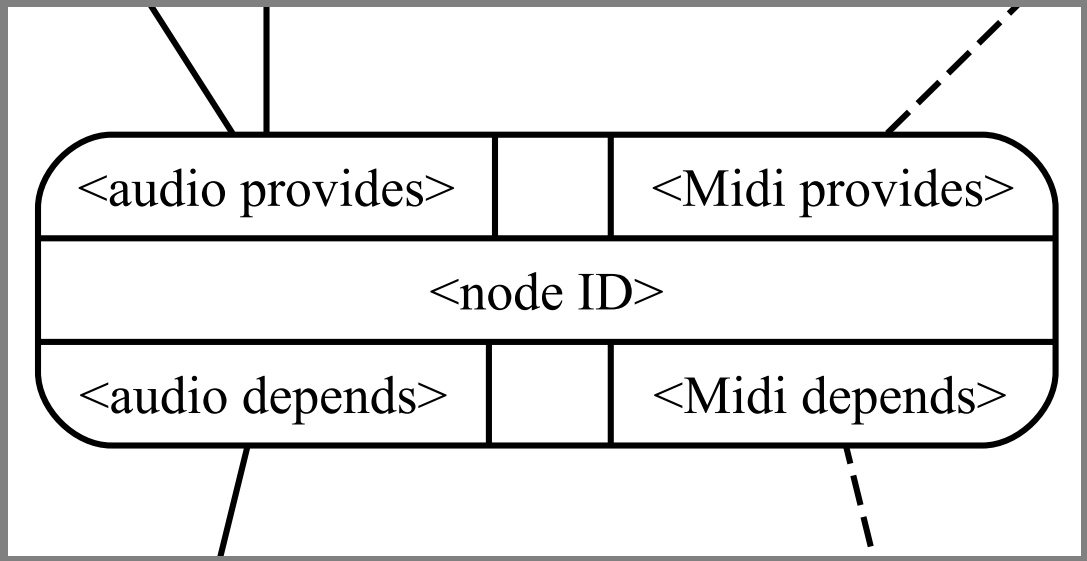


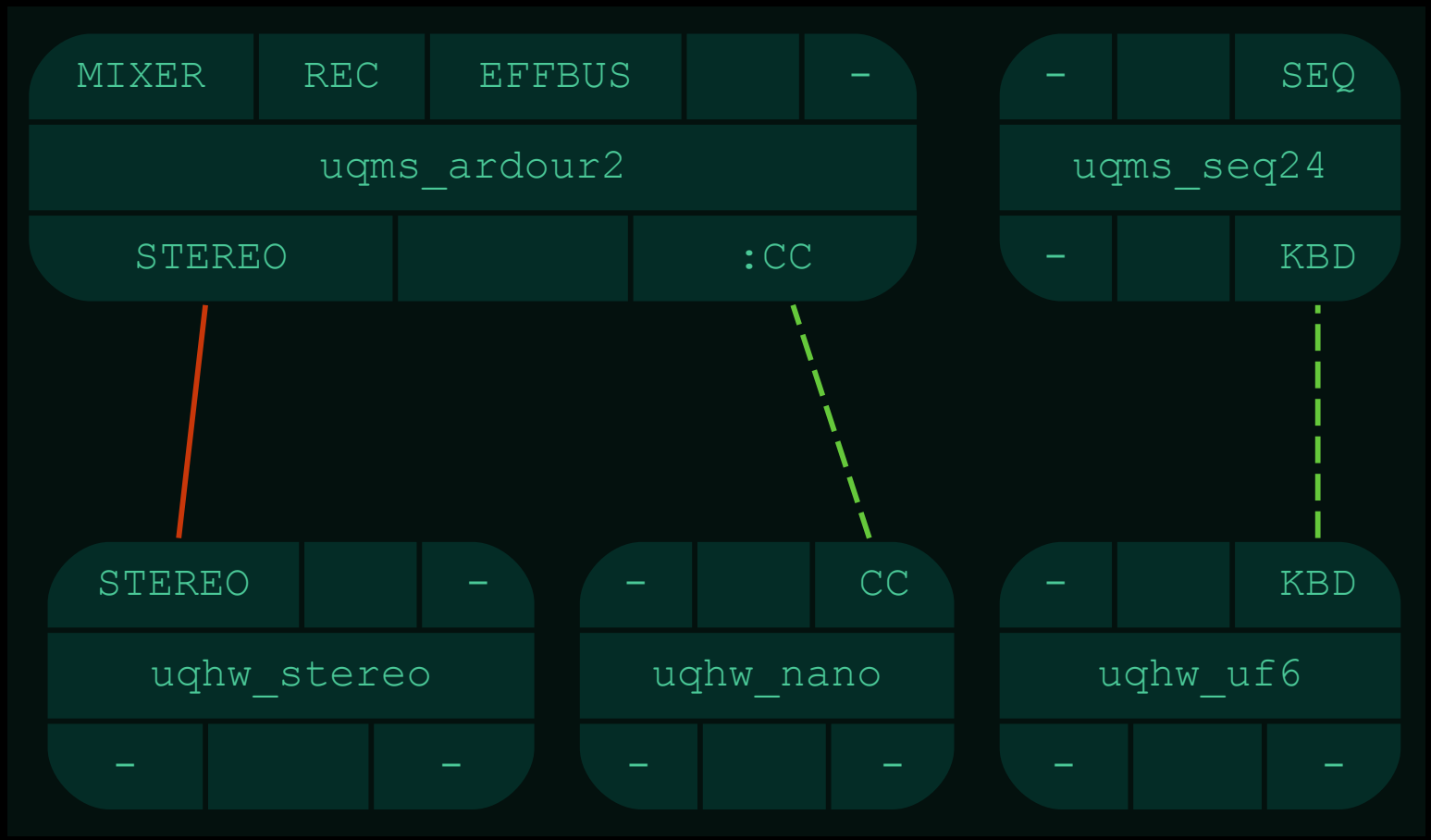
Chino

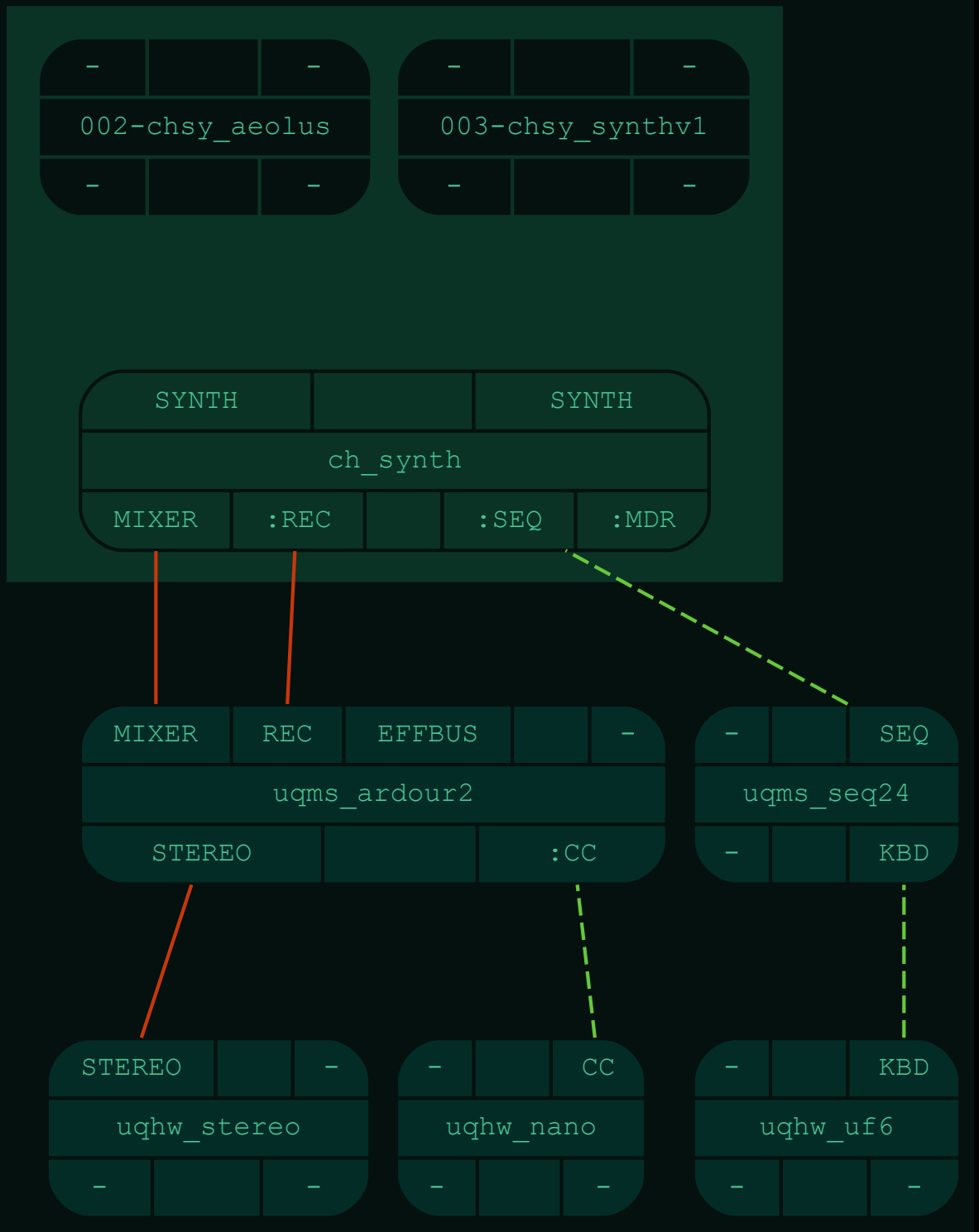
David Adler

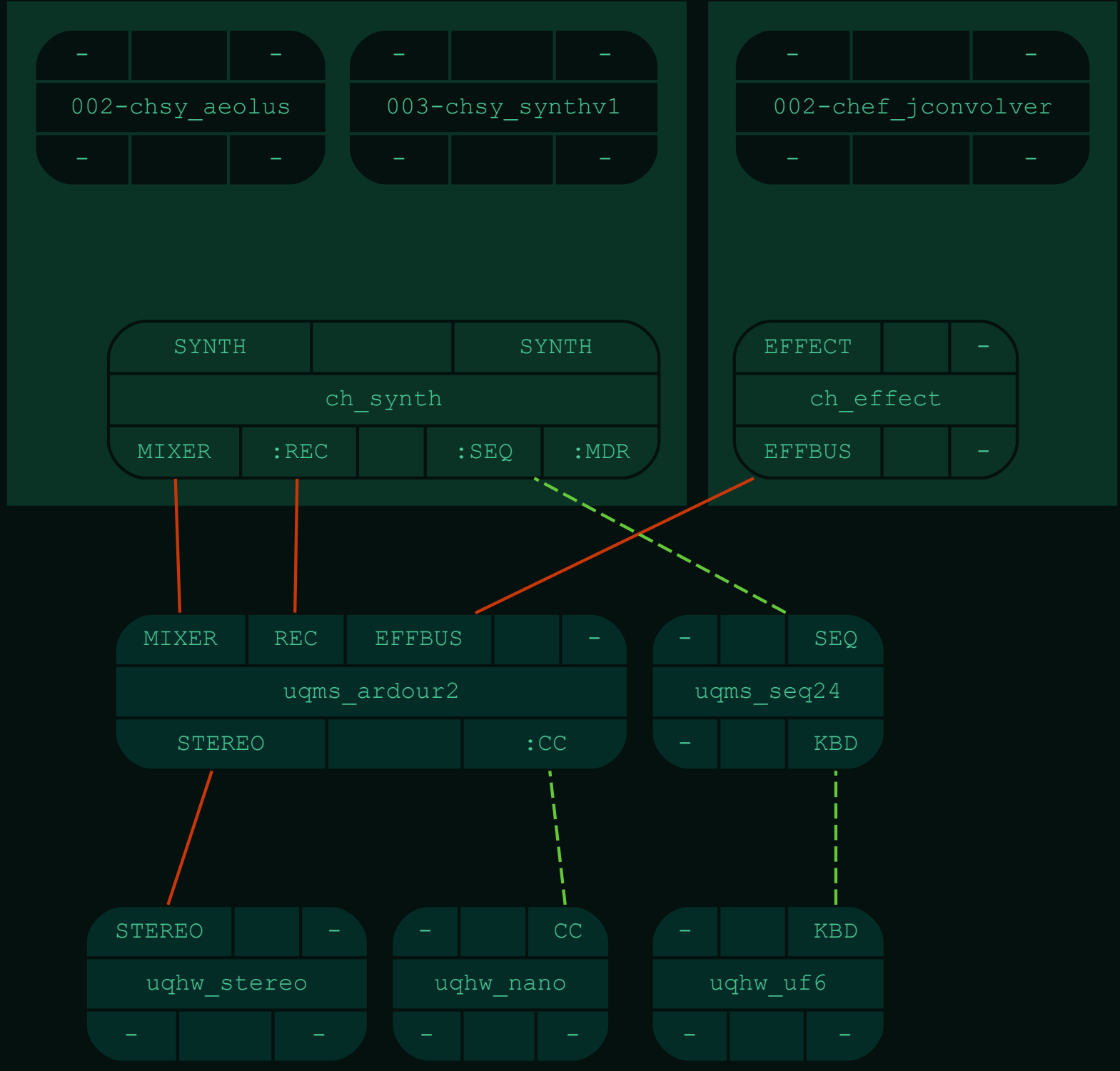
Dependencies – methods

Dependencies – nodes and anchors




MIXER
REC
EFFBUS
-
uqms_ardour2
STEREO
:CC
-
SEQ
uqms_seq24
-
KBD
STEREO
-
uqhw_stereo
-
-
-
CC
uqhw_nano
-
-
-
KBD
uqhw_uf6
-
-


-
-
002-chsy_aeolus
-
-
-
-
003-chsy_synthv1
-
-
SYNTH
SYNTH
ch_synth
MIXER
:REC
:SEQ
:MDR
MIXER
REC
EFFBUS
-
uqms_ardour2
STEREO
:CC
-
SEQ
uqms_seq24
-
KBD
STEREO
-
uqhw_stereo
-
-
-
CC
uqhw_nano
