

OSC in BEYOND

Introduction

OSC is a communication protocol, that defines the format of data and other communication aspects. The means by which we handle this data, depends on the application that it will be used, and typically, this is defined by the server.

OSC communication has two parts. A client will initiate a command(s), and the the server will execute the command(s). You will need to know what commands can be executed and are supported by the server. If the you send an unknown command to the server, the server will ignore that command.

OSC uses a standard network communication protocol, called UDP. UDP and TCP/IP are the most widely used communication methods, and they are common in computer devices, smart phones, and other equipment that use ethernet connections. To use OSC, you need to have your equipment connected in a network. For example, in a home environment you have a router, and then you may have a computer, smartphone or similar device connected to that router. The router assigns an IP address, and allows devices to communicate inside of this local network and with the internet. When using OSC, we only need to have a local network setup.

The base of OSC protocol, is the message. The message is divided into three parts. 1) address, 2) type of arguments, and 3) arguments. Think of the address within OSC, as being similar to the address you would use when sending a postcard to a client or friend. When you send a post card to someone, you note the destination it will be sent to, and generally include the country name, city, street and building number. When using OSC, it will represent this address information, as one string, and it will use slashes (/) to separate the information. An example would look something like this (again referring back to our address example): /country/city/street/building#

For the argument, we will once again refer back to our postcard example. Think of the argument as the text or message you would write on the back of the postcard. Within OSC, this is defined within a string, and that string states what the message is, and also contains the text and information relating to the message.

In thinking this way, an OSC message becomes quite easy to understand. It will define who receives the information (the address), it will define wha information needs to be communicated (type of argument) and then it will define precisely what the message is (the argument itself).

The OSC server should provide an address list and specify what argument should come with each message.

For example, if we want to control the master brightness of BEYOND, than the address would be: /beyond/master/brightness

After this, we must include a number that will represent the value of the brightness, this is our argument. If the message comes without any argument, or with more than one argument - it will not make sense to the OSC server. Each message should have only one argument, and the value specified must be

something that the server will understand.

BEYOND will ignore a message, if it includes incorrect argument types. BEYOND will also ignore the message, if it has an unknown address.

More information OSC can be found at: <http://opensoundcontrol.org/introduction-osc>

Using OSC in BEYOND - Three Directions

During the development of BEYOND, we started with the implementation of the server that will execute OSC messages. This server provides access to the master settings, to cues, to projection zones and so on. Step by step, it turns into a pretty large subsystem within BEYOND - But, the fact remains, that the server's only function is to analyze the incoming address and parameters, and call up the corresponding functions of BEYOND. The PangoScript tool within BEYOND also has similar functionality. Commands are represented using a text format. The scripter analyzes the text, detects the commands and arguments, and then executes them.

Again, the same logic used for PangoScript, is used for OSC, and the scripter calls up the corresponding functions of BEYOND. As such, the script interpreter for PangoScript and the OSC server of BEYOND are similarly related, and this also means that it is possible to create a gateway from the OSC server, to PangoScript and from PangoScript to the OSC server. The purpose of the gateway, is because OSC can handle some things better than the script. At the same time, the scrip offers more flexibility, and allows you to handle functions that the OSC server alone cannot do. The main point in this all, is that it is possible to use OSC servers inside of the PangoScript tool.

Using OSC servers in BEYOND and PangoScript is simple. You only need to know the address and the arguments that will be used. As soon as the first character of the PangoScript line is "/" then this line will be processed as an OSC message. An example of a PangoScript and equal OSC command are below:

```
Brightness 100  
/beyond/master/brightness 100
```

The first line is a pure PangoScript command. The second line is an OSC command. It starts from the slash. The interpreter will parse the address, and then read arguments one by one. The type of arguments are then detected automatically. After this, the interpreter creates an OSC-command-object and sends it to OSC server for execution. The OSC server has no idea who created the OSC-command-object. It can be a network-interface, or it can be a PangoScript. In both cases the result is equal, and the server will do the job.

One of the attractive feature of using OSC for PangoScripting, is the ability to use special characters like (*), that allow address to groups of recipients. A good example of this, is access to cues. An address for a cue is /cue/n/m/... where n is the page index, and m is the cue index.

