WING

PERSONAL MIXING CONSOLE

OSC Remote Control Documentation
for WING

[Wing FW 1.13.6 and above]

Table of Contents

Introduction 5
General features of the WING console 5
Sources vs. Inputs 7
WING Internal Data 8
WING File System 8
OS partition 9
Data Partition 9

Remote communication with WING 10
Number of simultaneously connected applications 10
Access to WING Internal Data from remote programs 10

OSC Remote Protocol 12
OSC Data Types 12
WING OSC Messages 13
Reading (Get) Parameter and Node data 13
Receiving OSC data on a specific port 14
Writing (Set) Parameter and Node data 15
Single Parameters 15
Enumerated strings 15
Node Data 16
Special Node Type/Arguments 17
OSC: Special Cases 19
Dynamic JSON Structure changes 19
OSC Tag Type 'blob' use 20
Subscribing to OSC Data 23
Effects and Plugins 24
Plugins 24
Effects 25
WING ae_data OSC commands list 27
Status 27
General Configuration 28
System Settings 32
Input/Output Settings 33
Channel Settings 39
Aux Settings 43
Bus Settings 45
Mains Settings 48
Matrix Settings 52
DCA Settings 55
Mutegroup Settings 55
Effects Settings 56
Cards Settings 57
USB Player Settings 59
Global Settings 59
WING ce_data OSC commands list 61
Control Settings 61

Appendix: Buttons (user/gpio, user/user, user/daw, user/) 73
user/gpio 73
user/user 73

user/daw1..4 73
user/1..16/..4 73
Appendix: Effects and Plugins' Parameters list 77
Effects 77
Standard effects 77
Premium effects 85
Channel effects 92
Plugins 97
Filter plugins 97
Gate plugins 98
EQ plugins 100
Compressor plugins 103
Appendix: WING Icons 107
Appendix: WING Colors 109
Appendix: WING GPIOs: 110
Appendix: MCU [DAW BUTTONS] commands list 111
Appendix: MCU [DAW V-POTS] commands list 112
Appendix: MCU [DAW REMOTE MCU] commands list 113
Appendix: WING Snapshot and JSON Data Structure: 114
Global Snapfile 114
Description 114
scopes 115
ae_data 116
ce_data 130
More JSON files 133

Introduction

Introduction

My name is Patrick-Gilles Maillot and I am authorized by Behringer to publish and maintain the "OSC Remote Control Documentation for WING", yet I am not a MusicTribe employee.

In 2019, Behringer has been designing a whole new digital mixing desk they would later call "personal mixing console". The WING was unveiled to the general public in November 2019 and first shipments took place in December. As to why calling it a "personal Mixing Console", here is a perfectly valid answer from one of the fathers of the console: "A fundamental idea of WING was providing a high level of customization options to the engineer, allowing to adapt the console surface to his personal preferences and needs".
The WING console was awaited by several X32 and M32 users as it carried the promise of new features, long expected since the first release the X32 and M32 family of digital mixing desks. It seems the WING receives a warm welcome from the community.

General features of the WING console

The Behringer WING provides 48-channel, 28-bus mixing with 24 motorized faders and a large 10" capacitive-touch LED screen. The desk is designed for live performance, live and studio recording, touring sound, A/V, club installs, and more. Three separate fader sections and a custom controls section can be easily and intuitively tailored to personal requirements.
Historically, consoles focus on input numbers assigned to channels and auxes. WING is offering a substantially different perspective by focusing on the Source as the reason for any mixing, say a bass drum signal or the lead vocal. Inputs are therefore given more properties than just a number. Sources can be in mono, stereo, or mid-side[1] mode, own headamp parameters like gain and phantom power, with specific source mute and metering. They can be given a color, icon, name and up to 8 console or user defined tags for grouping or filtering purposes. All of this describes the actual Source first, before channels are used for processing or mixing.
The 48-channel inputs [in/aux] and 28-channel mixes [bus/matrix/main] can all be in mono/stereo or mid-side mode, with specific source mutes and metering, and provide dynamics, EQ and FX processing. They too can be given a color, icon, name and up to 8 console or user defined tags for grouping and filtering purposes.
WING input channels provide low-cut & high-cut filters, tilt-EQs, all-pass or Sound Maxer, in addition to a 6-band parametric EQ. All buses, matrices, and mains feature 8-band parametric EQ. All channels and buses can also load high-end simulations modeled from hardware devices such as Pultec EQ, SSL Bus Compressor and Gate/Expander, SPL Transient Designer, Neve EQ, Compressor and Gate, Focusrite ISA and D3, DBX160, LA-2A, 1176, Elysia mPressor, Empirical Labs Distressor, and more. The built in FX rack supports 8 true stereo processors including TC VSS3 algorithms, Lexicon, Quantec, and EMT emulations. Other processing includes modulation, equalization, dynamics, nonlinear effects and four guitar amplifiers with cabinet simulations. A maximum of 16 stereo inserts can be used for applying internal FX or outboard processing to input channels or buses.
The channel editing section provides instant channel status overview and flow of operation. It allows working on the selected channel processing, even when the main display is used for something

[1] Mid/Side processing is a highly effective way of making adjustments to the spatialization perception of a mix or master. The Mid channel is the center of a stereo image. When the Mid channel is boosted, the listener perceives a more centered (mono) sound to the audio. The Side channel is the edges of a stereo image. When the Side channel is boosted, the listener perceives a more spacious (wider) sound to the audio.

completely unrelated. Touch-sensitive rotary controls allow you to display the most relevant information, all at your fingertips.
The central Custom Controls section offers user-assignable controls including 4 rotary encoders and 20 buttons with 2 LCDs that can be set as functions readily available. A big rotary wheel offers fine-adjustments for up to 8 user parameters or can be used for DAW remote control via USB MIDI. The control configuration also includes predefined functionality for USB and SD-card recorder transport, show control and mute groups.
WING includes 8 original MIDAS PRO microphone preamps and 8 XLR outputs with professional quality specifications. 8 TRS line auxiliary ins and outs help bring in signals from media players or computers. A brand new StageCONNECT[2] interface allows connecting breakout boxes and delivers up to 32 channels of low-latency input or output over a single standard XLR microphone cable.

WING can accommodate 374 inputs and 374 outputs thanks to 3 AES50 SuperMAC audio networking ports, which connect to digital stageboxes. In addition, 144 input and 144 output streams can be shared with other mixing consoles. There are 48 channels of USB audio and 64 channels of Audio over IP (AoIP module optional), plus AES/EBU stereo I/O. The WING expansion card slot features the LIVE SD recording card with 64x64 channels of audio or can accommodate option cards for various standards such as ADAT, MADI, DANTE, and WSG.
All digital processing takes place on 40-bit floating point Digital Signal Processors, at 48 or 44.1 kHz, with a 1ms round-trip latency.
WING provides MIDI In/Out and 2x2 GPIO (General Purpose Input Output) that can be used as console event triggers and external show controls.
Automixing is also implemented, with 2 groups of gain sharing on any 16 input channels. The management of the respective input channel gains depends on the levels received, reducing the sum gain in the group to maintain intelligibility and low noise during meetings, ideal when several speakers are collaborating to corporate events, panels, broadcast applications or house of worship.

2 https://www.klarkteknik.com/series.html?category=R-KLARKTEKNIK-STAGECONNECTSERIES

Sources vs. Inputs

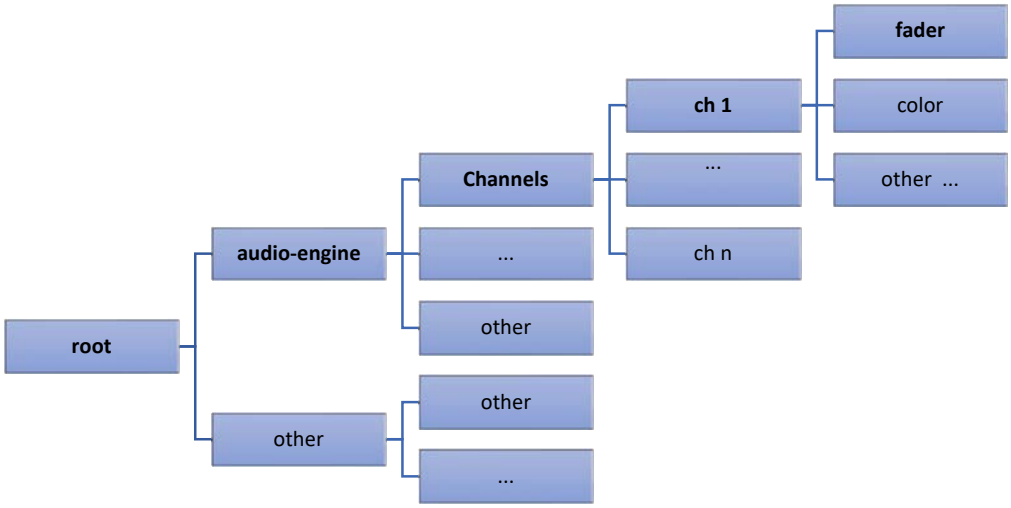
Unlike many digital or analogue desks, WING makes a clear separation between Sources and Input channels; Normally, consoles focus on input numbers assigned to Channels and Auxes.
WING is offering a different perspective by focusing on the Source as the reason for any mixing. Sources can be in mono, stereo or mid-side mode, own headamp parameters like gain and phantom power, with specific source mute and metering. They can be given a color, icon, name and up to 8 console or user defined tags for grouping and filtering purposes. All of this describes the actual Source first, before being patched to Input channels which focus on processing or mixing.
Sources can be labelled using the WING Co-Pilot app or other means such as **OSC** protocol described later in this document, or the **wapi** function calls[3], and no matter if the signal is patched to a channel, to SD recording or to any other output, it can always be referred to as its assigned Source label.

Notes
The internal real-time clock (RTC) is powered by a super-capacitor. If the WING is powered off for more than about two weeks it will most likely lose its clock data.

[3] Described in a separate document. Refer to https://github.com/pmaillot/wapi

WING Internal Data

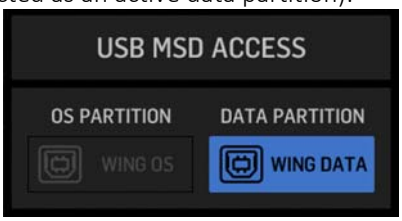
Like all digital or programmable devices, WING relies on an internal set of parameters that are stored/saved in non-volatile memory. This enables you to find the console in the same state you left it when powering it OFF.
WING data set is very large, and in line with the many features the console offers. Each button, each attribute, color setting, effect, parameter, etc. can be found as an internal variable, member of a hierarchical tree structure.
The WING tree is more than 25000 elements! In order to organize this large set of internal variables, WING uses a hierarchical tree of data, stating with a root and dispatching parameters into logical groups (sub-trees or branches) until the last element (leaves) that represent the actual parameter.
For example, the `fader` associated to `channel 1` is part of the `channels` sub-tree, and is one of the many attributes of `channel 1`. The channel sub-tree is part of the `audio-engine`, itself at the `root` level.
A quick representation would be as shown below:



Computers use specific data structures to represent trees. WING uses one of them, based on JSON[4] notation. It is important to know/understand the list of sub-trees (nodes), and leaves (parameters) WING contains as this is how you gain access to data. More detail on the WING data set is provided in appendix.