-
-
-
KBD
uqhw_uf6
-
-


002-chsy_aeolus
003-chsy_synthv1
002-chef_jconvolver
SYNTH
SYNTH
ch_synth
MIXER
:REC
:SEQ
:MDR
EFFECT
ch_effect
EFFBUS
MIXER
REC
EFFBUS
SEQ
uqms_ardour2
uqms_seq24
STEREO
:CC
KBD
STEREO
uqhw_stereo
CC
uqhw_nano
KBD
uqhw_uf6

Implementation

Steps

- `assign`
- `check`
- `list`
- `copy`
- `start`
- `acn`
- `mcn`
- the “tweak-and-save loop” (not a step)
- `unassign` or `kill`

Tasks

A *Task* is a series of steps accomplishing something useful. . .

step \ application	**a1**	**a2**	**a3**	**a4**
`assign`				
`check`				
`list`				
`copy`				
`start`				
`acn`				
`mcn`				
`unassign`				
`kill`				

Tasks

. . . like restarting an application. . .

step \ application	**a1**	**a2**	**a3**	**a4**
`assign`			`s2`	
`check`			`s3`	
`list`			`s4`	
`copy`			`s5`	
`start`			`s6`	
`acn`			`s7`	
`mcn`			`s8`	
`unassign`			`s1`	
`kill`				

Tasks

. . . re-establishing all audio connections. . .

step \ application	**a1**	**a2**	**a3**	**a4**
`assign`				
`check`				
`list`				
`copy`				
`start`				
`acn`	`s1`	`s2`	`s3`	`s4`
`mcn`				
`unassign`				
`kill`				

Tasks

. . . or starting an entire session.

