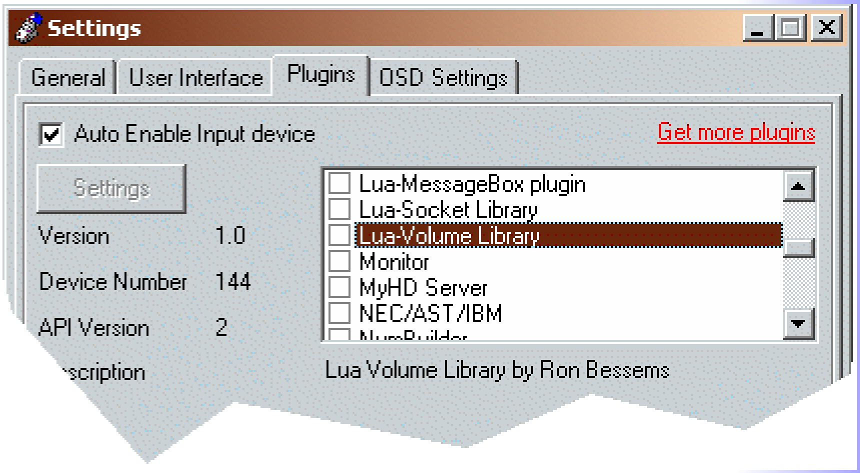
Lua Volume Controls Girder Plugin

Installation

This plugin comes installed in the default Girder distribution.

To use this plugin you'll have to enable it. You'll find the plugin listed on the configuration window. Scroll down in the list to find the Lua-Volume Library. To enable it tick the checkbox and press ok. Now enable the input devices in the main menu.



☑ Auto Enable Input device

Version	1.0
Device Number	144
API Version	2

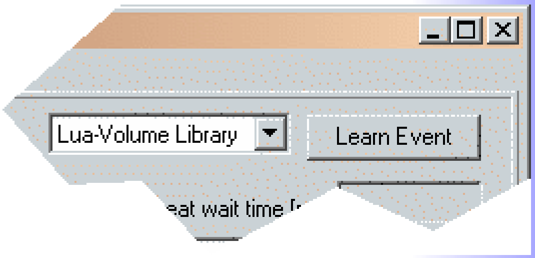
☐ Lua-MessageBox plugin
☐ Lua-Socket Library
☐ Lua-Volume Library
☐ Monitor
☐ MyHD Server
☐ NEC/AST/IBM

scription Lua Volume Library by Ron Bessems

Testing

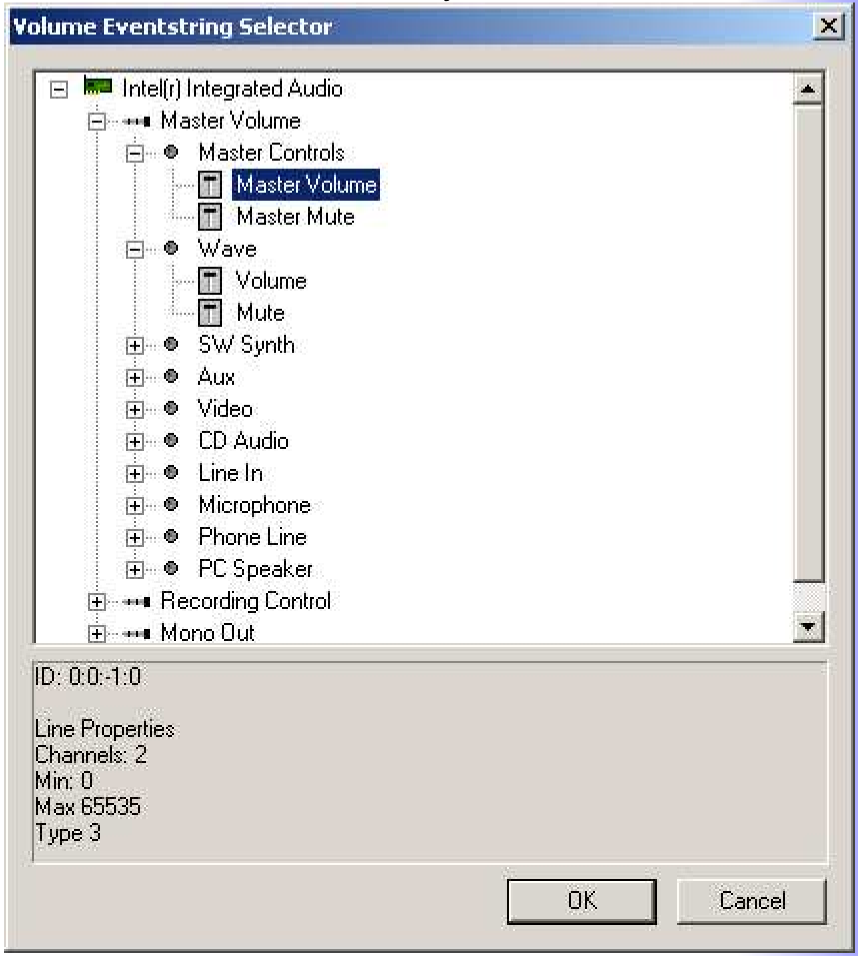
To test if the plugin is working enable the input devices (F9), make sure the green light at the bottom of the main window is green. Now open the Windows volume control or mixer. Slide the master volume control up and down. Girder's green light should now flash to yellow, signaling that events are coming in and that the plugin is working!