WING File System

At the difference of the X32, WING can be directly connected to a computer via USB; There are two ways WING can be visible to your computer, depending on the setting of the `SETUP→GENERAL` screen (shown below, with WING connected as an active data partition):



[4] JavaScript Object Notation: an efficient way to represent structured objects. Also used as a data-interchange format.

When actively connected to your PC either as an OS partition or a Data partition, the status at the top of the WING screen will show a red MSD tag.

OS partition

WING can be seen as an `OS PARTITION`, a directory where you can deposit the FW release you will use to boot from at next power up or reboot. Use with caution!

Data Partition

A USB connected WING presents itself as an external disk drive. Therefore, the standard cautions apply when connecting and more important, disconnecting from the computer; ***Ensure you unmount the WING file system to avoid losing data.***

If the choice for `USB MSD ACCESS` in `SETUP→GENERAL` is set to `DATA PARTITION`, the WING file system will show as a standard external USB drive. There may be some folders already there, such as 'global' or 'shows', with subfolders, such as 'global/ch_presets', 'global/fx_presets', 'global/routing_presets', 'global/snapshots'.

Remote communication with WING

WING communicates via ports 2222 [native UDP, TCP] and 2223 [OSC, UDP];
Initiating a communication with WING starts with sending the 5 bytes [UDP] datagram 'WING?' to the IP of your WING, port 2222.
WING will reply to the requesting IP and port with the following datagram:

```
'WING,' [c_ip] ',' [c_name] ',' [c_model] ',' [c_serial] ',' [firmware]
```

where

```
[c_ip]          e.g. '192.168.1.62'
[c_name]        ascii characters
[c_model]       'ngc-full' (standard Wing console)
[c_serial]      serial number (ascii)
[firmware]      version string (ascii)
```

From there on, general OSC communications take place over communication port 2223

Number of simultaneously connected applications

WING can simultaneously communicate with up to 16 **connected** 'clients'; The console will reject further connection requests, if the maximum number of simultaneous connections (16) is reached. What we call 'clients' above refer to actual TCP ports that communicate with the console. Some applications may use several ports and this will reduce the actual number of applications that can simultaneously connect and communicate with WING.
UDP communications such as used for OSC do not have this limitation, being "connection-less". WING's OSC remote protocol enables **only one** (1) subscription to data (for receiving event messages) at any given time. Subscriptions must be kept alive; they automatically die after 10 seconds.

Access to WING Internal Data from remote programs

WING offers several remote protocols with the capability to access (read or write) parameters of its internal structures and take full advantage of the numerous features of the digital desk, including remote control. One of them is WING's native (binary) interface and is covered in a separate document.

This document focuses on **OSC**.

WING hosts an **OSC** compliant remote protocol server that offers access to the full set of features of the desk.

WING OSC protocol data interface

OSC Remote Protocol

WING includes an OSC Remote Protocol server. This enables easy access to remote features for many professional, sound applications and extensions offered by third parties.
OSC remote control enables reading and modifying (when possible) all parameters included in the `ae_data` and `ce_data` JSON structures, all part of the main parameter tree.

WING OSC server implementation complies with the OSC standard[5] and proposes several ways to access data, parameters, and features. As all OSC compliant servers, the WING OSC server runs in the console and will reply to UDP on a specific port: `2223`.
When using standard UDP communication, clients will be replied onto their calling port. If needed, a specific feature enables WING to reply to a UDP port specified by the connected client, as explained later in this document.

OSC Data Types

In compliance with the OSC standard, WING supports the following types:

- `int32` (32bits, bi-endian),
- `float32` (32bits, IEEE 754, big endian),
- `string` (non-null ASCII characters followed by a null, followed by 0-3 additional null characters to make the total number of bytes a multiple of 4),
- `blob` (An int32 size count, followed by that many 8-bit bytes of arbitrary binary data, followed by 0-3 additional zero bytes to make the total number of bytes a multiple of 4).

As specified in the `OSC` standard, the unit of transmission of `OSC` is an `OSC Packet`. Any application that sends `OSC Packets` is an `OSC Client`; WING embeds and runs an `OSC Server`.

An `OSC Packet` consists of its contents, a contiguous block of binary data, and its size, the number of 8-bit bytes that comprise the contents. The size of an `OSC packet` is always a multiple of 4.
In the case of WING, the content of an `OSC packet` is always an `OSC Message`, i.e., `OSC Bundles` are not supported. Note that wildcards '`?`' and '`*`' in Address Patterns are reserved for special cases.
An `OSC Message` consists of an `OSC Address Pattern` followed by an `OSC Type Tag String` followed by zero or more `OSC Arguments`. Some older implementations of `OSC` may omit the `OSC Type Tag` string and WING supports this.

- `OSC Address Patterns` always start with the character '`/`'.
- `OSC Type Tags` can be `i, f, s, b` for int32, float32, string and blob, respectively
- `OSC Arguments` consist in a single or a contiguous sequence of the binary representations of each argument

The maximum UDP packet size is 32k bytes.

[5] See http://opensoundcontrol.org/spec-1_0

WING OSC Messages

In the following paragraphs, we assume a communication link exists between WING and a client program, and communication takes place with a WING console at a known IP address, using UDP on port `2223`.
All along this document, the character `'~'` will represent a NULL byte (`\0`). Patterns `->W` and `W->` represent data sent to WING and data received from WING followed by the actual number of bytes transmitted or received, respectively.

Retrieving WING console information can be completed by sending the OSC Address Pattern "`/?`"

```
->W,   4 B: /?~~
W->,  80 B:
/?~~,s~~WING,192.168.1.71,PGM,ngc-full,NO_SERIAL,1.07.2-40-g1b1b292b:develop~~~~
```

The actual bytes exchanged are displayed below (OSC is a binary protocol)

```
->W,   4 B: 2f3f0000
W->,  80 B:
2f3f00002c73000057494e472c3139322e3136382e312e37312c50474d2c6e67632d66756c6c2c4e4f5f5345524
9414c2c312e30372e322d34302d673162316232393932623a646576656c6f7000000000
```

The line below is using a more compliant OSC format, and will result in the same answer

```
->W,   8 B: /?~~,~~~
```

Reading (Get) Parameter and Node data

There are two main ways to gain access to WING data: using one-parameter-at-a-time or using "nodes".
WING "nodes" are a great way to access multiple parameters at a time, and therefore maximize communication bandwidth with the console. Nodes are represented as **string** OSC Data Type and are zero terminated (`\0` byte ending the string).
Nodes are also a good way to discover WING parameters, as they offer easy access to the full map of the JSON internal data structures.
We show below WING's first layer of JSON structure, and starting at the root, retrieved using OSC.

```
->W,   4 B: /~~~
W->, 116 B:
/~~~,ssssssssssssssss~~~$stat~~~cfg~$syscfg~io~~ch~~aux~bus~main~~~~mtx~dca~mgrp~~~~fx~~car
ds~~~play~~~~rec~$ctl~~~~
```

Retrieving a WING single parameter is quite easy: You must ensure your OSC request points to a leaf of the JSON structure (i.e., there is no more hierarchy data after the current one). This is the case for the fader value of a channel strip for example, or its mute state. Channel Strip 1 fader is represented as follows:

Or “ch”/”1”/”fdr”, which translates to OSC Address Pattern /ch/1/fdr:

```
->W,   12 B: /ch/1/fdr~~~
W->,   32 B: /ch/1/fdr~~~,sff~~~~-oo~[0.0000][-144.0000]
```

In the example above, the data [0.0000][-144.0000] are ascii interpretations of two 32bits big-endian float data values, each represented on 4 bytes as binary. The binary data actually received is as shown below, and in order to ease the reading of numerical information in this document, we use readable values in brackets rather than the actual binary data. The color highlights are there to help distinguish data elements.

```
W->,   32 B: 2f63682f312f666472000000 2c73666600000000 2d6f6f0000000000 c3100000
```

Depending on the OSC Address Pattern, WING returns ',s' for strings or enums, ',sff' (ascii, raw pos, float value) for floats, ',sfi' (ascii, raw pos, int value) for ints. In the example above, fader position is a float and WING returns the ascii representation, the raw [0.0..1.0] data and the actual float value in dB.

Similarly, requesting the mute state of channel strip 1 would return:

```
->W,   12 B: /ch/1/mute~~
W->,   32 B: /ch/1/mute~~,sfi~~~~1~~~[1.0000][     1]
W->,   32 B: 2f63682f312f6d7574650000 2c73666900000000 31000000 3f800000 00000001
```

It should be noted that WING will accept both OSC path or the native hash data for representing nodes or parameters; Indeed, all nodes and parameters in the console are assigned a binary address (a hash) as explained in the chapter on native interface to the console. For example, the channel 1 mute command above can be sent as OSC Address Patterns /ch/1/mute~~ as shown, or /#f50f69f8~~, and would return the same data as shown above. 0xf50f69f8 is the hash for command “Channel 1 mute”. The full set of WING hash values can be discovered by recursively traversing the JSON tree of WING nodes/commands, using the native binary interface or OSC protocol, but it is generally more convenient to use the more standard OSC node notation, rather than hexadecimal hash values to address the console features.

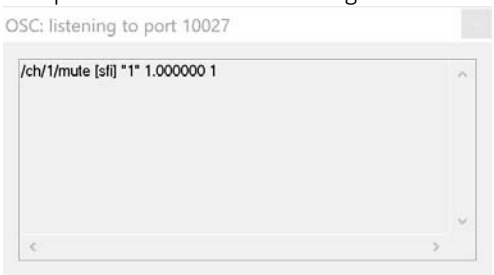
Receiving OSC data on a specific port

Some OSC programs will request that data is returned on a specific port rather than being sent back to the port used by the requesting client for sending data. In order to enable this capability, WING OSC includes an optional, special notation for all OSC commands:

Any OSC command can be prefixed with the /%<port>, with <port> in the form “12345” to enable receiving the expected answer onto the specified port number. For example, the OSC request:

```
->W,   20 B: /%10027/ch/1/mute~~~
```

Will receive the expected reply from WING on port 10027, as shown below, using a sniffer program on said port. The IP does not change.



