Protocol-OSC

Open Sound Control (OSC) is a widely used protocol for controlling all sorts of devices. This is a generic protocol implementation, so you have to know the exact commands your device reacts to, and how to connect to it.

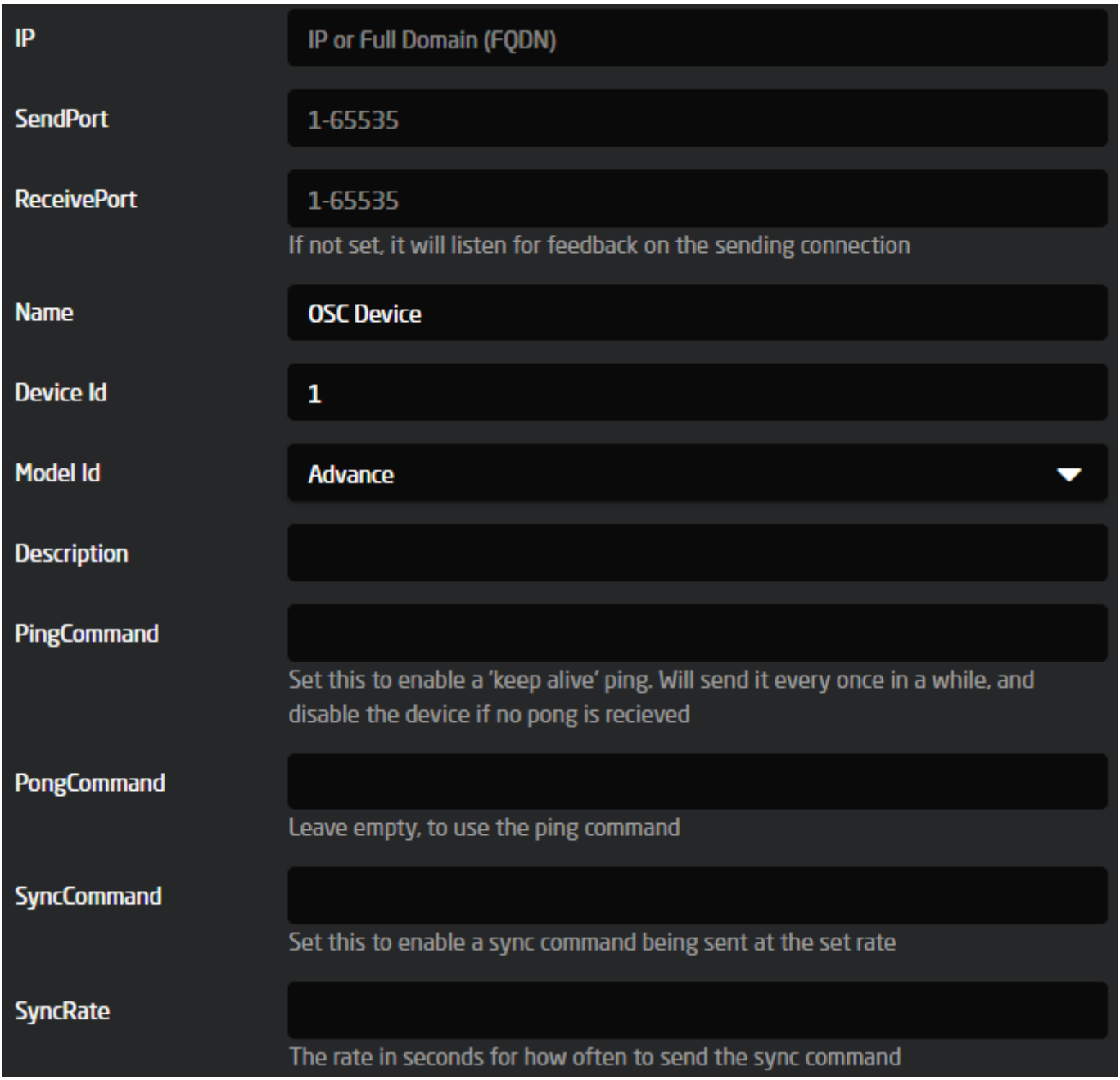
OSC works by sending commands with an address and often also an argument. An example: To control 'level' on a device, the command could look like this:
• Address: '*/level/1*'
• Argument: '*1.5*'

Most OSC commands uses slashes in the address to divide the address into fields. The address above '*/level/1*', could specifies that we want to control the level parameter, on channel number 1. These addresses needs to be known before hand, and can often be found in the manual or the OSC specifications for the device.

