MXWendler OSC Interface Description Version 2.3

09.12, valid for builds > 1918 (Version 4.2.18)

This document describes the MXwendler (MXW) OSC Command Interface. You will learn how to control MXwendler through the OSC interface. We will describe the hierarchical control approach and how to assemble OSC command strings.

1. What is OSC?

OSC stands for „Open Sound Control“. It was developed as a successor of the MIDI control. From the OpenSound Control home page1: „OpenSound Control ("OSC") is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology“.

OSC Features

- High resolution time tags
- Numeric and symbolic arguments to messages
- Open-ended, dynamic, URL-style symbolic naming scheme
- "Bundles" of messages whose effects must occur simultaneously
- Pattern matching language to specify multiple targets of a single message
- Query system to dynamically find out the capabilities of an OSC server and get documentation
- Broad support among all popular media control software

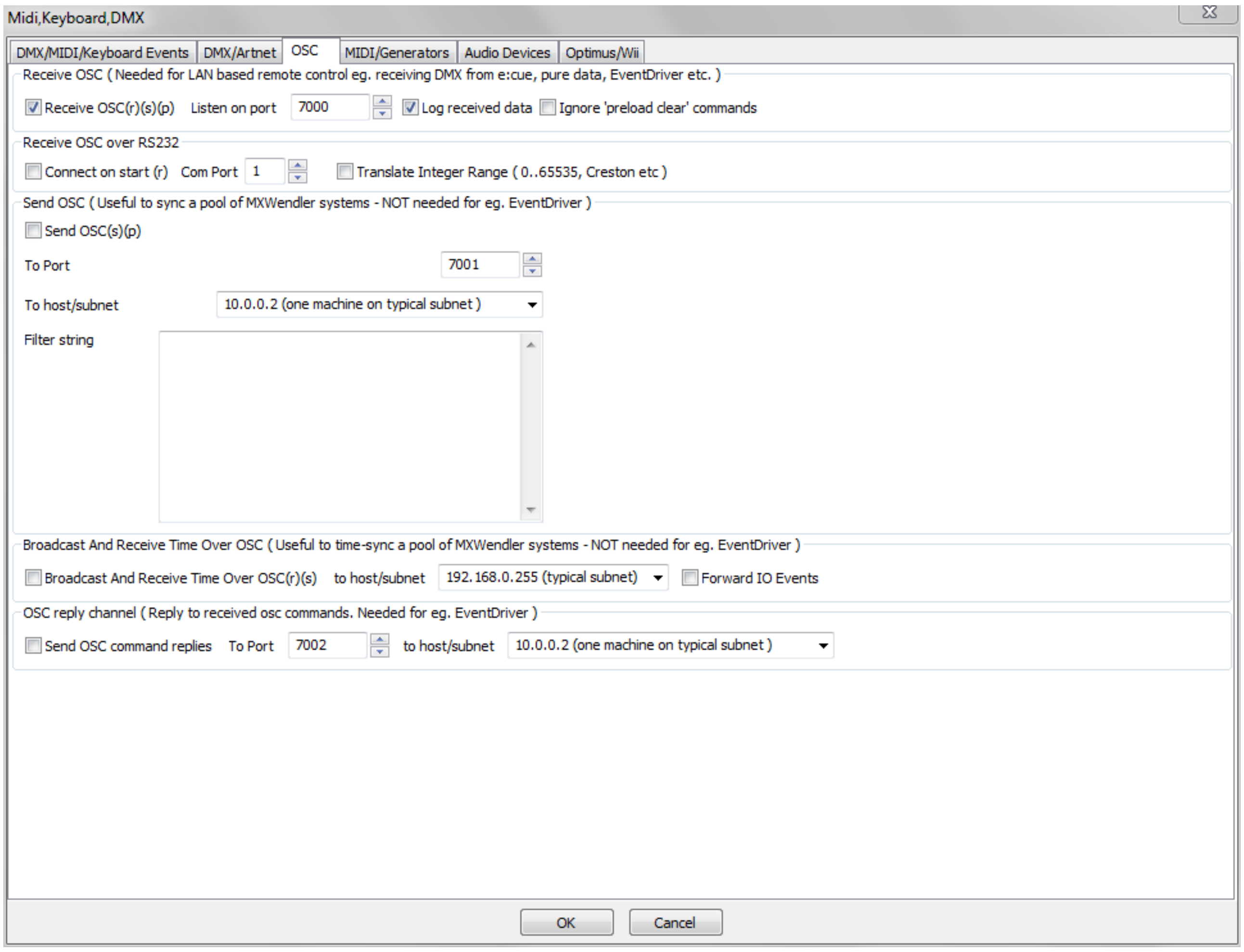
For more Information please see the OSC home page at the Berkeley University or the Internet Community Page opensoundcontrol.org.