Writing (Set) Parameter and Node data

Single Parameters

OSC can be used to set or modify WING data. Taking the fader and mute examples above, we can modify their respective values using OSC commands, sending string, big-endian int32 or big-endian float32 with the corresponding OSC Type Tag following the OSC Address Pattern respective of the parameter to change.
WING does not echo data sent over UDP by the client application. The client application may nevertheless be notified with an OSC event in case of an error.
Individual parameters can be strings, integer, or floats; WING OSC server implementation enables to use several data types and will manage the conversion to ensure proper value setting inside the console. For example, fader position is a floating-point internal value. It can be set as a string or a float using the following OSC commands (in this example setting channel 2 fader position to -2 or -3dB):

```
->W,   20 B: /ch/2/fdr~~~,s~~-2~~
->W,   12 B: /ch/2/fdr~~~
W->,   36 B: /ch/2/fdr~~~,sff~~~~-2.0~~~~[0.7000][-2.0000]

->W,   20 B: /ch/2/fdr~~~,f~~[-3.0000]
->W,   12 B: /ch/2/fdr~~~
W->,   36 B: /ch/2/fdr~~~,sff~~~~-3.0~~~~[0.6750][-3.0000]
```

Enumerated strings

One of the data WING uses is “enumerated strings”, or the choice of one string in a list of elements to represent a specific state or attribute value. For example, `/$ctl/user/1/1/enc/mode` can be any of the following strings: `OFF, FDR, PAN, DCA, SSND, FSND, FX, DAWMCU, SD A`, or `SD B`

This can be set via a string OSC tag, as shown below if one wants to set the `mode` parameter to `FX`:

```
/$ctl/user/1/1/enc
->W,    20 B: /$ctl/user/1/1/enc~~
W->,    52 B: /$ctl/user/1/1/enc~~,sss~~~~mode~~~~name~~~~$fname~~
->W,    24 B: /$ctl/user/1/1/enc/mode~
W->,    32 B: /$ctl/user/1/1/enc/mode~,s~~OFF~
/$ctl/user/1/1/enc/mode ,s FX
->W,    32 B: /$ctl/user/1/1/enc/mode~,s~~FX~~
/$ctl/user/1/1/enc/mode
->W,    24 B: /$ctl/user/1/1/enc/mode~
W->,    32 B: /$ctl/user/1/1/enc/mode~,s~~FX~~
```

But it can also be set as an int OSC tag, using the index of the list corresponding to the targeted value; in the example above, `FX` sits at index `6` in the list of `10` strings; This enables us to use the following OSC command to set the encoder mode to `FX`:

```
/$ctl/user/1/1/enc
->W,    20 B: /$ctl/user/1/1/enc~~
W->,    52 B: /$ctl/user/1/1/enc~~,sss~~~~mode~~~~name~~~~$fname~~
->W,    24 B: /$ctl/user/1/1/enc/mode~
W->,    32 B: /$ctl/user/1/1/enc/mode~,s~~OFF~
/$ctl/user/1/1/enc/mode ,i 6
->W,    32 B: /$ctl/user/1/1/enc/mode~,i~~[     6]
/$ctl/user/1/1/enc/mode
->W,    24 B: /$ctl/user/1/1/enc/mode~
W->,    32 B: /$ctl/user/1/1/enc/mode~,s~~FX~~
```

One can also note the extendibility character of WING nodes; indeed, after the previous command, the user 1/1 encoder has additional parameters:

```
/$ctl/user/1/1/enc
->W,    20 B: /$ctl/user/1/1/enc~~
```

```
W->,   60 B: /$ctl/user/1/1/enc~~,sssss~~mode~~~~name~~~~$fname~~fx~~par~
```

Node Data

WING nodes can also be used to set multiple values with using a single OSC "/" command, and offer a simple yet effective way to navigate within the hierarchical structure of JSON data. Say you want/need to set fader and mute values to -1 dB, 0 dB, OFF and ON for channels 1 and 2; This can be achieved in a single OSC request using the following syntax:

```
->W,  44 B: /~~~,s~~/ch.1.fdr=-1,mute=0,.2.fdr=0,mute=1~
```

Or setting channel 1 fader and mute values to 10 dB and ON, and setting bus 1 fader to 5 dB:

```
->W,  44 B: /~~~,s~~/ch.1.fdr=10,mute=1,/bus.1.fdr=5~~~~
```

As shown above, each parameter group is separated by a ',' character, the '/' character represents the root of the JSON parameter tree, and '.' characters are used to navigate up and down within the JSON parameter tree.

The console will reply with `/*~~,s~~OK~~` if the command was accepted, or one of the following:

```
/*~~,s~~NODE NOT FOUND~~
/*~~,s~~VALUE ERROR~~~~~
/*~~,s~~BUFFER OVERFLOW~
/*~~,s~~NODE IS NOT PAR~
/*~~,s~~INCOMPLETE DATA~
/*~~,s~~STACK EMPTY~~~~~
```

if an error occurred during the execution of the command.

Note: Nodes can return large amounts of data; as a result, some nodes cannot be returned using `OSC/UDP` as they would overflow the `32kB UDP` buffer limitation; In such situation, WING will return an error OSC message event.

Some nodes examples are provided below:

```
->W,  12 B: /ch/1/fdr~~~
W->,  32 B: /ch/1/fdr~~~,sff~~~~-oo~[0.0000][-144.0000]
->W,  12 B: /ch/1/mute~~
W->,  32 B: /ch/1/mute~~,sfi~~~~1~~~[1.0000][     1]
->W,  12 B: /ch/2/fdr~~~
W->,  32 B: /ch/2/fdr~~~,sff~~~~-oo~[0.0000][-144.0000]
->W,  12 B: /ch/2/mute~~
W->,  32 B: /ch/2/mute~~,sfi~~~~0~~~[0.0000][     0]

->W,  44 B: /~~~,s~~/ch.1.fdr=-1,mute=0,.2.fdr=0,mute=1~
W->,  12 B: /*~~,s~~OK~~

->W,  12 B: /ch/1/fdr~~~
W->,  36 B: /ch/1/fdr~~~,sff~~~~-1.0~~~~[0.7250][-1.0000]
->W,  12 B: /ch/1/mute~~
W->,  32 B: /ch/1/mute~~,sfi~~~~0~~~[0.0000][     0]
->W,  12 B: /ch/2/fdr~~~
W->,  32 B: /ch/2/fdr~~~,sff~~~~0.0~[0.7500][0.0000]
->W,  12 B: /ch/2/mute~~
W->,  32 B: /ch/2/mute~~,sfi~~~~1~~~[1.0000][     1]
```

Nodes can also be located deeper in the JSON structure tree. For example, changing a single parameter in the node channel 1 ["`/ch/1`"] can be done as shown below:

```
->W,  20 B: /ch/1~~~,s~~fdr=3~~~
W->,  16 B: /ch/1*~~,s~~OK~~

->W,  12 B: /ch/1/fdr~~~
W->,  32 B: /ch/1/fdr~~~,sff~~~~3.0~[0.8250][3.0000]
```

```
->W,  12 B: /ch/1/mute~~
W->,  32 B: /ch/1/mute~~,sfi~~~~0~~~[0.0000][     0]
```

The OSC command is replied to with an OK status if execution went well; error messages can be returned too, as explained earlier.

The same type of command can be used to set/change several parameters at once; For example, fader and mute values of channel 1 can be done as follows:

```
->W,  28 B: /ch/1~~~,s~~fdr=4,mute=1~~~~
W->,  16 B: /ch/1*~~,s~~OK~~

->W,  12 B: /ch/1/fdr~~~
W->,  32 B: /ch/1/fdr~~~,sff~~~~4.0~[0.8500][4.0000]
->W,  12 B: /ch/1/mute~~
W->,  32 B: /ch/1/mute~~,sfi~~~~1~~~[1.0000][     1]
```

Special Node Type/Arguments

There are three special tag/argument that are specifically implemented for nodes. They enable listing the complete set of data, parameter description, and description including values for the node provided as OSC address pattern. The arguments to use are '`*`', '`?`', and '`#`', respectively. Examples of use are provided below, applied to OSC address pattern `/fx/1` when no effect is loaded in order to keep the description as short as possible.

Node data dump:

When using this format, The data returned will strictly correspond to what would be saved in a snap file; Read-only and temporary data are not returned.

```
/fx/1 ,s *
->W,   16 B: /fx/1~~~,s~~*~~~
W->,   32 B: /fx/1~~~,s~~mdl=NONE,fxmix=100,~
```

Node parameter description:

```
/fx/1 ,s ?
->W,   16 B: /fx/1~~~,s~~?~~~
W->,  696 B: /fx/1~~~,s~~  mdl            list [NONE, EXT, HALL, ROOM, CHAMBER, PLATE,
CONCERT, AMBI, V-ROOM, V-REV, V-PLATE, GATED, REVERSE, DEL/REV, SHIMMER, SPRING, DIMCRS,
CHORUS, FLANGER, ST-DL, TAP-DL, TAPE-DL, OILCAN, BBD-DL, PITCH, D-PITCH, VSS3, BPLATE, GEQ,
PIA, DOUBLE, PCORR, LIMITER, DE-S2, ENHANCE, EXCITER, P-BASS, ROTARY, PHASER, PANNER, TAPE,
MOOD, SUB, RACKAMP, UKROCK, ANGEL, JAZZC, DELUXE, BODY, SOUL, E88, E84, F110, PULSAR,
MACH4, C5-CMB, SUB-M, V-IMG, SPKMAN, DEQ3, *EVEN*, *SOUL*, *VINTAGE*, *BUS*, *MASTER*]~
fxmix          lin [0 .. 100 %], 101 steps~  $esrc          int [0 .. 400]~  $emode
list [M, ST, M/S]~  $a_chn         int [0 .. 76]~  $a_pos         int [0 .. 1]~~~~
```

Node description including values:

```
/fx/1 ,s #
->W,   16 B: /fx/1~~~,s~~#~~~
W->,  816 B: /fx/1~~~,s~~  mdl            NONE                list [NONE, EXT, HALL, ROOM,
CHAMBER, PLATE, CONCERT, AMBI, V-ROOM, V-REV, V-PLATE, GATED, REVERSE, DEL/REV, SHIMMER,
SPRING, DIMCRS, CHORUS, FLANGER, ST-DL, TAP-DL, TAPE-DL, OILCAN, BBD-DL, PITCH, D-PITCH,
VSS3, BPLATE, GEQ, PIA, DOUBLE, PCORR, LIMITER, DE-S2, ENHANCE, EXCITER, P-BASS, ROTARY,
PHASER, PANNER, TAPE, MOOD, SUB, RACKAMP, UKROCK, ANGEL, JAZZC, DELUXE, BODY, SOUL, E88,
E84, F110, PULSAR, MACH4, C5-CMB, SUB-M, V-IMG, SPKMAN, DEQ3, *EVEN*, *SOUL*, *VINTAGE*,
*BUS*, *MASTER*]~  fxmix          100                 lin [0 .. 100 %], 101 steps~  $esrc
0              r/o int [0 .. 400]~  $emode         M              r/o list [M, ST, M/S]~
$a_chn         0              r/o int [0 .. 76]~  $a_pos         0              r/o int
[0 .. 1]~~~~
```