At the moment our OSC device core only works via UDP. This is the standard for most OSC devices, but a few uses TCP.

Configuration

To configure the device you need to set some device specific information.



- IP
  Set the IP address of the device to control
- Send Port
  Set the port to send OSC commands to. This can either be set on the device or found in the manual.

- Receive Port
  Set the port to listen for OSC commands. Some devices sends the return commands on the port it receives commands from. If this is the case, then leave it empty, otherwise set the port specified on the device or found in the manual.

- Ping Command
  To check if the device is connected and active, you can specify a command to send once in a while. If we receive a response, we expect the device to be connected. This is often

something like "/Ping" or "/KeepAlive".
If the Ping Command is not set, the device status will always be connected, since there is no way of knowing if there is a device listening.

- Pong Command
  On some devices, the response to the Ping Command are different from the command send. For instance, some devices will respond with "/Pong" when a "/Ping" is sent. Specify it here, or leave it empty to use the Ping Command.
- Sync Command
  Some devices needs to receive a sync command once in a while, to keep sending feedback. This can be left empty.

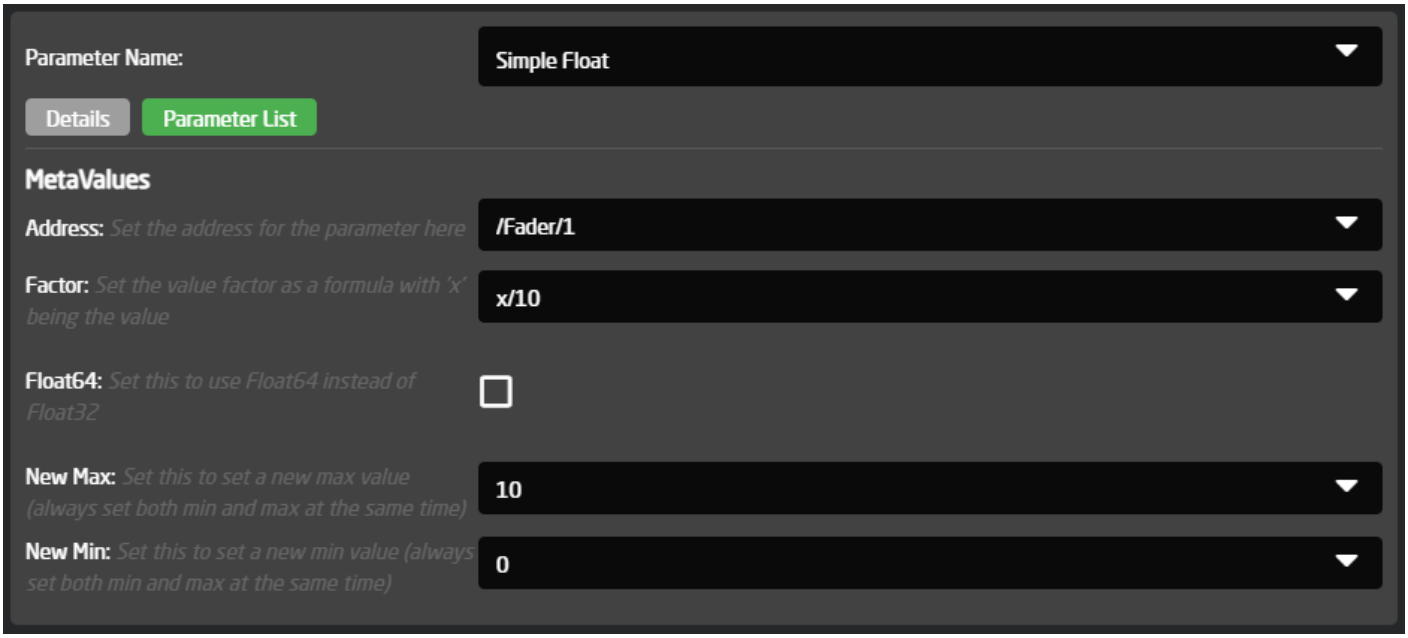
- Sync Rate
  Set how often the Sync Command should be send. In seconds.

Models

The OSC core provides 3 different models to use:

1. Simple model
This model includes parameters for very simple one way OSC control.

A control parameter has to be specified with a few meta parameters.



- Address: This is the address to send the command to.