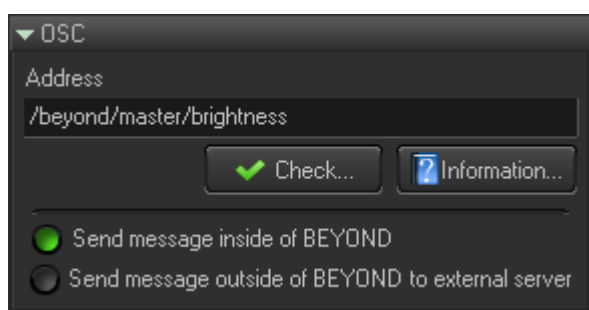
Such addresses look simple and compact but processing might take a bit of time, and all of this depends on how many objects the address string will cover. PangoScript commands do not have ability to send commands to multiple destinations (except "selected") because the OSC syntax extends the functionality

of PangoScript.

NOTE: If you want to check how OSC messages work, instead of using an OSC client running on a separate computer, you can test the message and command in the PangoScript Editor.

The third direction is the Universe window. It can also work as an OSC Client. The Universe window has components such as sliders, buttons, XY pads and so on. It is possible to define addresses and then click on components to generate OSC messages. The message can be sent inside of BEYOND, or outside of BEYOND to the network.

If you send the message inside of BEYOND, then an interesting situation occurs, whereby BEYOND is a server and client at the same time. If you choose to send messages outside to a network, then BEYOND will act as a standard OSC client. OSC standards do not specify a standard port number. This is instead configured by you, the client.

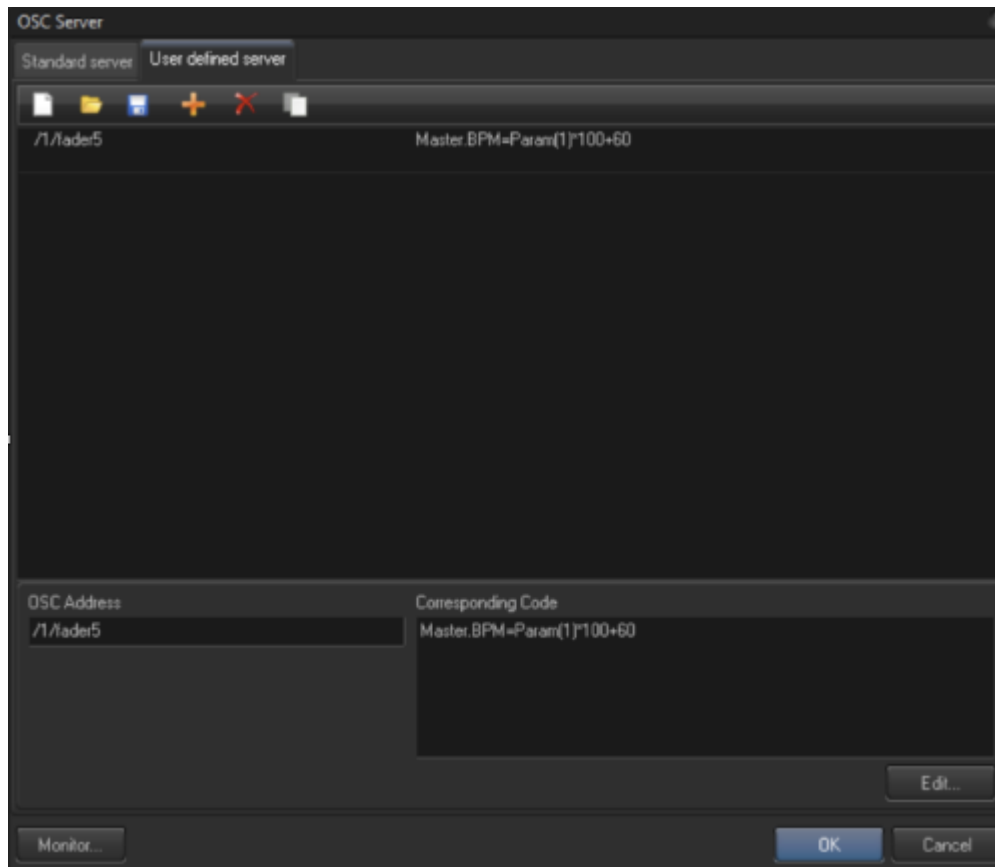


Dynamic BEYOND OSC Servers (OSC to PangoScript)

BEYOND has a number of servers with fixed addresses. The address structure and arguments are made to reflect BEYOND, in a logical way. The address and arguments are not made for an exact OSC client. There is also a small chance that demo layout of some OSC clients will have an address suitable for BEYOND. What this all means, is that you need to adjust the settings of the OSC client according to the address system in BEYOND.