As a second example, we give below the node data dump for OSC address pattern `/ch/1`, when loaded with default values after init:

```
/ch/1 ,s *
->W,   16 B: /ch/1~~~,s~~*~~~
W->, 1972 B:
/ch/1~~~,s~~in.set.srcauto=0,altsrc=0,inv=0,trim=0.0,bal=0.0,dlymode=M,dly=0.0,dlyon=0,.con
n.grp=LCL,in=1,altgrp=OFF,altin=1,..flt.lc=0,lcf=100.2,hc=0,hcf=10k02,tf=0,mdl=TILT,tilt=0.
00,.clink=0,col=1,name=,icon=1,led=1,mute=0,fdr=-oo,pan=0,wid=100,solosafe=0,mon=A,proc=GED
I,ptap=5,peq.on=0,1g=0.0,1f=100,1q=1.00,2g=0.0,2f=999,2q=1.00,3g=0.0,3f=10k0,3q=1.00,.gate.
on=0,mdl=GATE,thr=-40.0,range=40.0,att=10,hld=10,rel=199,acc=0,ratio='1:3',.gatesc.type=OFF
,f=1k0,q=2.00,src=SELF,tap=IN,.eq.on=0,mdl=STD,mix=100,lg=0.0,lf=80.2,lq=1.00,leq=SHV,1g=0.
0,1f=200.0,1q=1.00,2g=0.0,2f=601.4,2q=1.00,3g=0.0,3f=1k50,3q=1.00,4g=0.0,4f=3k99,4q=1.00,hg
=0.0,hf=11k99,hq=1.00,heq=SHV,.dyn.on=0,mdl=COMP,mix=100,gain=0.0,thr=-10.0,ratio=3.0,knee=
3,det=RMS,att=50,hld=20,rel=153,env=LOG,auto=1,.dynxo.depth=6.0,type=OFF,f=1k0,.dynsc.type=
OFF,f=1k0,q=2.00,src=SELF,tap=IN,.preins.on=0,ins=NONE,.main.1.on=1,lvl=0.0,pre=0,.2.on=0,l
vl=0.0,pre=0,.3.on=0,lvl=0.0,pre=0,.4.on=0,lvl=0.0,pre=0,..send.1.on=0,lvl=-oo,pon=0,ind=0,
mode=PRE,plink=0,pan=0,wid=100,.2.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.
3.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.4.on=0,lvl=-oo,pon=0,ind=0,mode=
PRE,plink=0,pan=0,wid=100,.5.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.6.on=
0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.7.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,p
link=0,pan=0,wid=100,.8.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.9.on=0,lvl
=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.10.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink
=0,pan=0,wid=100,.11.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.12.on=0,lvl=-
oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.13.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0
,pan=0,wid=100,.14.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.15.on=0,lvl=-oo
,pon=0,ind=0,mode=PRE,plink=0,pan=0,wid=100,.16.on=0,lvl=-oo,pon=0,ind=0,mode=PRE,plink=0,p
an=0,wid=100,..postins.on=0,mode=FX,ins=NONE,w=0.0,.tags=,~~~
```

OSC: Special Cases

Dynamic JSON Structure changes

As parameters get changed on the WING console, its JSON structure tree evolves to reflect the change; This can be a specific parameter that when changing to an `ON` state, offers new capabilities in the audio chain, or in the way the console will react.
It is also typical of **effects** and **plugins**: WING consoles support the dynamic allocation of effect or plugins, generating large changes within the default JSON tree. As already mentioned, WING nodes are a great way to list the parameters available for a given effect and therefore be able to get and possibly set effect parameter values.
The WING effects and plugins, and their respective parameters are listed later in this document[6].

The OSC commands below show how you can access effects slots, allocate an effect, and list parameters and later modify effect parameter values.
Accessing effects with currently no effect loaded in effect slot 1, listing the effect Node:

```
->W,    4 B: /fx~
W->,   88 B:
/fx~,sssssssssssssss~~~1~~~2~~~3~~~4~~~5~~~6~~~7~~~8~~~9~~~10~~11~~12~~13~~14~~15~~16~~
->W,    8 B: /fx/1~~~
W->,   60 B: /fx/1~~~,ssssss~mdl~fxmix~~~$esrc~~~$emode~~$a_chn~~$a_pos~~
->W,   12 B: /fx/1/mdl~~~
W->,   24 B: /fx/1/mdl~~~,s~~NONE~~~~
```

Loading a PIA effect in effect slot 1:

```
->W,   20 B: /fx/1/mdl~~~,s~~pia~
->W,   12 B: /fx/1/mdl~~~
W->,   20 B: /fx/1/mdl~~~,s~~PIA~
```

PIA effect is now loaded, listing the effect Node gives a different set of parameters:

```
->W,    8 B: /fx/1~~~
W->,  120 B:
/fx/1~~~,ssssssssssssssss~mdl~fxmix~~~$esrc~~~$emode~~$a_chn~~$a_pos~~mix~g~~~31~~63~~125
~250~500~1k~~2k~~4k~~8k~~16k~
```

We can now get/set effect 1 PIA parameters, for example the 125Hz band:

```
->W,   12 B: /fx/1/125~~~
W->,   32 B: /fx/1/125~~~,sff~~~~0.0~[0.5000][0.0000]
```

The 125Hz band is at 0dB, change it to 10dB and verify the change:

```
->W,   20 B: /fx/1/125~~~,f~~[10.000]
->W,   12 B: /fx/1/125~~~
W->,   36 B: /fx/1/125~~~,sff~~~~10.0~~~~[0.9233][10.000]
```

[6] Please refer to the “Effects” paragraph

OSC Tag Type 'blob' use

WING OSC server implementation supports the 'blob' OSC Tag type, enabling the use of 'native' commands[7] within OSC, making it is possible with the proper information at hand to send and receive binary data.

An alternative to standard node requests (such as the request on root below) is to use blob.

```
->W,    4 B: /~~~
W->,  116 B:
/~~~,ssssssssssssssss~~~$stat~~~cfg~$syscfg~io~~ch~~aux~bus~main~~~~mtx~dca~mgrp~~~~fx~~car
ds~~~play~~~~rec~$ctl~~~~
```

Blob types typically apply on WING nodes in order to retrieve the internal binary equivalent of the JSON tree level respective of a WING node.
Shown below is a request at root level using the native commands part of the blob data [all bytes sent shown as hex data]

```
/ ,b dd
```

Data actually sent (in hex): `->W,    16 B: 2f0000002c62000000000002dd000000`
WING's reply is:

```
W->,  440 B: /~~~,b~~425 bytes:
df00180000000097a00439000005247374617405535441544500 00df001100000000edca7af9000003636667000
000df001500000000f89818a60000072473797363666700000 0df00130000000294f7794000002696f03492f4f
0000df001700000000 70b10139 00000263680743484 14e4e454c0000df001c000000008fa3078d0000036175780
b415558204348414e4e454c0000df001400000000f46c185e000003627573034255530000df00160000000004d3
a3a80000046d61696e044d41494e0000df001700000000f82a5af20000036d7478064d41545249580000df00140
0000000e313aeff000003646361034443410000df001c00000000d252398b0000046d6772700a4d555445204752
4f55500000df001700000000473c913400000266780745464645435453000 0df002200000000b4296fc90000056
3617264730f455850414e53494f4e2043415244530000df00180000000057297a28000004706c617906504c4159
45520000df001900000000fab1762c000003726563085245434f524445520000df001900000000cbb9514300000
42463746c07434f4e54524f4c0000de
```

Lots of information are returned either as string, or more often as blob. In the reply above, after each '`df`' byte is a data length on two bytes, immediately followed by the binary address (the hash) where a node, parameter, or subtree data can be found. For example, the subtree entry for channel (`/ch`) can be found at address/hash `70b10139`

An example on retrieving the DAW node (hash is `df17c242`, part of the `$ctl` subtree) is shown below.
Sending the OSC blob:

```
/$ctl/daw ,b dd
```

or

```
/ ,b d7df17c242dd
```

Respectively translate in the following binary data being sent to the console:

```
->W,  24 B: 2f2463746c2f6461770000002c62000000000002dd000000
or
->W,  20 B: 2f0000002c62000000000007d7df17c242dd0000
```

To which the console replies with (it can also reply with one of the errors listed earlier in the OSC chapters):

```
W->,  876 B: /$ctl/daw~~~,b~~856 bytes:
df0022df17c2423cb129d50000026f6e0a44415720454e41424c450040000000000000000001df0028df17c2424e5
c7f340000004636f6e6e0a434f4e4e454354494f4e005000020344494e000355534200df0027df17c242e5681680
000004656d756c09454d554c4154494f4e00500002034d435500034855490 0df0071df17c24242701ca90000066
36f6e6669670000500004024343414435553544f4d20434f4e54524f4c53204f4e4c59044d5354520a53494e474c
45204d4355084d535452314558540e4d4355202b20455854454e444552084d53545232455854114d4355202b203
```

[7] Detail information on native commands is provided in a separate chapter

```
27820455854454e444552df002edf17c242ae1538a4000004636375701455534520555050455220434320464f52
2044415700400000000000000001df0035df17c2429fa4e732000006646973 6a6f671944495341424c452057484
5454c20445552494e4720504c41590040000000000000001df0097df17c242892e512d00000670726573657412
4c415354204c4f414445442050524553455400500008012d012d0663756261736506435542415345046c6976650
44c495645066c6f67696378074c4f4749432058066e75656e646f064e55454e444f0870726f746f6f6c73095052
4f20544f4f4c53067265617065720652454150455209737475646 96f6f6e650a53545544494f204f4e45df001fd
f17c242beefaeab000003246f6e06444157204f4e024000000000000000001df0027df17c2429631559f00000624
62706167650b425554544f4e2050414745004000000000000000004df0031df17c242012dc5460000092462746e7
46f7563681242544e53454c20464144455220544f554348004000000000000000001df002adf17c242775c19c200
00082462746e76706f740c42544e53454c20562d504f5400400000000000000001df002ddf17c24242aeb928000
00a2462746e7265637264790d42544e53454c2052454352445900400000000000000001df0029df17c242fccfbe
070000082462746e6175746f0b42544e53454c204155544f004000000000000000001df002adf17c24285cdce3f0
000082462746e7673656c0c42544e53454c20562d53454c004000000000000000001df002ddf17c24215abd96800
000a2462746e696e736572740d42544e53454c20494e534552540040000000000000001de
```

The above is more difficult to read than the more standard way of retrieving the node, but contains more information:

```
->W,  12 B: /$ctl/daw~~~
W->, 156 B:
/$ctl/daw~~~,sssssssssssss~on~~conn~~~~emul~~~~config~~ccup~~~~preset~~$on~$bpage~~$btntou
ch~~~$btnvpot~~~~$btnrecrdy~~$btnauto~~~~$btnvsel~~~~$btninsert~~
```

Matching the two representations tell us that:
`daw/on` is at binary address `3cb129d5`,
`daw/conn` at `4e5c7f34`,
`daw/emul` at `e5681680`,
`daw/config` at `42701ca9`,
`daw/ccup` at `ae1538a4`,
`daw/preset` at `892e512d`,
`daw/$on` at `beefaeab`,
and so on (highlighted values above).