step \ application	**a1**	**a2**	**a3**	**a4**
`assign`	`s1`	`s2`	`s3`	`s4`
`check`	`s5`	`s6`	`s7`	`s8`
`list`	`s9`	`s10`	`s11`	`s12`
`copy`	`s13`	`s14`	`s15`	`s16`
`start`	`s17`	`s18`	`s19`	`s20`
`acn`	`s21`	`s22`	`s23`	`s24`
`mcn`	`s25`	`s26`	`s27`	`s28`
`unassign`				
`kill`				

method type	unique methods						channel methods						
method ID	uq_aa		uq_bb				ch_aa			ch_bb			
appl. / **step**	a1	f	a2	a3	a4	f	a5	a6	f	a7	a8	a9	f
assign													
check													
list													
copy													
start													
acn													
mcn													
unassign													
kill													

method type	unique methods						channel methods						
method ID	uq_aa		uq_bb				ch_aa			ch_bb			
step \ **appl.**	a1	f	a2	a3	a4	f	a5	a6	f	a7	a8	a9	f
assign													
check													
list													
copy													
start													
acn	s1		s2	s3	s4		s5	s6		s7	s8	s9	
		f				f			f				f
mcn													
unassign													
kill													

method type	unique methods						channel methods						
method ID	uq_aa		uq_bb				ch_aa			ch_bb			
step \ **appl.**	a1	f	a2	a3	a4	f	a5	a6	f	a7	a8	a9	f
assign				s2									
						f							
check				s3									
						f							
list				s4									
						f							
copy				s5									
						f							
start				s6									
						f							
acn				s7									
						f							
mcn				s8									
						f							
unassign				s1									
						f							
kill													

Step functions

Chino executes a step by calling a corresponding function (“*step function*”) from the method library.

Step functions

```
# Step-functions in a method library.
#
# Naming:
# s_<method ID>_<step name>

# the assign step
s_ch_synth_assign()
{
    # some stuff
}

# the check step
s_ch_synth_check()
{
    # some stuff
}
```

Steps – helper functions

Inside a method's step function, usually a *helper function* is called that will accomplish the step in a standardised way.

Steps – usage of helper functions

```
# Helper-functions, prefixed h_, called by
# the corresponding step-functions.
#
# Naming:
# h_<step name>

s_uq_msc_check()
{
    declare -ri i=$1
    COPY_uq_msc[i]="$(h_check uq_msc $i)"
    echo "==    COPY_uq_msc[$i]  ${COPY_uq_msc[i]}"
}

s_uq_msc_list()
{
    declare -ri i=$1
    h_list uq_msc $i
}
```

h_assign() and h_copy()

h_assign() sources the application library, thereby implementing inheritance rules for libraries.

h_copy() copies the application files, thereby implementing inheritance rules for application files.

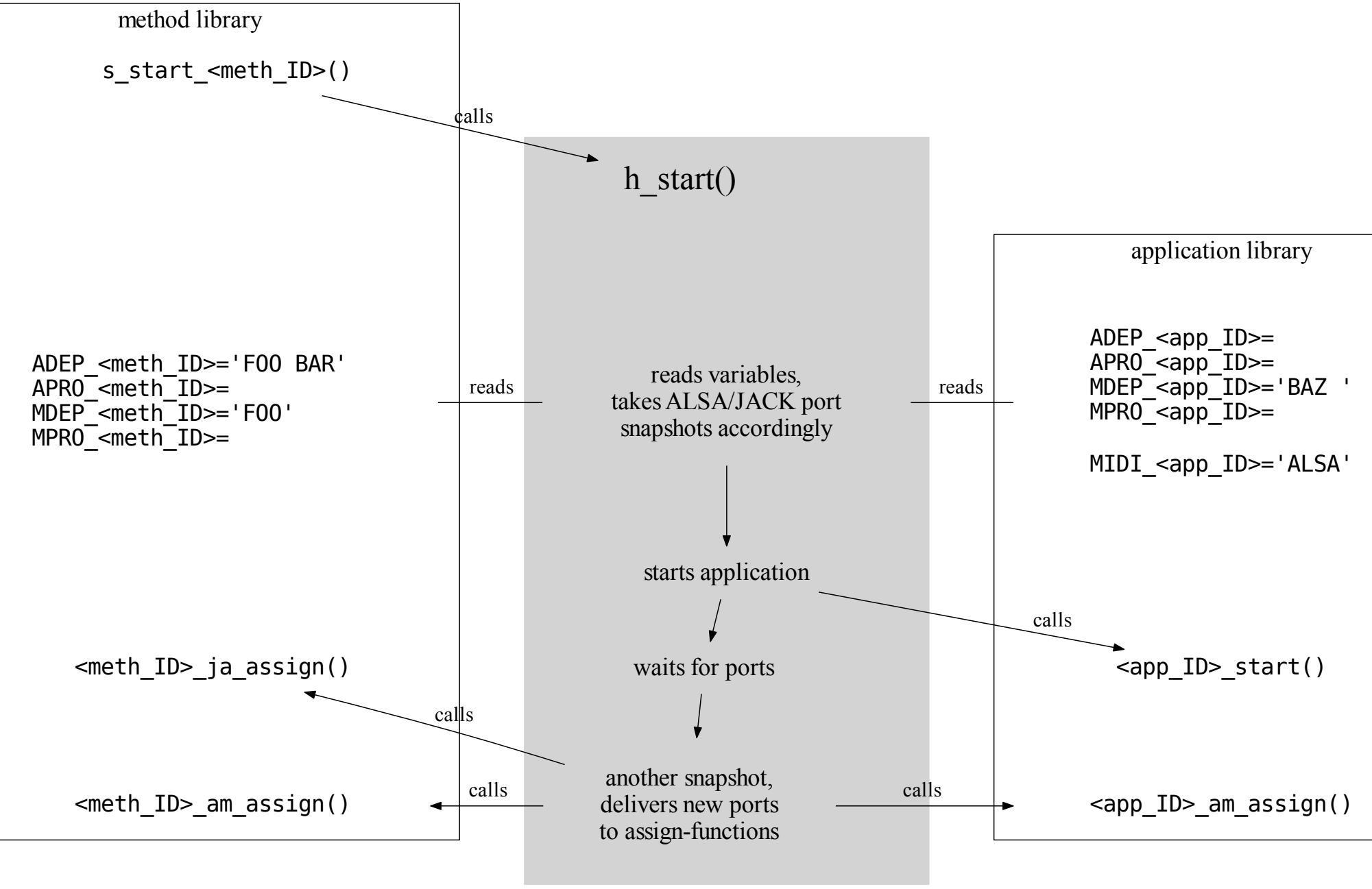
If the application requires it, h_copy() will call <application_ID>_move().