1.1 Why control MXW through OSC?

MXW is designed to be a high-performance media server. It interactively composits high-resolution footage and interactive media like flash media. OSC commands can be generated and sent from a wide range of applications and signaling frameworks like pd, Max/MSP, eyesweb, Ableton Live and many more. Separating the media servers from the controlling infrastructure gives you various tremedous creative options: create, control and sync panoramic scenes, separate scene analysis (eg. audio, video) from the compositing tasks for performance reasons, play distributed places, create custom interfaces for you or your customers or save money by seperating expensive content creation systems from cheap render slaves.

2. How do i receive OSC commands?

Enabling OSC receiving is very easy: Just turn it on by checking „Start OSC“ in the Open Sound Control Panel (Settings -> IO):


Midi,Keyboard,DMX

DMX/MIDI/Keyboard Events | DMX/Artnet | OSC | MIDI/Generators | Audio Devices | Optimus/Wii

Receive OSC (Needed for LAN based remote control eg. receiving DMX from e:cue, pure data, EventDriver etc.)

[✓] Receive OSC(r)(s)(p) Listen on port 7000 [✓] Log received data [ ] Ignore 'preload clear' commands

Receive OSC over RS232

[ ] Connect on start (r) Com Port 1 [ ] Translate Integer Range (0..65535, Creston etc)

Send OSC (Useful to sync a pool of MXWendler systems - NOT needed for eg. EventDriver)

[ ] Send OSC(s)(p)

To Port 7001

To host/subnet 10.0.0.2 (one machine on typical subnet)

Filter string

Broadcast And Receive Time Over OSC (Useful to time-sync a pool of MXWendler systems - NOT needed for eg. EventDriver)

[ ] Broadcast And Receive Time Over OSC(r)(s) to host/subnet 192.168.0.255 (typical subnet) [ ] Forward IO Events

OSC reply channel (Reply to received osc commands. Needed for eg. EventDriver)