The blob Type Tag can also be used to execute native/binary commands. Using for example the `daw/$on` hash/binary address value of `beefaeab,` we can set the console in and out of DAW mode, as if one would have pressed the DAW button.
For example, sending any of the following commands will set DAW mode ON:

```
/ ,b d7beefaeab01
->W,   20 B: /~~~,b~~6 bytes: d7beefaeab01
W->,   12 B: /*~~,s~~OK~~
/$ctl/daw/$on ,b 01
->W,   28 B: /$ctl/daw/$on~~~,b~~1 bytes: 01~~~
W->,   12 B: /*~~,s~~OK~~
```

In the binary data sent with the line above, the segment `01` is equivalent to asking the value of the parameter to be set using a 32bit integer with value 1.

The following lines are requesting to turn OFF DAW mode:

```
/ ,b d7beefaeab00
->W,   20 B: /~~~,b~~6 bytes: d7beefaeab00
W->,   12 B: /*~~,s~~OK~~
/$ctl/daw/$on ,b 00
->W,   28 B: /$ctl/daw/$on~~~,b~~1 bytes: 00~~~
W->,   12 B: /*~~,s~~OK~~
```

In both blob Type Tag commands above, the console replies with a blob. Depending on the cases, it can also return strings.

AS seen above, the Tag Type blob can be used to retrieve the description of WING parameters when using the native command 'data description' a.k.a. '**dd**'; In an example below, still using the DAW ON state, we can get the data using the following command:

```
/$ctl/daw/$on ,b dd
->W,   28 B: /$ctl/daw/$on~~~,b~~1 bytes: dd~~~
```

WING returns the following which includes the hash value for `/$ctl/daw/$on` and its full description:

```
W->,   60 B: /$ctl/daw/$on~~~,b~~35 bytes:
df001fdf17c242beefaeab000003246f6e06444157204f4e0040000000000000001de
parse 35 bytes node
    len: 31, parent: df17c242, hash: beefaeab, index: 0, flags: 0040
    name: $on longname: DAW ON, type: <int> [0..1]

End node
```

The blob Tag Type can be used to retrieve the value of WING parameters when using the native command 'data request', a.k.a. '**dc**'; In an example below, still using the DAW ON state, we can get the data using the following command:

```
->W,   20 B: /~~~,b~~6 bytes: d7beefaeabdc
W->,   20 B: /~~~,b~~7 bytes: d7beefaeab01de

With 01 indicating the DAW [Remote control] button is in an ON state.
```

Detailed information on the native/binary interface to WING and data value coding is provided in a separate document.

Subscribing to OSC Data

There are three main types of subscription covering binary or OSC messages.
At the time of this document, a maximum of 1 subscription can be active at any time, provided to the last requestor. Subscriptions must be renewed every `10 seconds` in order to keep alive by sending one of the 3 messages shown below.

`/*b~` (or `/*b~,~~~`) will enable receiving event driven binary messages
Binary messages are formatted exactly as the binary/native interface and therefore can be sent back to the console with no change.

Example using mutes and faders

```
->W,    4 B: /*b~
W->,   32 B: /~~~,b~~20 bytes: d738ae75c2d5c3100000d77e463474d5c3100000
W->,   24 B: /~~~,b~~12 bytes: d7f50f69f801d726855cd301
```

`/*s~` (or `/*s~,~~~`) will enable receiving event OSC messages
OSC messages are received as triplets of data, as previously presented[8], and shown below; Sending back data to WING will require to select one of the (up to) 3 parameters received, depending on the chosen format. The 'string' argument will always work for all messages.

Example using mutes and faders

```
->W,    4 B: /*s~
W->,   32 B: /ch/1/fdr~~~,sff~~~~-oo~[0.0000][-144.0000]
W->,   32 B: /ch/1/$fdr~~,sff~~~~-oo~[0.0000][-144.0000]
W->,   32 B: /ch/1/mute~~,sfi~~~~1~~~[1.0000][     1]
W->,   32 B: /ch/1/$mute~,sfi~~~~1~~~[0.5000][     1]
```

`/*S~` (or `/*S~,~~~`) will enable receiving event OSC messages
OSC messages are received as single tag data, as shown below; WING reports the native format of the OSC pattern (ex: '`f`' for floats, '`i`' for integers, etc.). Data received with events resulting of a `/*S~` subscription can be sent back to the console with no change.

Example using mutes and faders

```
->W,    4 B: /*S~
W->,   20 B: /ch/1/fdr~~~,f~~[-144.0000]
W->,   20 B: /ch/1/$fdr~~,f~~[-144.0000]
W->,   20 B: /ch/1/mute~~,i~~[     1]
W->,   20 B: /ch/1/$mute~,i~~[     1]
```

Using the simple forms of subscription requests will provide data from the console to the requesting IP/port. It is possible to redirect the data received from WING by prefixing the commands with a port specifier element as shown below:

`/%23456/*b~` will subscribe to binary messages, being sent by WING to port `23456`.
`/%23456/*s~` will subscribe to OSC messages, being sent by WING to port `23456`.
`/%23456/*S~` will subscribe to OSC messages, being sent by WING to port `23456`.

[8] Refer to "Writing (Set) Parameter and Node data", paragraph "Single Parameters"

Effects and Plugins

WING comes with an impressive number of effects, plugins and emulations that can be used on any channel without costing any FX slots. In every channel, Gate, or EQ Compressor can take different processing models you can organize and change on the fly. The following pages below present the different effects and their parameters.

Plugins

Plugins entries are directly included with channels, busses, etc. and can either default to WING standard algorithms or adapt to alternative plugins to color your sound or fit your taste when it comes to mixing. Plugins are showing under the main JSON structure, only when instantiated. WING Channel audio engines enable 4 sorts of plugins: Filter, Gate, EQ and Dynamics. Bus, Main and Matrix audio engines support EQ and Dynamics plugins.

The choice of plugin is represented by the name (or model) of the plugin, as set under the respective "`mdl`" token; After a console reset, the default channel Filter, Gate, EQ and Dynamics plugins will be "`TILT`", "`GATE`", "`STD`", and "`COMP`", respectively, and these can be changed to one of the multiple plugins available within the console (respecting the category they apply to of course).

The choice of plugin is represented by the name (or model) of the plugin, as set under the respective "`mdl`" token; authorized values are:

Filters:

```
TILT EQ, MAXER, AP 90, AP 180
```

Gates:

```
GATE/EXPANDER, DUCKER, EVEN 88 GATE, SOUL 9000 GATE, DRAW MORE 241, BDX902 DEESSER,
WAVE DESIGNER, DYNAMIC EQ, SOUL WARMTH PRE, 76 LIMITER AMP, LA LEVELER, AUTO RIDER,
SOURCE EXTRACTOR
```

Equalizers:

```
WING EQ, SOUL ANALOGUE, EVEN 88 FORMANT, EVEN 84, FORTISSIMO 110, PULSAR, MACH EQ4
```

Compressors:

```
WING COMPRESSOR, WING EXPANDER, BDX 160 COMP, BDX 560 EASY, DRAW MORE COMP, EVEN
COMP/LIM, SOUL 9000, SOUL BUS COMP, RED3 COMPRESSOR, 76 LIMITER AMP, LA LEVELER,
FAIR KID, ETERNAL BLISS, NO-STRESSOR, WAVE DESIGNER, AUTO RIDER, PIA2250 RACK,
LTA100 LEVELER
```

Starting with FW 1.12, plugins can also be presented as a set of effects grouped together under one name and providing a series of plugins dedicated to channel strip sound shaping; They can nevertheless use as standard effects too.

Channel

```
NONE, EXTERNAL, SOUL ANALOGUE, EVEN 88 FORMANT, EVEN 84, FORTISSIMO 110, PULSAR,
MACH EQ4, EVEN CHANNEL, SOUL CHANNEL, VINTAGE CHANNEL, BUS CHANNEL, MASTERING
```

Effects

Effects nodes are part of the main JSON structure, under the `fx.n` names, with `n: [1…16]` representing the 16 effects slots available for simultaneous use in the WIN audio processing. These 16 slots are divided in two sets of slots: 1-8 accepting premium, standard or channel effects, and slots 9-16 accepting standard and channel effects, respectively.

As in the case of plugins, the choice of effect is represented by the name (or model) of the effect, as set under the respective "`mdl`" token; authorized values are:

Premium

```
NONE, EXTERNAL, HALL REVERB, ROOM REVERB, CHAMBER REVERB, PLATE REVERB, CONCERT
REVERB, AMBIENCE, VSS3 REVERB, VINTAGE ROOM, VINTAGE REVERB, VINTAGE PLATE, GATED
REVERB, REVERSE REVERB, ELAY/REVERB, SHIMMER REVERB, SPRING REVERB, DIMENSION CRS,
STEREO CHORUS, STEREO FLANGER, STEREO DELAY, ULTRATAP DELAY, TAPE DELAY, OILCAN
DELAYB, BD DELAY, STEREO PITCH, DUAL PITCH
```

Standard

```
NONE, EXTERNAL, GRAPHIC EQ, PIA 560 GEQ, C5-COMBINATOR, DOUBLE VOCAL, PRECISION
LIMITER, 2-BAND DEESSER, ULTRA ENHANCER, EXCITER, PSYCHO BASS, ROTARY SPEAKER,
PHASER, TREMOLO/PANNER, TAPE MACHINE, MOOD FILTER, BODYREZ, SUB OCTAVER, SUB
MONSTER, PICH FIX, RACK AMP, UK ROCK AMP, ANGEL AMP, JAZZ CLEAN AMP, DELUXE AMP,
SOUL ANALOGUE, EVEN 88 FORMANT, EVEN 84, FORTISSIMO 110, PULSAR, MACH EQ4, VELVET
IMAGER, SPEAKER MANAGER, TRIPLE DEQ
```

Channel

```
NONE, EXTERNAL, SOUL ANALOGUE, EVEN 88 FORMANT, EVEN 84, FORTISSIMO 110, PULSAR,
MACH EQ4, EVEN CHANNEL, SOUL CHANNEL, VINTAGE CHANNEL, BUS CHANNEL, MASTERING
```

Effects can be used as dedicated inserts at defined locations within the audio path.
If an effect is part of a channel insert, assigning the effect to a different channel will remove the effect from its previous channel assignment. In order to create a more traditional effect bus, WING requires to dedicate one of the channels to the operation; Channels that want to use the effect bus can the send their audio (or a part of it) to the channel that carries the effect, creating an effect mix bus that will apply the same effect to several sources mixed into the effect channel and provide the resulting effect as a traditional effect return that can be routed to a bus.

As for the case of plugins, Effect types/engines are represented by their respective model name under the "`mdl`" tag, enabling the selection (loading) of a specific in one of the 16 available effect slots.