h_start()

h_start() calls:

- a function <application_ID>_start() from the application library to start the program;
- *Assign functions* in which we must assign ports to variables, in accordance with the method's and application's depends and provides.

h_start()


method library
s_start_<meth_ID>()
calls
h_start()
application library
ADEP_<app_ID>=
APRO_<app_ID>=
MDEP_<app_ID>='BAZ '
MPRO_<app_ID>=
MIDI_<app_ID>='ALSA'
ADEP_<meth_ID>='FOO BAR'
APRO_<meth_ID>=
MDEP_<meth_ID>='FOO'
MPRO_<meth_ID>=
reads
reads variables,
takes ALSA/JACK port
snapshots accordingly
reads
starts application
calls
<app_ID>_start()
waits for ports
<meth_ID>_ja_assign()
calls
another snapshot,
delivers new ports
to assign-functions
calls
<meth_ID>_am_assign()
calls
<app_ID>_am_assign()

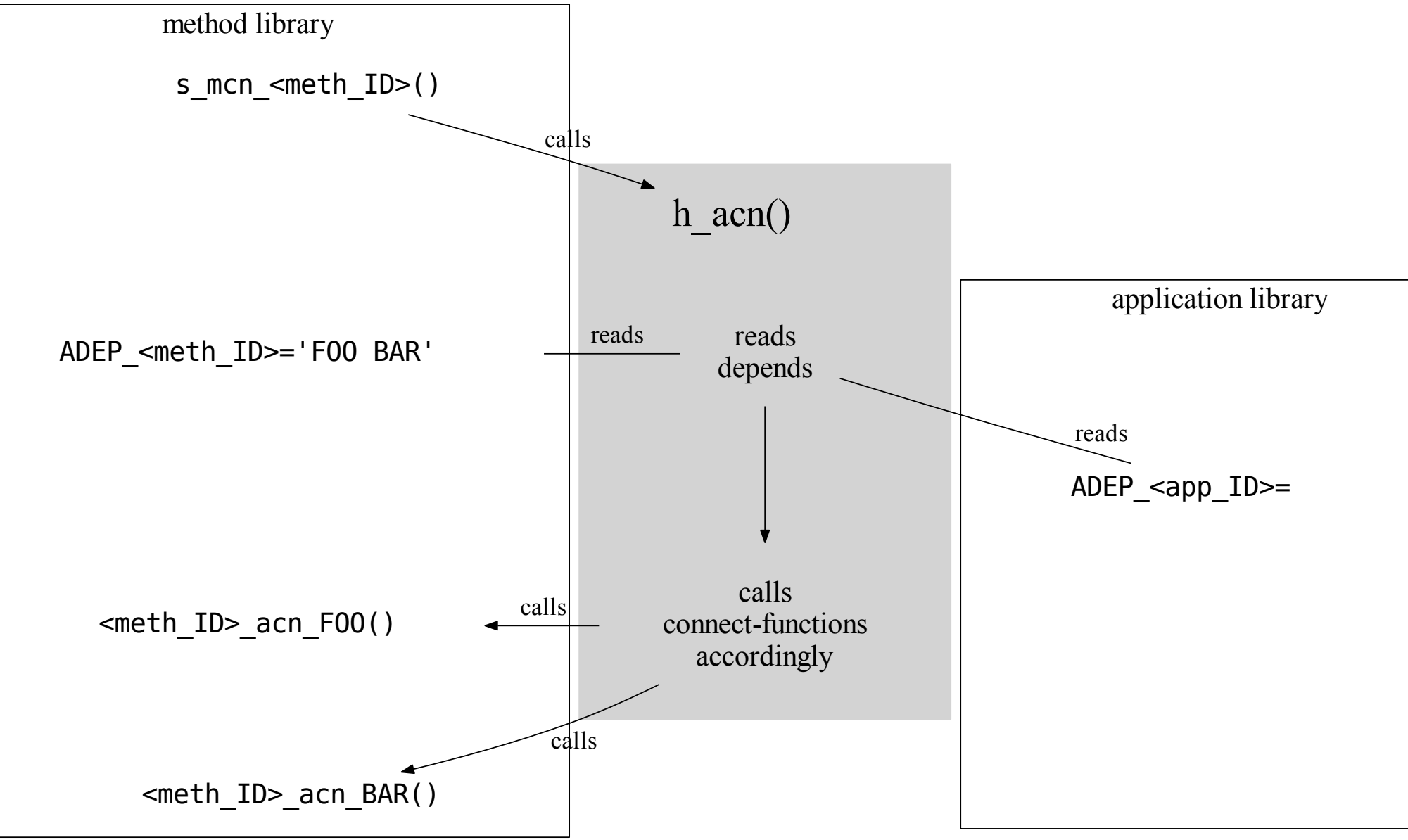
h_start() – assign-functions

```
APRO_CH_effect='EFFECT'
ADEP_CH_effect='EFFBUS'

# The method ch_effect has audio depends/provides,
# so an assign-function for audio is required.
ch_effect_ja_assign()
{
    declare -ri chan=$1
    declare -r  inports=$2
    declare -r  outports=$3

    EFFECT_AIN_L[chan]=$(echo "$inports" | sed -n 1p)
    EFFECT_AIN_R[chan]=$(echo "$inports" | sed -n 2p)
    EFFECT_AOUT_L[chan]=$(echo "$outports" | sed -n 1p)
    EFFECT_AOUT_R[chan]=$(echo "$outports" | sed -n 2p)
    echo "==    EFFECT_AIN_L[$chan]  ${EFFECT_AIN_L[chan]}"
    echo "==    EFFECT_AIN_R[$chan]  ${EFFECT_AIN_R[chan]}"
    echo "==    EFFECT_AOUT_L[$chan]  ${EFFECT_AOUT_L[chan]}"
    echo "==    EFFECT_AOUT_R[$chan]  ${EFFECT_AOUT_R[chan]}"
}
```