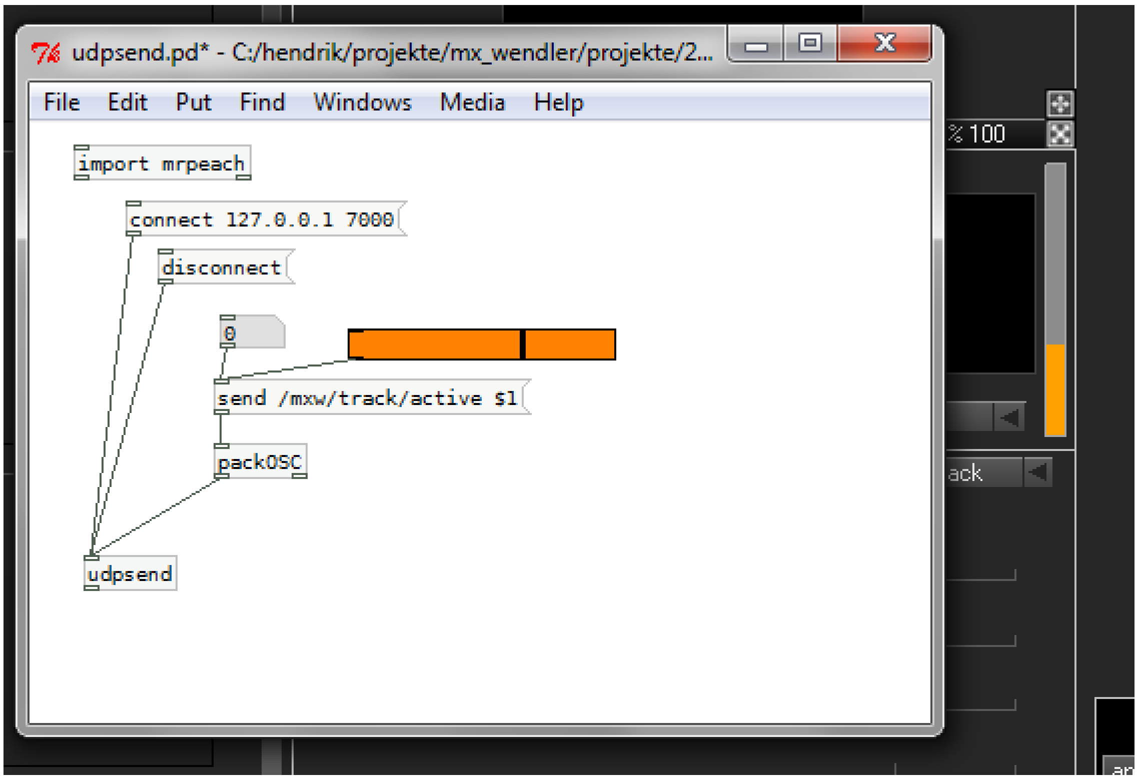
[ ] Send OSC command replies To Port 7002 to host/subnet 10.0.0.2 (one machine on typical subnet)

- Leave the port to 7000.
- Log received data will log commands in the Error and Log Window
- You have to restart your application to apply any changes.

Leave the rest of the OSC settings as they are for now.

3. How do i send OSC commands from PD (puredata.org)?

You need a PD with a running sendOSC object. A minimal patch looks like this:



```
import mrpeach
connect 127.0.0.1 7000
disconnect
0
send /mxw/track/active $1
packOSC
udpsend
```

After you click on [connect localhost 7000] you should see „udpsend: connecting to port 7000“ in your command window. If you see something else, this port is either already in use or you are blocked by a firewall.

To send an OSC command, you have to send a string with a special syntax. To put the MXW main fader in the middle position, create a patch like the patch above:

1. enter edit mode in pd (ctrl-e)
2. create a packOSC object
2. format OSC message: „send /mxw/track/active $1“
3. connect a slider to the messagebox, botton value 0.0 top value 1.0
4. connect the messagebox to the sendOSC object
5. leave edit mode and drag the slider

Moving the PD slider should now move the „active track“ slider in MXW. Of course you can change from localhost (127.0.0.1) to any other IP adress and thus remote control an instance of MXW.

4. MXW OSC command structure

Commands to the MXW OSC interface always follow the same syntax:

```
/mxw/[address of widget] [MSGID0001] [floatval from 0..1|string]
```

Commands always begin with „*/mxw/*“. The address of the widget is determined by a hierarchical access beginning with the tracks, continuing over the layers descending to the clips. There are some static adresses like /mxw/master which do not refer to a widget.

There is an optional component, the message id. If you send messages that generate a reply, you can prepend the message reply with an optional id, you send along with the command. This MSGID must follow the address as the first string OSC message argument. Example:

```
            MSGID0001
            MSGID123345667373726
```

The command

```
/mxw/preload/3 MSGID0123 clipinfo
```

will return

```
/mxw/reply MSGID0123 /mxw/preload/3 112 320 240 40 cachecomplete
c:\footage\clip.avi
```

The command receivers listen either to a float (if they are widgets) or to strings (to choose eg. patches or clips). Float values are always in the range 0.0 .. 1.0 and will be scaled to the actual ranges. Eg. the clip playback rate can be set in a range from -5.0 .. 5.0 by dragging from the slider from bottom to top -the according OSC values are nevertheless 0.0 .. 1.0. This behaviour is designed to be logically as close as possible to Midi ranges, which are always from 0 .. 127 and will be scaled to the actual useful ranges as well.