The JSON tree dedicated to effects has the following structure:

```
"fx": {
        "1": {
                "mdl": "NONE",
                "fxmix": 100
        }
        "2"…"16": {}
}
```

In fact, there are a few more, read-only[9] elements in the actual WING structure of a non-affected effect slot, resulting in the following JSON structure:

```
"fx": {
        "1": {
                "mdl": "NONE",
                "fxmix": 100,
                "$esrc": 0,     external source: [0…400]
```

[9] Read-only JSON elements start with a '$' character

```
                "$emode": M,     external mode: Mono, Stereo, Mid/Side
                "$a_chn": 0,     assign channel: [0…76]
                "$a_pos": 0      assign position: 0, 1]
        }
        "2"…"16": {}
}
```

Once an effect is assigned to a slot, the JSON structure for the respective slot is extended to include the parameters for the assigned effect. For example, installing reverb effect "ROOM" in effect slot 5 will result in the following update to the JSON of effect 5:

```
"fx": {
        …
        "5": {
                "mdl": "ROOM",
                "fxmix": 100,
                "$esrc": 0,      [0…400]
                "$emode": M,     [M, ST, M/S]
                "$a_chn": 0,     [0, 1]
                "$a_pos": 0,     [0, 1]
                "pdel":          pre-delay
                "size":          room size
                "dcy":           decay
                "mult":          bass multiplier
                "damp":          damping
                "lc":            low cut
                "hc":            high cut
                "shp":           shape
                "sprd":          spread
                "diff":          diffusion
                "spin":          spin
                "ecl":           echo left
                "ecr":           echo right
                "efl":           feed left
                "efr":           feed right
        }
        …
}
```

Each available effect is a sort of program including a set of dedicated parameters. When choosing a specific effect, the effect program is instantiated in one of the available slots and its parameters are mapped to the main Jason parameters lists for that effect slot, thus enabling for example up to 16 different copies[10] of the same effect to be active on every effect slot, with differentiated parameters for each slot.

The tables in "Appendix: Effects and Plugins' Parameters list, provide all effect names and parameters, and the parameter types associated with each known effect.

[10] For standard effects, 8 for premium effects

WING ae_data OSC commands list

The next chapters provide an abridged[11] list of all OSC commands available for WING.

All commands and parameters below are part of the ***ae_data*** section in JSON snapshot files. Other console control commands part of the ***ce_data*** section in JSON snapshot files are described later in this document.

Status

Command	Type	Range	Text	Description
/$stat	N			Status node
/$stat/modtype	S		NONE, DANTE, WSG	Internal Module type [RO]
/$stat/A	N			AES50 A node
/$stat/A/stat	S		-, OK, ERR, UPD	AES50 A state [RO]
/$stat/A/dev	S		128 chars max	AES50 A Device [RO]
/$stat/B	N			AES50 B node
/$stat/B/stat	S		-, OK, ERR, UPD	AES50 B state [RO]
/$stat/B/dev	S		128 chars max	AES50 B Device [RO]
/$stat/C	N			AES50 C node
/$stat/C/stat	S		-, OK, ERR, UPD	AES50 C state [RO]
/$stat/C/dev	S		128 chars max	AES50 C Device [RO]
/$stat/lock	I	0..1		Clock lock [RO]
/$stat/ppm	I	-200..200		Clock ppm [RO]
/$stat/solo	I	0..1		Solo [RO]
/$stat/sip	I	0..1		Solo In Place [RO]
/$stat/rtcerr	I	0..1		Real Time Clock Error [RO]
/$stat/time	S		12 chars max	Clock time (depending on time format) [RO]
/$stat/date	S		12 chars max	Clock date (depending on date format) [RO]
/$stat/usbstate	S		-, ERR, IDLE, BUSY	USB Player state [RO]
/$stat/usbvolname	S		20 chars max	USB Player volume name [RO]
/$stat/sc_stat	S		OK, ERR	StageConnect status [RO]
/$stat/sc_devices	S		128 chars max	StageConnect devices [RO]
/$stat/sc_upcnt	I	0..32		StageConnect upstreams [RO]
/$stat/sc_dncnt	I	0..32		StageConnect downstreams [RO]
/$stat/sc_uprout	S		32 char max	StageConnect upstream routing [RO]
/$stat/rmt_a	S		16 chars max	Name of the console connected on AES50 port A [RO]
/$stat/rmt_b	S		16 chars max	Name of the console connected on AES50 port A [RO]
/$stat/rmt_c	S		16 chars max	Name of the console connected on AES50 port A [RO]

[11] It includes the set of commands for the first element of a series. For example, `/ch/1` set of OSC commands are listed, but not `/ch/2` to `/ch/40`.

General Configuration

Command	Type	Range	Text	Description
/cfg	N			General Configuration node
/cfg/mainlink	S		OFF, 2, 2-3, 2-4	Main Link
/cfg/dcamgrp	I	0..1		DCA mutegroups (DCA mute mutes all channels assigned to DCA)
/cfg/mon	N			Monitor buses config node
/cfg/mon/1	N	1..2		Monitor bus 1 node
/cfg/mon/1/$lvl	F	-144..10	-oo..10 in 1024 steps	Monitor bus 1 level (dB)[12]
/cfg/mon/1/inv	I	0..1		Monitor bus 1 invert (polarity)
/cfg/mon/1/pan	F	-100..100	201 steps	Monitor bus 1 pan
/cfg/mon/1/wid	F	-150..150	61 steps	Monitor bus 1 width (%)
/cfg/mon/1/eq	N			Monitor bus 1 EQ node
/cfg/mon/1/eq/on	I	0..1		Monitor bus 1 EQ off/on
/cfg/mon/1/eq/lsg	F	-15..15	301 steps	Monitor bus 1 EQ low shelf gain (dB)
/cfg/mon/1/eq/lsf	F	20..2000	641 steps	Monitor bus 1 EQ low shelf frequency (Hz)
/cfg/mon/1/eq/1g	F	-15..15	301 steps	Monitor bus 1 EQ band 1 gain (dB)
/cfg/mon/1/eq/1f	F	20..20000	961 steps	Monitor bus 1 EQ band 1 frequency (Hz)
/cfg/mon/1/eq/1q	F	0.44..10	181 steps	Monitor bus 1 EQ band 1 Q
/cfg/mon/1/eq/2g	F	-15..15	301 steps	Monitor bus 1 EQ band 2 gain (dB)
/cfg/mon/1/eq/2f	F	20..20000	961 steps	Monitor bus 1 EQ band 2 frequency (Hz)
/cfg/mon/1/eq/2q	F	0.44..10	181 steps	Monitor bus 1 EQ band 2 Q
/cfg/mon/1/eq/3g	F	-15..15	301 steps	Monitor bus 1 EQ band 3 gain (dB)
/cfg/mon/1/eq/3f	F	20..20000	961 steps	Monitor bus 1 EQ band 3 frequency (Hz)
/cfg/mon/1/eq/3q	F	0.44..10	181 steps	Monitor bus 1 EQ band 3 Q
/cfg/mon/1/eq/4g	F	-15..15	301 steps	Monitor bus 1 EQ band 4 gain (dB)
/cfg/mon/1/eq/4f	F	20..20000	961 steps	Monitor bus 1 EQ band 4 frequency (Hz)
/cfg/mon/1/eq/4q	F	0.44..10	181 steps	Monitor bus 1 EQ band 4 Q
/cfg/mon/1/eq/5g	F	-15..15	301 steps	Monitor bus 1 EQ band 5 gain (dB)
/cfg/mon/1/eq/5f	F	20..20000	961 steps	Monitor bus 1 EQ band 5 frequency (Hz)
/cfg/mon/1/eq/5q	F	0.44..10	181 steps	Monitor bus 1 EQ band 5 Q
/cfg/mon/1/eq/6g	F	-15..15	301 steps	Monitor bus 1 EQ band 6 gain (dB)
/cfg/mon/1/eq/6f	F	20..20000	961 steps	Monitor bus 1 EQ band 6 frequency (Hz)
/cfg/mon/1/eq/6q	F	0.44..10	181 steps	Monitor bus 1 EQ band 6 Q
/cfg/mon/1/eq/hsg	F	-15..15	301 steps	Monitor bus 1 EQ high shelf gain (dB)
/cfg/mon/1/eq/hsf	F	50..20000	833 steps	Monitor bus 1 EQ high shelf frequency (Hz)
/cfg/mon/1/lim	F	-40..0	41 steps	Monitor bus 1 limiter level(dB)
/cfg/mon/1/dly	N			Monitor bus 1 delay node
/cfg/mon/1/dly/on	I	0..1		Monitor bus 1 delay off/on

[12] This command is considered RO on the full-size WING, and can be set for other devices where the actual surface control potentiometer is not present.

/cfg/mon/1/dly/m	F	0.1..100	1000 steps	Monitor bus 1 delay (meters)
/cfg/mon/1/dim	F	40..0	41 steps	Monitor bus 1 delay dim level (dB)
/cfg/mon/1/pfldim	F	40..0	41 steps	Monitor bus 1 PFL Dim (dB)
/cfg/mon/1/eqbdtrim	F	0..24	25 steps	Monitor bus 1 band solo trim {dB)
/cfg/mon/1/srclvl	F	-144..10	-oo..10 in 1024 steps	Monitor bus 1 source level
/cfg/mon/1/srcmix	F	-144..10	-oo..10 in 1024 steps	Monitor bus 1 source mix (dB)
/cfg/mon/1/src	S		OFF, MAIN.1..MAIN.4, MTX.1..MTX.8, BUS.1..BUS.16, AUX.1..AUX.8	Monitor bus 1 source
/cfg/mon/1/$lvlact	F	-144..10	-oo..10 in 1024 steps	Monitor bus 1 fader level [RO]
/cfg/mon/1/tags	S		Up to 80 chars	Monitor bus 1 tags
/cfg/solo	N			Solo config node
/cfg/solo/mode	S		LIVE, STUDIO, SIP	Solo mode
/cfg/solo/mon	S		A, B, A+B	Solo monitor
/cfg/solo/mute	I	0..1		Solo mute
/cfg/solo/$dim	I	0..1		Solo dim off/on
/cfg/solo/$mono	I	0..1		Solo mono off/on
/cfg/solo/$flip	I	0..1		Solo flip
/cfg/solo/chtap	S		PFL, AFL	Solo channel tap
/cfg/solo/bustap	S		PFL, AFL	Solo bus tap
/cfg/solo/maintap	S		PFL, AFL	Solo main tap
/cfg/solo/mtxtap	S		PFL, AFL	Solo matrix tap
/cfg/solo/srcsolo	S		OFF, CH39, AUX7	Source Solo Enable
/cfg/solo/$srcsolo	I	0..1		Source Solo
/cfg/solo/$srcsgrp	I	1..13		Source Solo Group
/cfg/solo/$srcsin	I	1..64		Source Solo In
/cfg/rta	N			RTA config node
/cfg/rta/$src	I	1..76		RTA source [RO]
/cfg/rta/$tap	S		IN, POST, FILT, PREEQ, POSTEQ, PREFDR, GATEK, DYNK, DYNXO, PRETAP, SOLO, MON.A, MON.B, FXIN, FXOUT	RTA source tap [RO]
/cfg/rta/$dec	S		SLOW, MED, FAST	RTA Decay [RO]
/cfg/rta/$det	S		PEAK, RMS	RTA Detector [RO]
/cfg/rta/rtasrc	I	0..76		RTA source (indexed)
/cfg/rta/rtatap	S		IN, POST, FILT, PREEQ, POSTEQ, PREFDR, GATEK, DYNK, DYNXO, PRETAP, SOLO, MON.A, MON.B	RTA source tap
/cfg/rta/rtadecay	S		SLOW, MED, FAST	RTA decay
/cfg/rta/rtadet	S		PEAK, RMS, AVG	RTA detector
/cfg/rta/rtarange	F	30, 60		RTA range (dB)
/cfg/rta/rtagain	F	-5..50	56 steps	RTA gain (dB)
/cfg/rta/rtaauto	I	0..1		RTA autogain
/cfg/rta/eqdecay	S		SLOW, MED, FAST	RTA eq decay
/cfg/rta/reqdet	S		PEAK, RMS, AVG	RTA eq detector
/cfg/rta/eqrange	F	30, 60		RTA eq range (dB)
/cfg/rta/eqgain	F	-5..50	56 steps	RTA eq gain (dB)
/cfg/rta/eqauto	I	0..1		RTA eq autogain
/cfg/mtr	N			Meter config node
/cfg/mtr/$scopesrc	I	1..76		Meter scope source [RO]
/cfg/mtr/$scopetap	S		IN, POST, FILT, PREEQ, POSTEQ, PREFDR, GATEK, DYNK, DYNXO,	Meter scope source tap point [RO]