BEYOND is flexible though, and can be adjusted for the OSC client. This is not the best way, but it is possible. Some free OSC applications (in the AppStore for example) have no setup at all. The addresses are fixed and cannot be changed. Some clients are able to compose the address but only by using a rule, and in this case, it might not match the BEYOND address system. In such cases, BEYOND offers the OSC to PangoScript gateway as an alternative.

The idea behind the OSC to PangoScript gateway is also quite simple. We've created pairs of OSC addresses and corresponding PangoScripts. BEYOND receives the OSC message and compares the address of the pair. If the address is equal, then BEYOND executes the corresponding script. All arguments from OSC messages are accessible by means of the ExtValue() or Param() functions of PangoScript. The number of pairs is not limited.



Sending an OSC Message, from PangoScript

Here is a command called `OscOut`. The first argument here is the OSC address, the second and others are used as arguments in the OSC message.

Example:

```
OscOut "/myserver/myfunction", 1
```

This command generates outgoing OSC messages (to the network) with address `"/myserver/myfunction"` and the only numeric argument is equal to 1.

There is a similar command - `OscOutTTS`. The only difference with this command is the type of tag string (tts). TTS describes the types of arguments.

Supported types are:

s	string
i	integer
f	float

For some OSC clients, the data type and number are critical. `OscOut` uses a float point number, for all numbers. If your OSC client requires an integer, then use `OscOutTTS`

Sending Feedback to an OSC Client

Currently BEYOND does not generate feedback messages to the client, that reflect the change of BEYOND objects. There are only two exceptions to this rule. The main reason we've done this, is to eliminate the high risk of overload of to the network and BEYOND software.

For example, BEYOND supports up to 100 pages and up to 100 cues in the page. Each Cue has an embedded LiveControl object, with ~ 50 variables inside. $100 \times 100 \times 50 = 500,000$. Half of a million UDP messages is a serious load.

BEYOND uses thousands of objects, and many of them are accessible using OSC. Generating feedback for each change in the system is simply not practical.

Our current solution for generating the feedback is PangoScript. Using PangoScript, it is possible to organize the background script that will check the state of important BEYOND objects, and it will also generate the OSC messages by means of an OscOut command.

NOTE: BEYOND generate feedback for **/b/** server and for **/beyond/zonesetup**

Universe Server

Lets begin by using an example. The Universe window contains layouts created by you, the client. The name of the Layout and the name of the Component are user-defined. For example, we put a slider on the layout. To access the slider value in PangoScript, use a code like the one below:

```
Layout1.Slider1.Value = 100
```

Here we see 3 identifiers - The identifier of the layout, the identifier of the slider, and the identifier of the property - Value. The Universe window can be accessed from OSC by means of **/u/** server.

Example:

```
/u/layout1/slider1/value
```

A third example that demonstrates the **/u/** server, is access to Scripters. BEYOND supports multitasking for scripts. Each scripiter (interpreter) that execute a script is a separate object. The Command CodeName sets the identifier for the scripiter. Initially, it is used for the CallCode() operator. But later it gets another role, as an identifier of the Scripiter object. The script can define variables. All such variables are accessible in object style - scripiter.variable. An example of a practical use of such a feature, is to create a script that acts like the VLJ (which clicks on cues), and then you can modify the page number from OSC.

/b/ Server - a gateway between objects and osc

/b/ server allow you to operate with ALL objects of the BEYOND software. This is one of latest additions in the OSC area.

Example:

```
Master.Brightness = 100
```

Here we see a PangoScript operator that sets the Brightness property of the Master object to 100. The address for the OSC message is: **/b/master/brightness**

As you may have noticed, the difference is the leading /b/ and after that the name stays as is. But now we see a dot character replacing the slash... Very simple.