To address the main fader and fade from 0 .. 1 in 10 steps, you would send a command sequence

```
     /mxw/render 0.0
     /mxw/render 0.1
     /mxw/render 0.2
     ...
     /mxw/render 1.0
```

You are free to send finer grained values. The commands will be evaluated as soon as they arrive. The last command before a new frame is drawn will set the widgets value. There is no need the send commands more often than the MXWendler renderer updates.

5. MXW hierarchical command addressing scheme

Commands to the MXW OSC interface follow a hierarchical system, no matter if you use OSC to access MXW or MIDI or your keyboard. To address a certain widget, use always the following system:

```
"/mxw/keystone_io_animator",              // keystone io animator -
"/mxw/set",                               // choose a patch
"/mxw/editor",                            // coose ui tab
"/mxw/trackmanager",                      // choose active track
"/mxw/layermanager",                      // choose active layer
"/mxw/beatbutton",                        // send to beatbutton, 'tap' beats
"/mxw/playlist/gotostart",                // go to first track on first list
"/mxw/playlist/gotoprev",                 // go to previous cue
"/mxw/playlist/play",                     // play active item
"/mxw/playlist/pause",                    // pause active playling item
"/mxw/playlist/gotonext",                 // go to next cue
"/mxw/playlist/gotocue",                  // go to specific cue and immediately play
"/mxw/set/first",                         // go to first patch on first list
"/mxw/set/prev",                          // go to previous patch
"/mxw/set/play",                          // play active item
"/mxw/set/next",                          // go to next patch
"/mxw/render",                            // send value to render
"/mxw/render/opacity",                    // change o of render
"/mxw/render/scale",                      // ..
"/mxw/render/scalexy",                    // ..
"/mxw/render/translationx",               // ..
"/mxw/render/translationy",               // ..
"/mxw/render/rotation",                   // ..
"/mxw/render/aspectratio",                // ..
"/mxw/render/reset",                      // ..
"/mxw/render/effect/1/param/1",           // first param of first effect
"/mxw/track/1",                           // send value to specific track
"/mxw/track/active",                      // like render
"/mxw/preload/1/trigger",                 // trigger preload 1 content to current track
"/mxw/preload/2/trigger",                 // ..
"/mxw/preload/3/trigger",                 // ..
"/mxw/preload/[put no. here]/trigger",    // ..
"/mxw/preload/1/flipflop",                // flipflop preload 1 content to current track
"/mxw/preload/2/flipflop",                // ..
"/mxw/preload/3/flipflop",                // ..
"/mxw/preload/[put no. here]/flipflop",   // ..
"/mxw/fixture/1/opacity",         // send value to first slot of first track
"/mxw/fixture/2/scale",           // send value to second slot of first track
"/mxw/fixture/3/scalexy",         // send value to third slot of first track
"/mxw/fixture/4/translationx",    // send value to fourth slot of first track
"/mxw/fixture/5/translationy",    // send value to first slot of second track
"/mxw/fixture/6/rotation",        // send value to second slot of second track
"/mxw/fixture/7/mode",            // send value to third slot of second track
"/mxw/fixture/8/aspectratio",     // send value to fourth slot of second track
"/mxw/fixture/9/reset",           // send value to first slot of third track
"/mxw/fixture/10/clip/speed",     // send value to second slot of third track
"/mxw/fixture/11/clip/position",  // send value to third slot of third track
"/mxw/fixture/12/clip/reset",     // send value to fourth slot of third track
"/mxw/track/active/layer/active/opacity",         // change of active layer,track
"/mxw/track/active/layer/active/scale",           // ..
"/mxw/track/active/layer/active/scalexy",         // ..
"/mxw/track/active/layer/active/translationx",    // ..
"/mxw/track/active/layer/active/translationy",    // ..
```

```
"/mxw/track/active/layer/active/rotation",                       // ..
"/mxw/track/active/layer/active/mode",                           // ..
"/mxw/track/active/layer/active/aspectratio",                    // ..
"/mxw/track/active/layer/active/reset",                          // ..
"/mxw/track/active/layer/active/clip/keyin",                     // send to key in of clip of active layer,track
"/mxw/track/active/layer/active/clip/keyout",                    // ..
"/mxw/track/active/layer/active/clip/speed",                     // ..
"/mxw/track/active/layer/active/clip/position",                  // ..
"/mxw/track/active/layer/active/clip/mode",                      // ..
"/mxw/track/active/layer/active/clip/reset",                     // ..
"/mxw/track/active/layer/active/clip/effect",                    // ..
"/mxw/track/active/layer/active/clip/moreeffects",               // create effect. [delete effect] is missing
"/mxw/track/active/layer/active/clip/effectbatch",               // choose active effect
"/mxw/track/active/layer/active/clip/clips",                     // choose from clip library
"/mxw/track/active/layer/active/clip/effect/1/param/1",          // set param 1 of effect 1
"/mxw/track/active/layer/active/clip/effect/1/param/2",          // set param 2 of effect 1
"/mxw/track/active/layer/active/clip/effect/2/param/1",          // set param 1 of effect 2
"/mxw/track/active/layer/active/clip/effect/1/color/1/red",             // set red component of color param
"/mxw/track/active/layer/active/clip/effect/1/color/1/green",           // set green component param
"/mxw/track/active/layer/active/clip/effect/1/color/1/blue",            // set blue component param
"/mxw/track/active/layer/active/clip/effect/1/color/1/hue",             // set hue component param
"/mxw/track/active/layer/active/clip/effect/1/color/1/saturation",      // set saturation component param
"/mxw/track/active/layer/active/clip/effect/1/color/1/value",           // set value component param
"/mxw/keystone/element/1/pivot/col/1/row/1/xposition",                  // move keystone pivot in x
"/mxw/keystone/element/1/pivot/col/1/row/1/yposition",                  // move keystone pivot in y
```