			PRETAP, SOLO, MON.A, MON.B, FXIN, FXOUT	
/cfg/mtr/scopesrc	I	0..76		Meter scope source
/cfg/mtr/scopetap	S		IN, POST, FILT, PREEQ, POSTEQ, PREFDR, GATEK, DYNK, DYNXO, PRETAP, SOLO, MON.A, MON.B	Meter scope source tap point
/cfg/mtr/mtrsfc	N			Meters fader node [Setup→Surface]
/ cfg/mtr/mtrsfc/in	S		PRE, POST	Meters fader section channel tap
/ cfg/mtr/mtrsfc/bus	S		PRE, POST	Meters fader section bus tap
/ cfg/mtr/mtrsfc/main	S		PRE, POST	Meters fader section main tap
/ cfg/mtr/mtrsfc/mtx	S		PRE, POST	Meters fader section matrix tap
/ cfg/mtr/mtrsfc/dca	S		PRE, POST	Meters fader section DCA tap
/cfg/mtr/mtrpage	N			Meters page node (Meters screen)
/ cfg/mtr/mtrpage/in	S		PRE, POST	Meters page channels tap
/ cfg/mtr/mtrpage/bus	S		PRE, POST	Meters page bus tap
/ cfg/mtr/mtrpage/main	S		PRE, POST	Meters page mains tap
/ cfg/mtr/mtrpage/mtx	S		PRE, POST	Meters page matrix tap
/ cfg/mtr/mtrpage/dca	S		PRE, POST	Meters page DCA tap
/ cfg/mtr/mainmtr	S		MAIN.1.. MAIN.4, MTX.1.. MTX.8, SEL_CH	Main meter
/ cfg/mtr/mainpos	S		AUTO, PRE, POST	Main position
/cfg/talk	N			Talkback config node
/cfg/talk/assign	S		OFF, CH40, AUX8	Talkback assignments
/cfg/talk/$lvl	F	-144..10	-oo..10 in 1024 steps	Talkback level (dB) [RO]
/cfg/talk/indiv	I	0..1		Use individual Bus/Main TB send levels
/cfg/talk/A	N			Talkback A node
/cfg/talk/A/$on	I	0..1		Talkback A off/on
/cfg/talk/A/mode	S		AUTO, PUSH, LATCH	Talkback A mode
/cfg/talk/A/mondim	I	0..1		Talkback A monitor dim
/cfg/talk/A/busdim	F	0..40	41 steps	Talkback A bus dim
/cfg/talk/A/B1	I	0..1		Talkback A bus 1 assign
/cfg/talk/A/B2	I	0..1		Talkback A bus 2 assign
/cfg/talk/A/B3	I	0..1		Talkback A bus 3 assign
/cfg/talk/A/B4	I	0..1		Talkback A bus 4 assign
/cfg/talk/A/B5	I	0..1		Talkback A bus 5 assign
/cfg/talk/A/B6	I	0..1		Talkback A bus 6 assign
/cfg/talk/A/B7	I	0..1		Talkback A bus 7 assign
/cfg/talk/A/B8	I	0..1		Talkback A bus 8 assign
/cfg/talk/A/B9	I	0..1		Talkback A bus 9 assign
/cfg/talk/A/B10	I	0..1		Talkback A bus 10 assign
/cfg/talk/A/B11	I	0..1		Talkback A bus 11 assign
/cfg/talk/A/B12	I	0..1		Talkback A bus 12 assign
/cfg/talk/A/B13	I	0..1		Talkback A bus 13 assign
/cfg/talk/A/B14	I	0..1		Talkback A bus 14 assign
/cfg/talk/A/B15	I	0..1		Talkback A bus 15 assign
/cfg/talk/A/B16	I	0..1		Talkback A bus 16 assign
/cfg/talk/A/M1	I	0..1		Talkback A main 1 assign
/cfg/talk/A/M2	I	0..1		Talkback A main 2 assign
/cfg/talk/A/M3	I	0..1		Talkback A main 3 assign

/cfg/talk/A/M4	I	0..1		Talkback A main 4 assign
/cfg/talk/B	N			Talkback B node
/cfg/talk/B/$on	I	0..1		Talkback B off/on
/cfg/talk/B/mode	S		AUTO, PUSH, LATCH	Talkback B mode
/cfg/talk/B/mondim	I	0..1		Talkback B monitor dim
/cfg/talk/B/busdim	F	0..40	41 steps	Talkback B bus dim
/cfg/talk/B/B1	I	0..1		Talkback B bus 1 assign
/cfg/talk/B/B2	I	0..1		Talkback B bus 2 assign
/cfg/talk/B/B3	I	0..1		Talkback B bus 3 assign
/cfg/talk/B/B4	I	0..1		Talkback B bus 4 assign
/cfg/talk/B/B5	I	0..1		Talkback B bus 5 assign
/cfg/talk/B/B6	I	0..1		Talkback B bus 6 assign
/cfg/talk/B/B7	I	0..1		Talkback B bus 7 assign
/cfg/talk/B/B8	I	0..1		Talkback B bus 8 assign
/cfg/talk/B/B9	I	0..1		Talkback B bus 9 assign
/cfg/talk/B/B10	I	0..1		Talkback B bus 10 assign
/cfg/talk/B/B11	I	0..1		Talkback B bus 11 assign
/cfg/talk/B/B12	I	0..1		Talkback B bus 12 assign
/cfg/talk/B/B13	I	0..1		Talkback B bus 13 assign
/cfg/talk/B/B14	I	0..1		Talkback B bus 14 assign
/cfg/talk/B/B15	I	0..1		Talkback B bus 15 assign
/cfg/talk/B/B16	I	0..1		Talkback B bus 16 assign
/cfg/talk/B/M1	I	0..1		Talkback B main 1 assign
/cfg/talk/B/M2	I	0..1		Talkback B main 2 assign
/cfg/talk/B/M3	I	0..1		Talkback B main 3 assign
/cfg/talk/B/M4	I	0..1		Talkback B main 4 assign
/cfg/amix	N			Automixing node
/cfg/amix/x	I	0..1		Automix X group enable
/cfg/amix/y	I	0..1		Automix Y group enable

System Settings

Command	Type	Range	Text	Description
/$syscfg	N			System configuration node
/$syscfg/consolename	S		16 chars max	Console name
/$syscfg/logflags	S		256 char max	Log flags
/$syscfg/ipmode	S		DHCP, STATIC	IP Mode
/$syscfg/ip0	I	0..255		IP first number
/$syscfg/ip1	I	0..255		IP second number
/$syscfg/ip2	I	0..255		IP third number
/$syscfg/ip3	I	0..255		IP fourth number
/$syscfg/msk0	I	0..255		IP mask first number
/$syscfg/msk1	I	0..255		IP mask second number
/$syscfg/msk2	I	0..255		IP mask third number
/$syscfg/msk3	I	0..255		IP mask fourth numbe
/$syscfg/gw0	I	0..255		IP gateway first number
/$syscfg/gw1	I	0..255		IP gateway second number
/$syscfg/gw2	I	0..255		IP gateway third number
/$syscfg/gw3	I	0..255		IP gateway fourth number
/$syscfg/$ipapply	I	0..1		IP applied
/$syscfg/$firmware	S		64 chars max	Firmware version number [RO]
/$syscfg/$serial	S		64 chars max	Serial number [RO]
/$syscfg /tcplock	I	0..1		Prevent modifications from TCP input [RO]
/$syscfg/usbh_spd	S		FS, HS	USB driver speed setting Full Speed, High Speed[13]
/$syscfg/timefmt	S		24H, 12H	Time format
/$syscfg/datefmt	S		YMD, DMY	Date format

[13] When in FS, record is limited to 2 tracks/16bits. 4 tracks/24bits playing at once from USB stick may be affected; USB 3.1 capable memory sticks are recommended.

Input/Output Settings

Command	Type	Range	Text	Description
/io	N			Input/Output node
/io/altsw	I	0..1		Main/Alt switch
/io/in	N			Input node
/io/in/LCL	N			Local Input node
/io/in/LCL/1	N	1..8		Local Input 1 node
/io/in/LCL/1/mode	S		M, ST, M/S	Local Input 1 mode
/io/in/LCL/1/g	F	-3..45.5	98 steps	Local Input 1 gain (dB)
/io/in/LCL/1/vph	I	0..1		Local Input 1 phantom
/io/in/LCL/1/mute	I	0..1		Local Input 1 mute
/io/in/LCL/1/pol	I	0..1		Local Input 1 polarity
/io/in/LCL/1/col	I	1..12		Local Input 1 color
/io/in/LCL/1/name	S		16 chars max	Local Input 1 name
/io/in/LCL/1/icon	I	0..999		Local Input 1 icon (indexed)
/io/in/LCL/1/tags	S		80 chars max	Local Input 1 tags
/io/in/LCL/1/$ha	I	0..5		Local input 1 ha type [RO]
/io/in/LCL/1/rmt	S		OFF, AES A, AES B, AES C	Local input 1 remote control
/io/in/LCL/1/$ract	I	0..1		Local input 1 remote active [RO]
/io/in/LCL/1/$rdest	S		7 chars max	Local input 1 remote dest [RO]
/io/in/LCL/1/rcvc	I	0..1		Local input 1 remote customizations sync
/io/in/LCL/1/$mute	I	0..2		Local input 1 mute [RO]
/io/in/AUX	N			Aux Input node
/io/in/AUX/1	N	1..8		Aux Input 1 node
/io/in/AUX/1/mode	S		M, ST, M/S	Aux Input 1 mode
/io/in/AUX/1/mute	I	0..1		Aux Input 1 mute
/io/in/AUX/1/pol	I	0..1		Aux Input 1 polarity
/io/in/AUX/1/col	I	1..12		Aux Input 1 color
/io/in/AUX/1/name	S		16 chars max	Aux Input 1 name
/io/in/AUX/1/icon	I	0..999		Aux Input 1 icon (indexed)
/io/in/AUX/1/tags	S		80 chars max	Aux Input 1 tags
/io/in/AUX/1/$mute	I	0..2		Aux input 1 mute [RO]
/io/in/A	N			AES50 A Input node
/io/in/A/1	N	1..48		AES50 A Input 1 node
/io/in/A/1/mode	S		M, ST, M/S	AES50 A Input 1 mode
/io/in/A/1/g	F	-3..45.5	98 steps	AES50 A Input 1 gain (dB)
/io/in/A/1/vph	I	0..1		AES50 A Input 1 phantom power
/io/in/A/1/mute	I	0..1		AES50 A Input 1 mute
/io/in/A/1/pol	I	0..1		AES50 A Input 1 polarity
/io/in/A/1/col	I	1..12		AES50 A Input 1 color
/io/in/A/1/name	S		16 chars max	AES50 A Input 1 name
/io/in/A/1/icon	I	0..999		AES50 A Input 1 icon (indexed)
/io/in/A/1/tags	S		80 chars max	AES50 A Input 1 tags
/io/in/A/1/$ha	I	0..5		AES50 A input 1 ha type [RO]
/io/in/A/1/rmt	S		OFF, AES A, AES B, AES C	AES50 A input 1 remote control
/io/in/A/1/$ract	I	0..1		AES50 A input 1 remote active [RO]
/io/in/A/1/$rdest	S		7 chars max	AES50 A input 1 remote dest [RO]
/io/in/A/1/rcvc	I	0..1		AES50 A input 1 remote customizations sync
/io/in/A/1/$mute	I	0..2		AES50 A input 1 mute [RO]