Type specific meta parameters.

- New Max: This specifies the max of the range for float and integer parameters.
- New Min: This specifies the min of the range for float and integer parameters.

- Float64: OSC can send both Float32 and Float64 arguments. The default is Float32 but set this to use Float64.
- Factor: Float commands can have a factor set before being send. For instance some faders go between -90 to 10, but the argument send needs to be between 0-1. A factor can be specified to do this. Set 'x' as the value and write the math needed. For instance '(x+90)/100'.
- As Integer: For Bool commands it can either send it with an argument of 'true' or 'false', or as an integer with value 1 or 0.
- Invert: Bool commands can be inverted when sent.

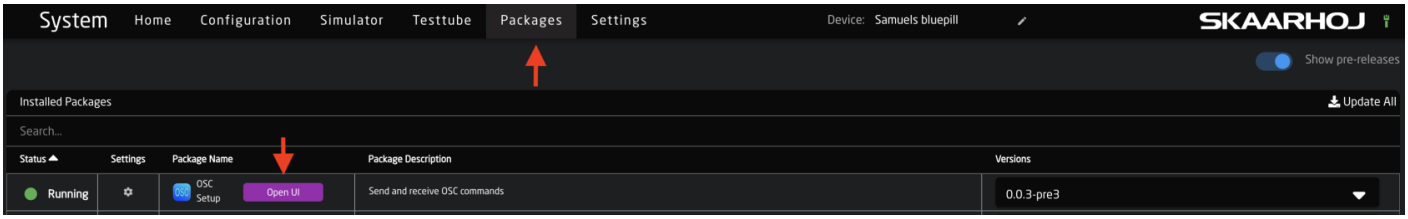
2. Dynamic model

This model includes everything from the simple model, but also a way to receive feedback on parameters.
Each parameter have a number of dimensions. These dimensions is assigned different commands, that then can be controlled, or could listen to feedback.

For each control type, there are 3 parameters. The control parameter itself, a command parameter and a config parameter. Each of these have the same amount of dimensions.

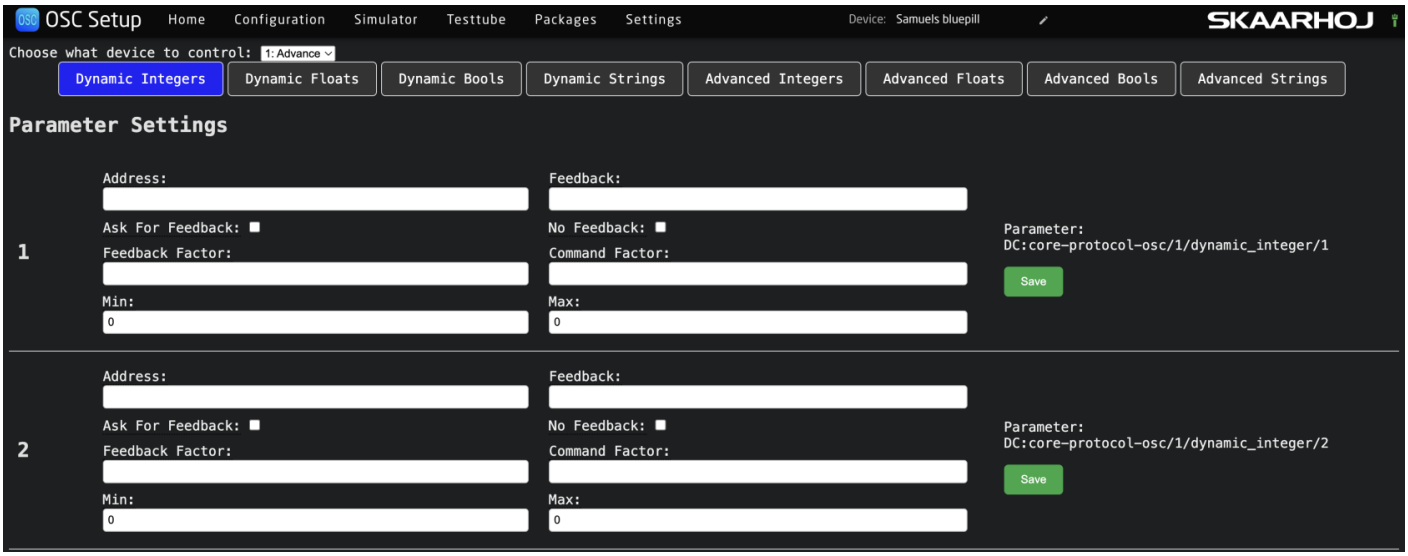
- The control parameter.
  This is the main parameter, where you can control what you want.
- The command parameter.
  This is a very simple parameter that shows the current address.
- The config parameter.
  This offers one way to setup all the info needed for the control parameter to work.