6. MXW replies

When 'Send OSC command replies' is activated, certain commands generate replies. These replies will be sent to the reply receiver set in the OSC panel. The following commands generate replies:

```
[no command]
-> /mxw/reply livesign 610342 15286 22481348 37.9632
```
MXW will send every second a status message containing the application run time in milliseconds, the drawn frames, the current memory usage and the session cpu usage.

```
/mxw/sync [long]
-> /mxw/reply /mxw/sync 4203
```
MXW will send the difference between the application runt time and the new time and will continue with the new time.

```
/mxw/preload/5 clipinfo
-> /mxw/reply /mxw/preload/3 112 320 240 40 cachecomplete
c:\footage\clip.avi
```
MXW will send a clipinfo containing clip length, width, height, millispersecond, cachestate and path.

```
/mxw/preload/5 preview
-> /mxw/reply /mxw/preload/3 [blob with a 320x240 jpeg preview]
```
MXW will send a preview jpeg 320x240 of the preload.

Generally those commands can contain a message ID that will be sent back to the reply receiver. This message id has to be a string beginning with 'MSGID' followed by a integer. This ID MUST be placed directly after the widgets address.

```
/mxw/preload/5 MSGID0001 clipinfo
-> /mxw/reply MSGID0001 /mxw/preload/3 112 320 240 40
cachecomplete c:\footage\clip.avi
```