The second thing to test is to see if the plugin recognizes all your volume controls. To do this add a command to the Girder window (Control-A). Select the Lua-Volume Library from the Learn even dropdown box and press learn event. The window that now appears should have all your audio lines listed.



Lua Interface

The plugin has several functions that allow manipulation of the audio lines. First the general concept will be described. The way this library access the audio controls is through numbers. The first sound card in your system has number 0, the second has number 1 and so forth.
Each sound card has several *destinations*, a destination is for example the master mixer or the recording control. Each destination has several *sources* these typically are the master control, the wave source, the cd audio and the line in. These *sources* again have *controls*. Think of *controls* as the volume button on your stereo, in the picture to the right you can see a *control* highlighted (Master Volume).



ID: 0:0:-1:0

Line Properties
Channels: 2
Min: 0
Max 65535
Type 3

Exploring your sound devices

To find out which *controls* are available we first need to know how many there are, Return values of all function will be zero or positive for success and negative for failure.

(number **count**) GetDeviceCount().

This function will tell you exactly how many sound cards (*devices*) are installed on your computer. Now that we know how many *devices* there are we need some way to access them. This is simply done by a number, if there are 2 cards in your system they will have number 0 and 1.

It is possible to find out what the name is of each object that we work with, so for a device the function is called

(*string* **name**) GetDeviceName(*number* **Device**)

After we are done with the device the **handle** can be closed so we don't hog the windows resources.

CloseVolumeControls(*number* **Device**)

Putting it all together,

```
name,err = GetDeviceName(0)

if ( err<0 ) then

  print("An Error occurd while opening the sound card")

else


  print(name)

  CloseVolumeControls(0) -- optional not advised if you are using
the mixer a lot.

end
```

Example 1

Addressing

Addressing of the controls is done by numbers, as said the first sound card has number 0, the second has number 1, etc.

So for the *device* numbers we get the following rule

No smaller then 0 and no bigger than GetDeviceCount()-1

For *Destinations,*

No smaller then 0 and no bigger than GetDestinationCount()-1

For *Sources,*

No smaller then –1 and no bigger than GetSourceCount()-1

For *Controls,*

No smaller then 0 and no bigger than GetControlCount()-1

For *Channels,*

No smaller then –1 and no bigger then GetChannelCount()-1

Note 1: *Sources* allows the index to be –1, this is a special case an always retrieves the master volume source.

Note 2: *Channels* allows the index to be –1, this is a convenience, say a control has 2 channels but you wanted to set both at the same time, normally you would have to give 2 commands, however if you address the *channel* as –1 the library will take care of setting all channels.

For example the volume control on the master line is indexed as:

0:0:-1:0, the mute channel is 0:0:-1:1

The volume control of the wave channel is:

0:0:0:0, the mute channel is 0:0:0:1

Hint! You can find these numbers on the event picker window. The volume control is usually BUT not always number 1.

Enumerating all sound cards

We now have all tools needed to enumerate all soundcards, a quick example is always good so here we go.

```
DevCnt = GetDeviceCount()

for i=0, DevCnt-1 do

  name,err = GetDeviceName(i)

  if ( err < 0 ) then

    print ("Error accesing sound card:"..i)

  else


    print ("Card "..i.." has name: "..n)

    CloseVolumeControls(i)

  end


end
```

Example 2

Enumerating all *Destinations*

Now for the first example of addressing. We are going to list all *destinations* in your soundcard.

```
DestCnt = GetDestinationCount(0)

if ( DestCnt <0 ) then

    print ("Error accessing sound card:"..i)

else


  for i=0,DestCnt-1 do

    name,err = GetDestinationName(0, i)

    print (name);


  end;

  CloseVolumeControls(0)

end;
```

Example 3

As you can see GetDestinationName takes as input the mixer handle and a number specifying which *Destination* you want to access.