The other way to setup the control parameters, is through the webui. This offers a easier way to make multiple parameters ready to use. To open the webui, a dynamic or advanced model needs to be configured. Open reactor, and navigate to the 'Packages' menu. Here you can find the osc package, and a button to open the UI.



At the top of the page, there are a dropdown menu to select what device to configure, along with a row of tabs for each type of parameter available. If a dynamic model has been configured, 4 tabs with the datatypes 'Integers', 'Floats', 'Bools' and 'Strings' are available.

Each tab contains a row of parameter entries to set. Here it is possible to setup everything needed to setup each command to control.



The dynamic config parameter includes many of the same things as the simple parameter. The new things are:

- Ask For Feedback: Some devices won't reply to a command, but needs to receive a command without an argument to return the current value.
- Feedback: Some devices have a different address for feedback compared to the control address. Specify that here or leave it blank.
- Feedback Factor: If a Command Factor has been set, the opposite Feedback Factor is also needed. If the Command Factor is 'x*10' the Feedback Factor needs to be 'x/10'.
- No Feedback: If the device does not return feedback for this parameter, the No Feedback parameter can be enabled.

3. Advanced model
This model includes everything from the both the simple and dynamic model, but also a way to setup multiple dimensions of a parameter. For instance if you have many faders to control, you only have to configure it one time, and it will auto populate the second dimension of control.

The Advance parameters have the same 3 parameters for each type as the dynamic parameters. The main difference is , that when setting the address or feedback, a string formatter can be used. For instance if you have 20 level commands ('/Level/1', '/Level/2', '/Level/n'...). Instead of setting them up one by one, the Address can be set to '/Level/%d', and then the Count meta parameter set to 20. Now the parameter will be populated with 20 sub dimensions to control, one for each level control.
Some parameters will need leading zeros, for instance '/Level/01'. To set this, use '/Level/%02d'. That will make it a width of 2 and pad it with zeros.

Example

As an example, lets say we want to control a simple audio mixer with 12 channels. From the manual, we find the following OSC parameters we want to control:

Fader control - a 0 to 1 float with the address '/mixer/level/x'. The x marks the channel number from 01-12.
Mute control - a 0 or 1 integer with the address '/mixer/mute/x'.
Pan control - a -100 to 100 integer with the address '/mixer/pan/x'.

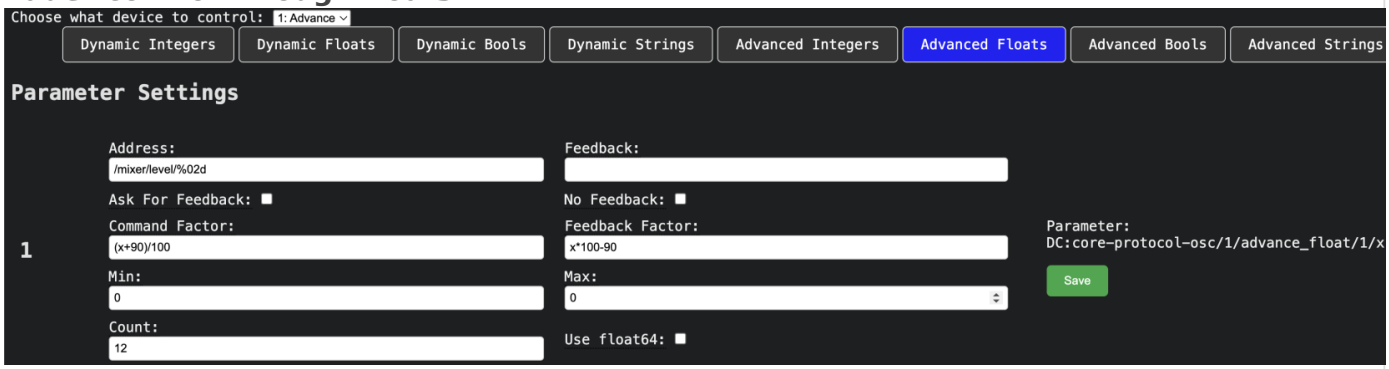
In the manual, we also see that in order to get a current value from the device, we can send the address without a value, and it will return the current value.