7. Examples:

```
/mxw/track/active 0.5
```
Fades the active track to 0.5

```
/mxw/preload/1 add
```
Adds a layer with the contents of preload slot 1

```
/mxw/track/1/layer/1/opacity 0.5
```
Sets the opacity of the first layer of the lower track to 0.5

```
/mxw/set 5
```
The active track will load patch 5

```
/mxw/track/active/layer/active/clip/speed 1.0
```
Sets the speed of the clip of the active layer of the active track to 1.0 (5x fast forward)

```
/mxw/track/active/layer/active/clip/speed 0.5
```
Sets the speed of the clip of the active layer of the active track to 0.5 (clip stops)

```
/mxw/track/active/layer/active/clip/speed 0.6
```
Sets the speed of the clip of the active layer of the active track to 0.6 (normal clip speed)

```
/mxw/trackmanager 0.2
```
Activate the first track

```
/mxw/trackmanger 0.7
```
Activate the third track

```
/mxw/track/2/layer/1/clip effect 0.1
```
Instructs the clip of the second layer of the upper track to load the first of ten effects

```
/mxw/track/2/layer/1/clip/effect/1/param/1 0.8
```
Sets the effect parameter 0 of the first effect of the clip of the first layer of the upper track to the value 0.8

```
/mxw/preload/2 c:/footage/clip.avi
```
Preload slot 2 will load c:/footage/clip.avi (and answer with clipinfo). Note the direction of the slashes.

```
/mxw/preload/2 clear
```
Preload slot 2 will unload

```
/mxw/preload/2 clipinfo
```
Preload 2 will answer with clip info like
```
/mxw/reply /mxw/preload/3 112 320 240 40 cachecomplete
c:\footage\clip.avi
```

```
/mxw/preload/2 MSGID0123 clipinfo
```

Preload 2 will answer with clip info like
```
/mxw/reply MSGID0123 /mxw/preload/3 112 320 240 40 cachecomplete
c:\footage\clip.avi
```

```
/mxw/fixture/10 load 2
```
Load content of preload 2 in fixture 10 (second slot of third track. every track has four slots)

```
/mxw/fixture/10 load c:/footage/clip.avi
```
Fixture 10 will load clip.avi (not recommended, use preloads instead.)

```
/mxw/fixture/10 load -1
```
Unload fixture 10

```
/mxw/fixture/10/mode 0.5
```
Load corresponding layer mode

```
/mxw/fixture/10/clip/speed 0.5
```
Stop fixture 10

```
/mxw/fixture/10/clip/effectlist 0.1
```
Load corresponding effect

```
/mxw/fixture/10/clip/effect/1/param/1 0.1
```
Control corresponding parameter

```
/mxw/master/load_showfile c:/footage/showfile.mxw
```
Load the stated showfile

```
/mxw/master//mxw/master/load_keystonefile
c:/footage/keystonefile.mxw_keystone2
```
Load the stated keystonefile

Sending OSC

MXW can send OSC commands, too. There are two distinct options:

1. Send OSC: the 'boygroup mode'
   This option will send any action on the running host to another host/network as stated in the command options. The osc commands are sent exactly as they would generate the same behaviour on the master machine. So eg. moving a track slider will generate the commands that make a remote move to the same position. The goal is to make another machine (pretended it carries the same media data) produce exactly the same result as the master machine.

   Sometimes it is desireable to send only a selction af the generated commands. You can filter the sent commands using the following rules:
   - Adding a + before a 'token'
   - Adding a - before a 'token'
   'token' is a part of the string like 'playlist' or 'opacity'. So eg. adding the filter

   +,*playlist*

   will result into that only commands containing 'playlist' will be sent

2. Broadcast and Receive Time over OSC
   This option will broadcast the MXW time into the selected subnet with the goal of synchronizing the time amoung a pool of machines.
   1. There is a selection mechanism: every machine sends its time until it receives the time from a machine with a lower IP adress. In the end a 'master' will be selected – the machine with the lowest IP – that controls the pool time.
   2. IO Events can be forwarded, too. You can eg. attach a MIDI device to a system and it will send the MIDI events to the pool

1 http://www.cnmat.berkeley.edu/OpenSoundControl/
2 OSC resource: http://www.opensoundcontrol.org