`h_acn()` and `h_mcn()`

`h_acn()` and `h_mcn()` call connect-functions in which we must establish connections, one for each of the method's and application's depends.

h_acn()

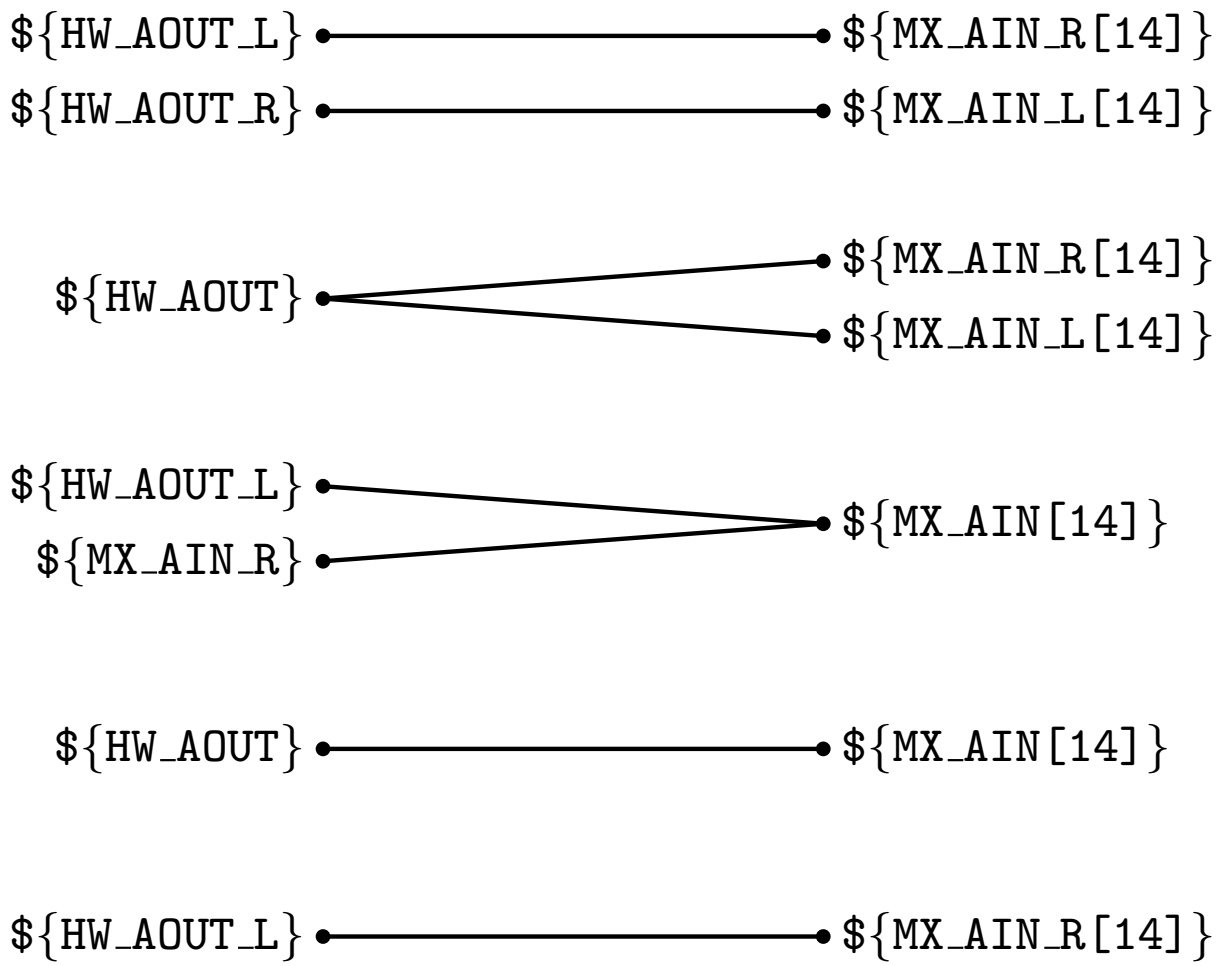

method library
s_mcn_<meth_ID>()
calls
h_acn()
application library
ADEP_<meth_ID>='FOO BAR'
reads
reads
depends
reads
ADEP_<app_ID>=
calls
connect-functions
accordingly
calls
<meth_ID>_acn_FOO()
calls
<meth_ID>_acn_BAR()