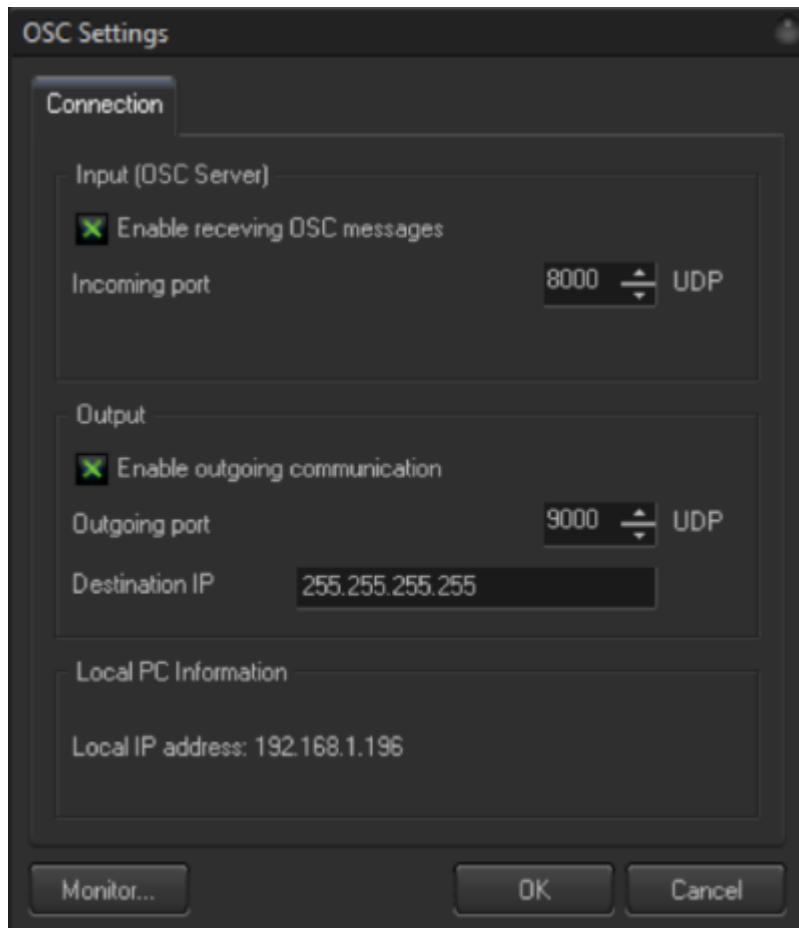
This is a **bidirectional** gateway. As soon as BEYOND receives incoming OSC commands for an object, it will generate feedback for it. The idea here is simple. There are so many objects, and as such, creating feedback for everything is just not possible. And as a practical note, nobody would really need to use all of these anyway. So in essence, you control a parameter, that you will want to use feedback with.

NOTE: **/b/** server has a limitation. It does not support special characters in the address, like (*) or (?). This is a direct and conceptually simple gateway, based on the replacement of a dot to a slash and visse versa.