Enumerating all *Sources* of all *Destinations*

Now expanding on the previous example we also want to list all *sources* of all these *destinations.*

```
DestCnt = GetDestinationCount(0)

if ( DestCnt <0 ) then

    print ("Error accessing sound card:"..i)

else



  for i=0,DestCnt-1 do

    Name,err = GetDestinationName(0, i)
    print (name)

    -- request sourcecount of the current destination (i)
    SrcCnt = GetSourceCount(0, i)

    -- start at -1 to include the master channel as well
    for j=-1, SrcCnt-1 do

      -- now we are accessing the name of the source, this needs
      -- two indices i and j.
      Name,err = GetSourceName(0, i,j)
      print ("  "..name) -- indented for clarity


    end


  end

  CloseVolumeControls(0)

end
```

Example 4

Enumerating all *Controls* of all *Sources* of all *Destinations*

Now expanding on the previous example we also want to list all *sources* of all these *destinations.*

```
DestCnt = GetDestinationCount(0)

if ( DestCnt <0 ) then

    print ("Error opening sound card:"..i)

else


  for i=0,DestCnt-1 do

    Name,err = GetDestinationName(0, i)
    print (name)

    -- request sourcecount of the current destination (i)
    SrcCnt = GetSourceCount(0, i)

    -- start at -1 to include the master channel as well
    for j=-1, SrcCnt-1 do

      -- now we are accessing the name of the source, this needs
      -- two indices i and j.
      Name,err = GetSourceName(0, i,j)
      print ("  "..name) -- indented for clarity

      -- get the number of controls on the current source
      CtrlCnt = GetControlCount(0, i,j)
      for k=0, CtrlCnt-1 do

        Name,err = GetControlName(0,i,j,k)
        print("    "..name)

      end


    end


  end

  CloseVolumeControls(0)

end
```

Example 5

Accessing the volume levels

Enough beating around the bush, let's get to the core of the problem. We want to modify the volume levels. To do this you'll need to know what control you want to change. This can be done by looking at the event picker window (Learn Event button).

Say we wanted to set the volume of the wave channel (id 0:0:0:0) to 50%. We need two things for that, first we have to find out the full range of the channel and secondly we need the function to change the volume.

To find the range of a control the function use

```
type, min , max = GetControlBounds(Device, Dest, Source, Control)
```

To set the value of a channel use

```
err = SetChannelValue(Device,Dest,Source,Control,Channel,Item,val)
```

```
Dest=0
Source=0
Control=0 – again, note the value of 0 is usually the volume

type, min , max = GetControlBounds(0, Dest, Source, Control)

if (  type < 0 ) then

  print("An Error Occurd while accessing the sound card")

else

  val = (max - min)/2 + min;

  -- the -1 indicates that we want to set all channels at the
same time
  -- the 0 is the Item (unused parameter)
  SetChannelValue(0, Dest,Source,Control,-1,0,val)

  CloseVolumeControls(0)

end
```

Example 6

Accessing the Volume levels (cont.)

To read the value of a channel use

```
value,err = GetChannelValue(device, Dest, Source, Control, Channel, Item)
```

If the variable **err** is less then zero there was an error. The following code decreases the wave volume by 50%.

```
Dest=0
Source=0
Control=0

type, min , max = GetControlBounds(0, Dest, Source, Control)

if ( type < 0 ) then

  print("An Error Occurd while accessing the sound card")

else

  Val,err = GetChannelValue(0, Dest, Source, Control, -1,0)

  val = val / 2;

  if ( val < min ) then
    val = min
  end

  -- the -1 indicates that we want to set all channels at the same
time
  -- the 0 is the Item (unused parameter)
  SetChannelValue(0, Dest,Source,Control,-1,0,val)

  CloseVolumeControls(0)

end
```

Example 7

Event Generating

The plugin by default sends events from all *devices* to Girder. You can change this by unchecking “Automatically register for all soundcard volume events” in the settings dialog and then using these Lua functions:

```
number error RegisterForVolumeEvents( number device)
```

If **error** is negative there was an error, for example the device does not exist. If the value is 0 the operation was successful. If error is 1 then the devices was already registered for sending events, nothing changed.