Since we have multiple channels for each command, we will use an advanced model.

To setup each command, we open the UI as described earlier.

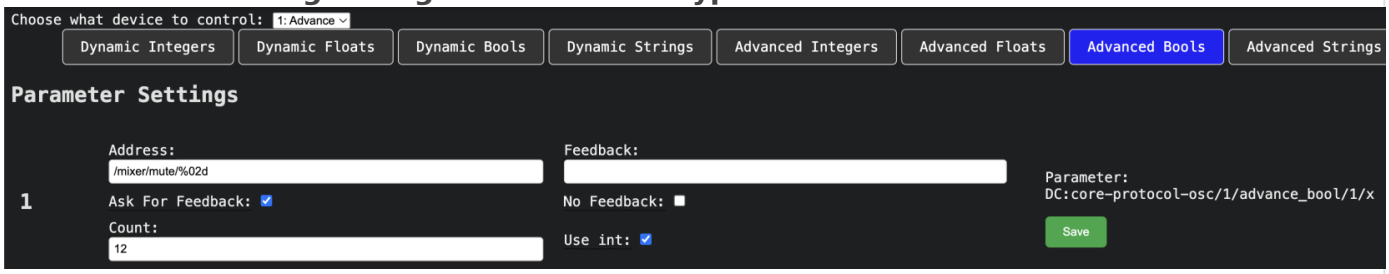
Each of the advanced tabs will be configured as following.

Fader control through floats



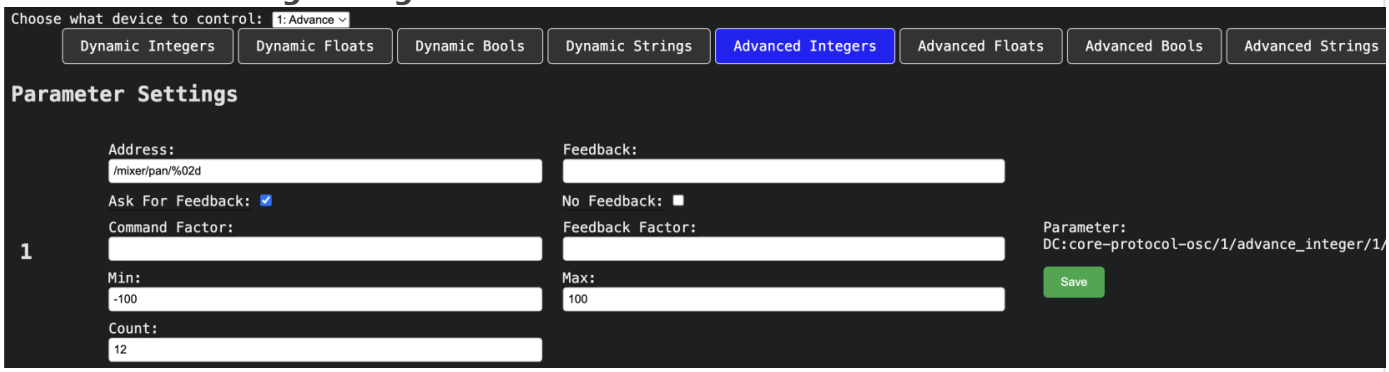
Here the address is setup with the last part as %02d. This is a string formatter that will take the channel number and add is as a number with a width of 2 padded with zeros. Eg. Channel 1 would be: '/mixer/level/01'.
The feedback is left empty, and will then default to the same as the address.
Ask for feedback is enabled, since we can get the current value by asking for it.
The min and max is set to the range of the fader, from -90dB to 10dB.
From the manual, we know that the float to send, needs to be from 0-1. To get this, we use a command factor of '(x+90)/100', and the reverse for feedback factor 'x*100-90'.
Finally we set the Count to 12, since that is the amount of channels we want to control.

Mute control through integers with a bool type



Most things are setup as above. The only difference is, that the 'Use int' is checked. This tells the core to use an integer of 0 or 1 instead of 'true' or 'false' values.

Pan control through integers



Again, many of the things are the same as above. The min and max is now set directly to -100 and 100, with no factors. This will send the raw value from -100 to 100.

Revision #6
Created 5 September 2023 06:13:16 by Samuel Jakobsen
Updated 8 January 2025 08:30:40 by Samuel Jakobsen