h_acn() – connect-functions

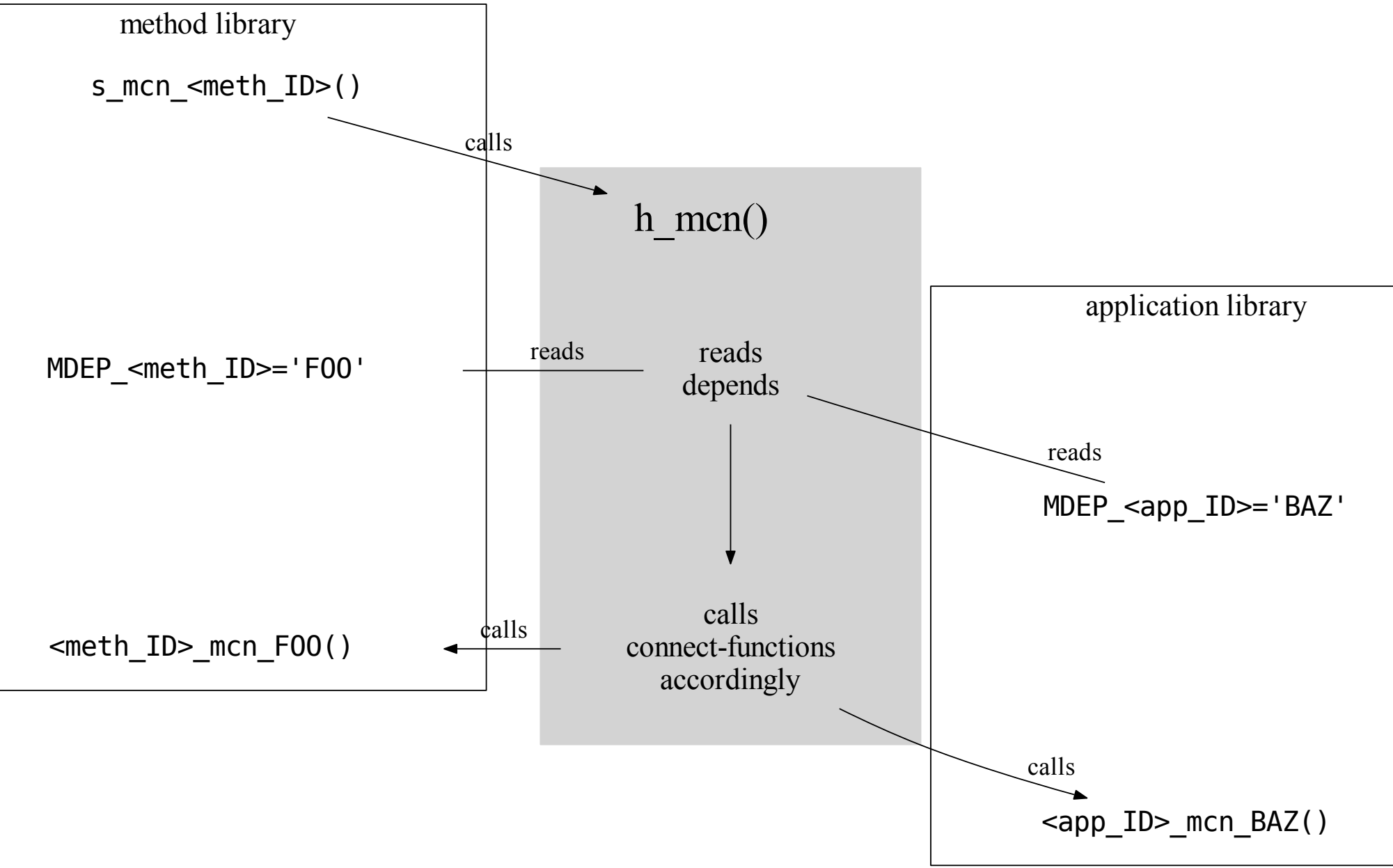
```
ADEP_ch_effect='EFFBUS'

# The method ch_effect depends on EFFBUS, so a
# corresponding audio connect-function required.
ch_effect_acn_EFFBUS()
{
    declare -ri chan=$1
    # inputs from send ports
    declare -i i=0
    while [ $i -lt $MIXER_NCHAN ] ; do
        msaudioconnect EFFBUS_SEND "$chan,$i" EFFECT_AIN $chan
        (( i++ ))
    done
    # outputs to return ports
    msaudioconnect EFFECT_AOUT $chan EFFBUS_RETURN $chan
}
```

`msaudioconnect HW_AOUT 0 MX_AIN 14`

will, depending on existance of port-variables, connect:



h_mcn()