If you want to stop receiving events from some device use,

```
number error UnregisterForVolumeEvents( number device)
```

The events have 4 payloads, being Device, Destination, Source and Control in pld1, pld2, pld3 and pl4 respectively.

So you can either learn each channel (see Display Wave Volume command) to a command or use a “On Lua-Volume” event and dispatch in some Lua code (see “Event Switch Example” group on how to assign the event).

LUA Helper Functions

LuaVolume automatically loads LUAVolume.LUA. This script contains many helper functions to make life easier for the Girder programmer. The script is located in the Girder directory and also provides the user with an excellent example of how to use the LuaVolume library.

Constants

LuaVolume declares several constants for ease of use. All constants are declared within the table LVF (Lua Volume Functions).

```
— Default Mixer
LVF.DefaultMixer = 0

--Destination Type Constants
LVF.DestinationTypeDestination = 0 — playback
LVF.DestinationTypeSource = 1 — recording

--Source Type Constants
LVF.SourceTypeMaster = -1
LVF.SourceTypeWave = 0

--Control Type Constants
-- this list includes common control types
LVF.ControlTypeBass = 1342373890
LVF.ControlTypeEqualizer = 1342373892
LVF.ControlTypeFader = 1342373888
LVF.ControlTypeMute = 536936450
LVF.ControlTypeMux = 1879113729
LVF.ControlTypeOnOff = 536936449
LVF.ControlTypeTrebel = 1342373891
LVF.ControlTypeVolume = 1342373889
LVF.ControlTypeLoudness = 536936452
```

Functions

Several Lua functions are provided to make life a little easier on the unsuspecting programmer. They are:

`name = GetControlTypeString` (i)
Returns a text description of a control. Return control type as a number if control name is unknown.

`control = GetControlForSource (device,destination,source,controltype)`
Returns the numeric number of a control. See the ControlType constants above for common controls.

`err = SetSourceVolume (device, destination, source, volume)`
Sets the volume for the specified source. Volume is in percent (0-100).

`volume,err = GetSourceVolume (device, destination, source)`
Gets the volume for the specified source. Volume is in percent (0-100).

`err = SetSourceMute (device, destination, source,mute)`
Sets the mute for the specified source. 0 = mute off, 1 = mute on

`mute,err = GetSourceMute (device,destination,source)`
Gets the mute state for the specified source. Mute state is as above.

```
err = SetControlBalance (device, destination, source, control, bal-
ance)
```
Sets the Balance for a specified source.

The following functions are to provide quick and easy access to the Master and Wave devices.

```
err = SetMasterVolume (volume)
volume,err = GetMasterVolume ()
err = SetMasterMute (mute)
mute,err = GetMasterMute ()
err = SetWaveVolume (volume)
volume,err = GetWaveVolume ()
err = SetWaveMute (mute)
mute,err = GetWaveMute ()
```

Lastly, the details of a mixer device are enumerated by calling this function.
`function MixerPrintDetails (device)`

Function Reference

(also check the simplified functions on the previous page)

CloseVolumeControls

input

Number	**Device**

output

Number	**Error**

GetDeviceName

input

Number	**Device**

output

String	**Name**
Number	**Error**

GetDestinationName

input

Number	**Device**
Number	**DestinationIndex**

output

String	**Name**
Number	**Error**

GetSourceName

input

Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**

output

String	**Name**
Number	**Error**

GetControlName

input

Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**

output

String	**Name**
Number	**Error**

GetDeviceCount

input

none

output

Number	**Count**

GetDestinationCount

input

Number	**Device**

output

Number	**Count**

GetSourceCount

input

Number	**Device**
Number	**DestinationIndex**

output

Number	**Count**

GetControlCount

input

Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**

output

Number	**Count**

GetChannelCount

input

Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**

output

Number	**Count**

GetItemCount

input

Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**

output

Number	**Count**

GetChannelValue

input	
Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**
Number	**Channel**
Number	**Item**
output	
Number	**Value**
Number	**Error**

SetChannelValue

input	
Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**
Number	**Channel**
Number	**Item**
Number	**Value**
output	
Number	**Error**

GetControlType

input	
Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**
output	
Number	**Type**

GetControlBounds

input	
Number	**Device**
Number	**DestinationIndex**
Number	**SourceIndex**
Number	**ControlIndex**
output	
Number	**Type**
Number	**Min**
Number	**Max**

RegisterForVolumeEvents

input	
Number	**Device**
output	
Number	**Error**

UnregisterForVolumeEvents

input	
Number	**Device**
output	
Number	**Error**