Configuration Dialog

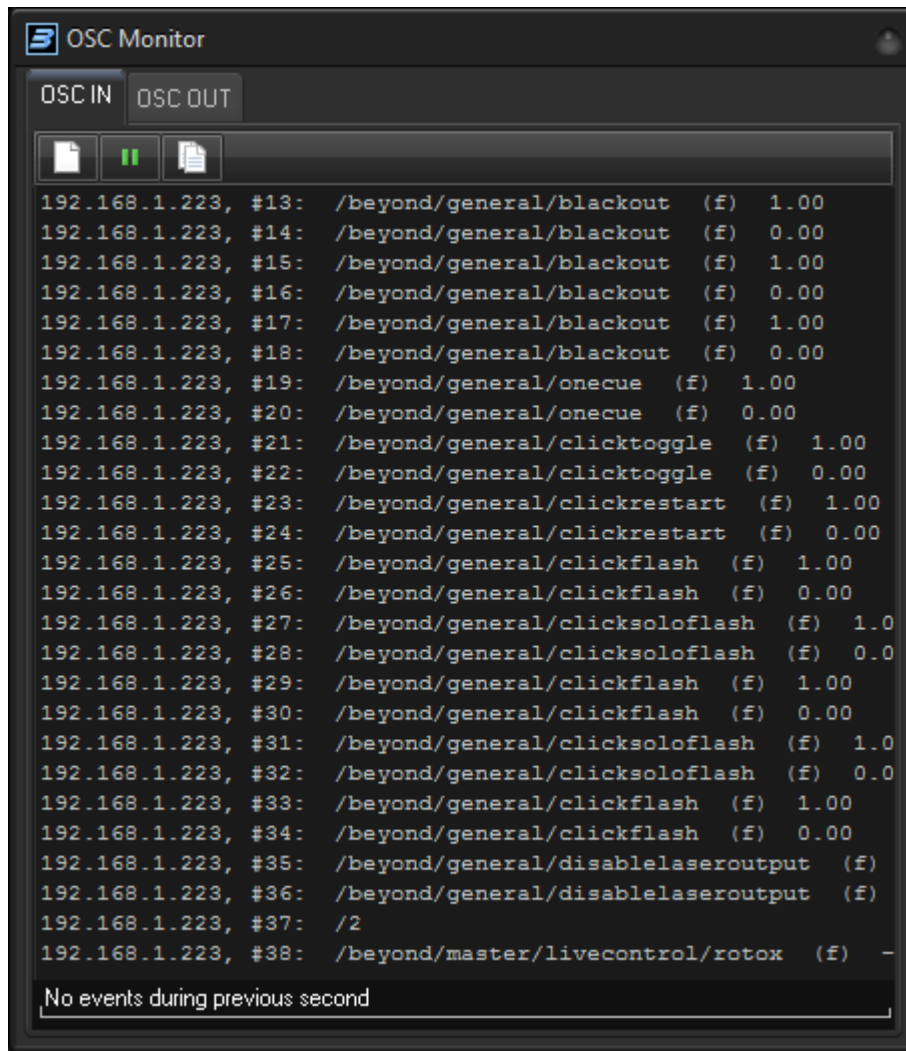
BEYOND uses UDP protocol for OSC. OSC clients use various ports. Please check what ports your OSC client uses, before configuring BEYOND.

255.255.255.255 is a broadcast IP address. It means that all PCs in this local network will receive the messages. OSC clients may block broadcast messages. Furthermore, when using a network, it is much better to use the exact IP of the recipient.



The OSC Monitor

The OSC monitor captures all of the packets from the communication layer. This is a great diagnostic tool. If you send a message to BEYOND and it does not react, then please open the Monitor. If you see incoming messages, then the communication layer is working properly, and the problem could be within the address or arguments. If you do not see the messages, then the problem is in the communication level – network, IP address settings, client, or similar. The monitor is there to help you understand where the problem lies, inside or outside.



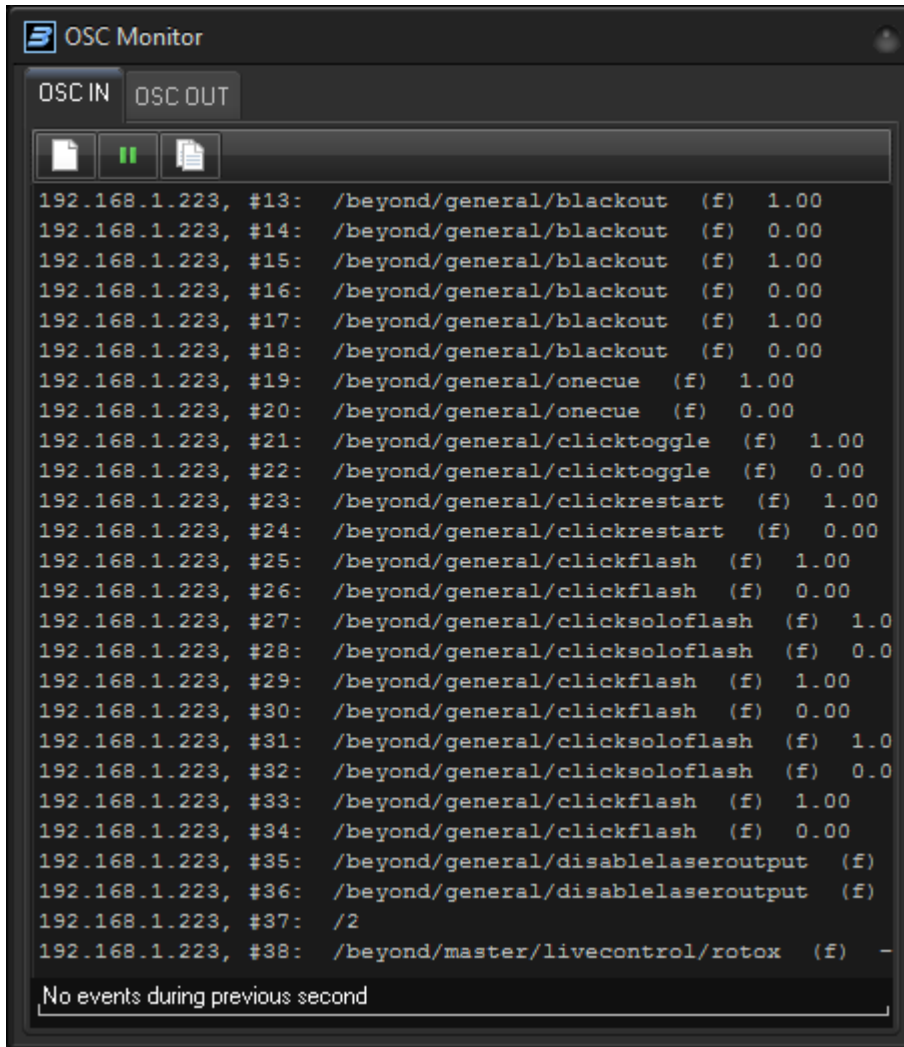
The Monitor shows:

1. The IP address of the sender
2. The OSC address
3. The Type Tag String - On the screenshot you see (f), this is TTS. It means one float point number.
4. The Values themselves, and they can be the number(s) of string(s)

At the bottom of window there is a timeline of messages. Each message adds a marker on the timeline. For example, when you drag a slider in TouchOSC, it generates a sequence of OSC messages. If the messages arrive correctly, then you will see a nearly linear marker distribution by time.

In some cases the routers (or network itself) do not work as fast as they should, and it may appear to be delayed. In this case markers appear in groups, and there will be big gaps between the markers. This information is important because when messages come with a delay or pause in between, you will see that delay within the reaction of BEYOND, and it may appear as though BEYOND has a problem when it actually does not. The Monitor is a tool that helps you make correct diagnostics quickly.

Hint: There is quick OSC monitor, Configuration dialog, “Laser Preview” tab, “Display OSC messages” checkbox.



```
OSC Monitor
OSC IN  OSC OUT
192.168.1.223, #13: /beyond/general/blackout (f) 1.00
192.168.1.223, #14: /beyond/general/blackout (f) 0.00
192.168.1.223, #15: /beyond/general/blackout (f) 1.00
192.168.1.223, #16: /beyond/general/blackout (f) 0.00
192.168.1.223, #17: /beyond/general/blackout (f) 1.00
192.168.1.223, #18: /beyond/general/blackout (f) 0.00
192.168.1.223, #19: /beyond/general/onecue (f) 1.00
192.168.1.223, #20: /beyond/general/onecue (f) 0.00
192.168.1.223, #21: /beyond/general/clicktoggle (f) 1.00
192.168.1.223, #22: /beyond/general/clicktoggle (f) 0.00
192.168.1.223, #23: /beyond/general/clickrestart (f) 1.00
192.168.1.223, #24: /beyond/general/clickrestart (f) 0.00
192.168.1.223, #25: /beyond/general/clickflash (f) 1.00
192.168.1.223, #26: /beyond/general/clickflash (f) 0.00
192.168.1.223, #27: /beyond/general/clicksoloflash (f) 1.0
192.168.1.223, #28: /beyond/general/clicksoloflash (f) 0.0
192.168.1.223, #29: /beyond/general/clickflash (f) 1.00
192.168.1.223, #30: /beyond/general/clickflash (f) 0.00
192.168.1.223, #31: /beyond/general/clicksoloflash (f) 1.0
192.168.1.223, #32: /beyond/general/clicksoloflash (f) 0.0
192.168.1.223, #33: /beyond/general/clickflash (f) 1.00
192.168.1.223, #34: /beyond/general/clickflash (f) 0.00
192.168.1.223, #35: /beyond/general/disablelaseroutput (f)
192.168.1.223, #36: /beyond/general/disablelaseroutput (f)
192.168.1.223, #37: /2
192.168.1.223, #38: /beyond/master/livecontrol/rotox (f) -
No events during previous second
```

From:

<https://wiki.pangolin.com/> - Complete Help Docs

Permanent link:

https://wiki.pangolin.com/doku.php?id=beyond:osc_in_beyond

Last update: **2020/06/11 19:20**