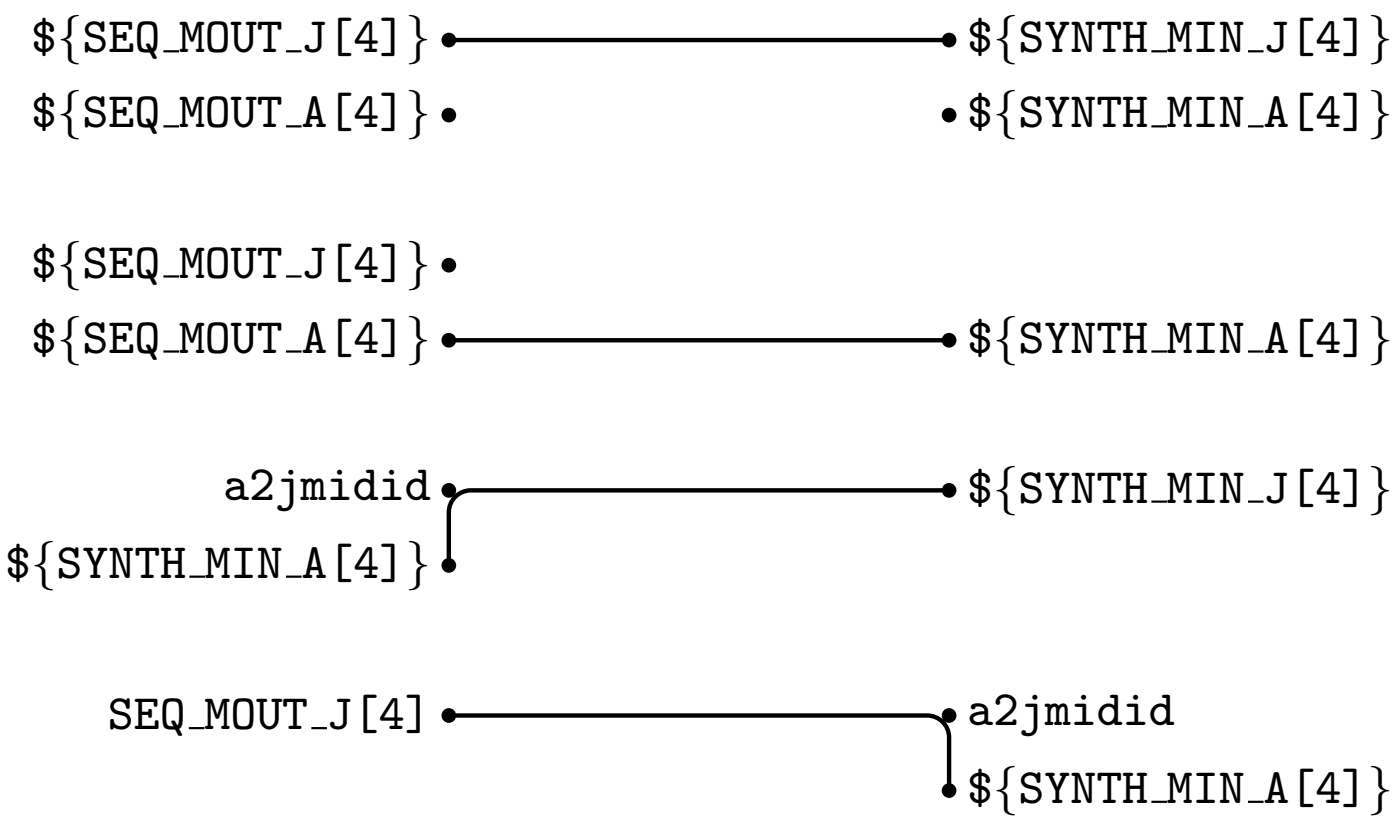

method library
s_mcn_<meth_ID>()
calls
h_mcn()
application library
MDEP_<meth_ID>='FOO'
reads
reads
depends
reads
MDEP_<app_ID>='BAZ'
<meth_ID>_mcn_FOO()
calls
calls
connect-functions
accordingly
calls
<app_ID>_mcn_BAZ()

h_mcn() – connect-functions

```
MDEP_chef_jackrack='CC'

# The application chef_jackrack depends on CC, so a
# corresponding Midi connect-function is required.
chef_jackrack_mcn_CC()
{
    declare -ri chan=$1
    ajmidiconnect  CC_MOUT 0 chef_jackrack_MIN $chan
}
```

`ajmidiconnect SEQ_MOUT 4 SYNTH_MIN 4`

will, depending on existance of port-variables, connect:



Keeping the graph intact

Connect-functions aren't exclusively called by `h_acn()` and `h_mcn()`.

E.g. the task for starting an application will call connect-functions of nodes depending on any newly provided port-group.

User interface

- command line options and arguments
- runtime user interface
- the session definition file

User interface – command line

```
[foo@bar ~]$ chino -h

chino -h
chino -v
chino -w [-p <ppath>:<pname> -t <tpath>:<tname]
chino -m [-p <ppath>:<pname> -t <tpath>:<tname]
chino -a [-p <ppath>:<pname> -t <tpath>:<tname]
chino -n name [-p <ppath>:<pname> -t <tpath>:<tname>]
chino -o sdef

  -h                    display this help text
  -v                    print version number
  -w                    write session definition file prototype
  -m                    create new method library
  -a                    create new application library
  -n <name>             start new session with name <name>
  -o <sdef>             open session using session definition file <sdef>
  -p <ppath>:<pname>    Use preset <pname> in directory <ppath> as preset,
                        overriding the default from ~/.chinorc.
  -t <tpath>:<tname>    Use session <tname> in directory <tpath> as template.
                        If not specified, the preset will serve as template.
```

User interface - at runtime

```
      M  -  add a method
      A  -  add an application
      X  -  remove an application
      R  -  restart an application

      .  -  localise application/method library
      :  -  localise all used libraries
      f  -  force sourcing of application/method library

      l  -  list current session
      d  -  check dependency tree
      g  -  toggle session graph display
      w  -  write changes to session definition file

      a  -  redo audio connections
      m  -  redo midi connections

      s  -  store connection snapshot
      r  -  restore connection snapshot
      u  -  un-store connection snapshot

 ctrl+c  -  quit
```

User interface - session definition file

```
NAME=opus2
PRESET=/usr/share/chino/presets/default:default
TMPL=

# methods and applications
UQMETHS=hw msc meter
uq_hw=stereo
uq_msc=ardour2 vkeybd nonseq
uq_meter=jkmeter japa

CHMETHS=synth
ch_synth-CH-001=yoshimi
ch_synth-CH-002=
ch_synth-CH-003=
ch_synth-CH-004=
ch_synth-CH-005=
ch_synth-CH-006=
ch_synth-CH-007=
ch_synth-CH-008=
```

Miscellaneous

Over the network? `ssh`!

Portability

- self-contained or shared preset
- program versions
- hardware requirements
- sampling rates
- local configuration files
- external (audio)files

The Future

- application groups
- guided fixing of unresolved depends
- have a preset determine window placement for applications—via scriptable WM
- text-only sessions in git

Thanks to

- LAD, LAU and the wider Linux audio community, for all that software and the reading matter they provide.
- `http://tuxfamily.org` for their friendly no-BS hosting.

Demo

Q&A

http://chino.tuxfamily.org

david.jo.adler@gmail.com