/io/in/B	N			AES50 B Input node
/io/in/B/1	N	1..48		AES50 B Input 1 node
/io/in/B/1/mode	S		M, ST, M/S	AES50 B Input 1 mode
/io/in/B/1/g	F	-3..45.5	98 steps	AES50 B Input 1 gain (dB)
/io/in/B/1/vph	I	0..1		AES50 B Input 1 phantom power
/io/in/B/1/mute	I	0..1		AES50 B Input 1 mute
/io/in/B/1/pol	I	0..1		AES50 B Input 1 polarity
/io/in/B/1/col	I	1..12		AES50 B Input 1 color
/io/in/B/1/name	S		16 chars max	AES50 B Input 1 name
/io/in/B/1/icon	I	0..999		AES50 B Input 1 icon (indexed)
/io/in/B/1/tags	S		80 chars max	AES50 B Input 1 tags
/io/in/B/1/$ha	I	0..5		AES50 B input 1 ha type [RO]
/io/in/B/1/rmt	S		OFF, AES A, AES B, AES C	AES50 B input 1 remote control
/io/in/B/1/$ract	I	0..1		AES50 B input 1 remote active [RO]
/io/in/B/1/$rdest	S		7 chars max	AES50 B input 1 remote dest [RO]
/io/in/B/1/rcvc	I	0..1		AES50 B input 1 remote customizations sync
/io/in/B/1/$mute	I	0..2		AES50 B input 1 mute [RO]
/io/in/C	N			AES50 C Input node
/io/in/C/1	N	1..48		AES50 C Input 1 node
/io/in/C/1/mode	S		M, ST, M/S	AES50 C Input 1 mode
/io/in/C/1/g	F	-3..45.5	98 steps	AES50 C Input 1 gain (dB)
/io/in/C/1/vph	I	0..1		AES50 C Input 1 phantom power
/io/in/C/1/mute	I	0..1		AES50 C Input 1 mute
/io/in/C/1/pol	I	0..1		AES50 C Input 1 polarity
/io/in/C/1/col	I	1..12		AES50 C Input 1 color
/io/in/C/1/name	S		16 chars max	AES50 C Input 1 name
/io/in/C/1/icon	I	0..999		AES50 C Input 1 icon (indexed)
/io/in/C/1/tags	S		80 chars max	AES50 C Input 1 tags
/io/in/C/1/$ha	I	0..4		AES50 C input 1 ha type [RO]
/io/in/C/1/rmt	S		OFF, AES A, AES B, AES C	AES50 C input 1 remote control
/io/in/C/1/$ract	I	0..1		AES50 C input 1 remote active [RO]
/io/in/C/1/$rdest	S		7 chars max	AES50 C input 1 remote dest [RO]
/io/in/C/1/rcvc	I	0..1		AES50 C input 1 remote customizations sync
/io/in/C/1/$mute	I	0..2		AES50 C input 1 mute [RO]
/io/in/SC	N			StageConnect Input node
/io/in/SC/1	N	1..32		StageConnect Input 1 node
/io/in/SC/1/mode	S		M, ST, M/S	StageConnect Input 1 mode
/io/in/SC/1/mute	I	0..1		StageConnect Input 1 mute
/io/in/SC/1/pol	I	0..1		StageConnect Input 1 polarity
/io/in/SC/1/col	I	1..12		StageConnect Input 1 color
/io/in/SC/1/name	S		16 chars max	StageConnect Input 1 name
/io/in/SC/1/icon	I	0..999		StageConnect Input 1 icon (indexed)
/io/in/SC/1/tags	S		80 chars max	StageConnect Input 1 tags
/io/in/SC/1/$mute	I	0..2		StageConnect 1 mute [RO]
/io/in/USB	N			USB Input node
/io/in/USB/1	N	1..48		USB Input 1 node
/io/in/USB/1/mode	S		M, ST, M/S	USB Input 1 mode
/io/in/USB/1/mute	I	0..1		USB Input 1 mute
/io/in/USB/1/pol	I	0..1		USB Input 1 polarity

/io/in/USB/1/col	I	1..12		USB Input 1 color
/io/in/USB/1/name	S		16 chars max	USB Input 1 name
/io/in/USB/1/icon	I	0..999		USB Input 1 icon (indexed)
/io/in/USB/1/tags	S		80 chars max	USB Input 1 tags
/io/in/USB/1/$mute	I	0..2		USB Input 1 mute [RO]
/io/in/CRD	N			Card Input node
/io/in/CRD/1	N	1..64		Card Input 1 node
/io/in/CRD/1/mode	S		M, ST, M/S	Card Input 1 mode
/io/in/CRD/1/mute	I	0..1		Card Input 1 mute
/io/in/CRD/1/pol	I	0..1		Card Input 1 polarity
/io/in/CRD/1/col	I	1..12		Card Input 1 color
/io/in/CRD/1/name	S		16 chars max	Card Input 1 name
/io/in/CRD/1/icon	I	0..999		Card Input 1 icon (indexed)
/io/in/CRD/1/tags	S		80 chars max	Card Input 1 tags
/io/in/CRD/1/$mute	I	0..2		Card Input 1 mute [RO]
/io/in/MOD	N			Module Input node
/io/in/MOD/1	N	1..64		Module Input 1 node
/io/in/MOD/1/mode	S		M, ST, M/S	Module Input 1 node
/io/in/MOD/1/mute	I	0..1		Module Input 1 mute
/io/in/MOD/1/pol	I	0..1		Module Input 1 polarity
/io/in/MOD/1/col	I	1..12		Module Input 1 color
/io/in/MOD/1/name	S		16 chars max	Module Input 1 name
/io/in/MOD/1/icon	I	0..999		Module Input 1 icon (indexed)
/io/in/MOD/1/tags	S		80 chars max	Module Input 1 tags
/io/in/MOD/1/$mute	I	0..2		Module Input 1 mute [RO]
/io/in/PLAY	N			USB Player Input node
/io/in/PLAY/1	N	1..4		USB Player Input 1 node
/io/in/PLAY/1/mode	S		M, ST, M/S	USB Player Input 1 mode
/io/in/PLAY/1/mute	I	0..1		USB Player Input 1 mute
/io/in/PLAY/1/pol	I	0..1		USB Player Input 1 polarity
/io/in/PLAY/1/col	I	1..12		USB Player Input 1 color
/io/in/PLAY/1/name	S		16 chars max	USB Player Input 1 name
/io/in/PLAY/1/icon	I	0..999		USB Player Input 1 icon (indexed)
/io/in/PLAY/1/tags	S		80 chars max	USB Player Input 1 tags
/io/in/PLAY/1/$mute	I	0..2		USB Player Input 1 mute [RO]
/io/in/AES	N			AES/EBU Input node
/io/in/AES/1	N	1..2		AES/EBU Input 1 node
/io/in/AES/1/mode	S		M, ST, M/S	AES/EBU Input 1 mode
/io/in/AES/1/mute	I	0..1		AES/EBU Input 1 mute
/io/in/AES/1/pol	I	0..1		AES/EBU Input 1 polarity
/io/in/AES/1/col	I	1..12		AES/EBU Input 1 color
/io/in/AES/1/name	S		16 chars max	AES/EBU Input 1 name
/io/in/AES/1/icon	I	0..999		AES/EBU Input 1 icon (indexed)
/io/in/AES/1/tags	S		80 chars max	AES/EBU Input 1 tags
/io/in/AES/1/$mute	I	0..2		AES/EBU Input 1 mute [RO]
/io/in/USR	N			User Signal Input node
/io/in/USR/1	N	1..24		User Signal Input 1 node
/io/in/USR/1/mode	S		M, ST, M/S	User Signal Input 1 mode
/io/in/USR/1/mute	I	0..1		User Signal Input 1 mute
/io/in/USR/1/pol	I	0..1		User Signal Input 1 polarity

/io/in/USR/1/col	I	1..12		User Signal Input 1 color
/io/in/USR/1/name	S		16 chars max	User Signal Input 1 name
/io/in/USR/1/icon	I	0..999		User Signal Input 1 icon (indexed)
/io/in/USR/1/tags	S		80 chars max	User Signal Input 1 tags
/io/in/USR/1/$mute	I	0..2		User Signal Input 1 mute [RO]
/io/in/USR/1/user	N			User Signal 1 source node
/io/in/USR/1/user/grp	S		OFF, CH, AUX, BUS, MAIN, MTX	User Signal 1 source group
/io/in/USR/1/user/in	I	1..40		User Signal 1 source number
/io/in/USR/1/user/tap	S		PRE, POST	User Signal 1 source tap point
/io/in/USR/1/user/lr	S		L+R, L, R	User Signal 1 source take
/io/in/OSC	N			Oscillator Input node
/io/in/OSC/1	N	1..2		Oscillator Input 1 node
/io/in/OSC/1/mode	S		M, ST, M/S	Oscillator Input 1 mode [RO]
/io/in/OSC/1/mute	I	0..1		Oscillator Input 1 mute
/io/in/OSC/1/col	I	1..12		Oscillator Input 1 color
/io/in/OSC/1/name	S		16 chars max	Oscillator Input 1 name
/io/in/OSC/1/icon	I	0..999		Oscillator Input 1 icon (indexed)
/io/in/OSC/1/tags	S		80 chars max	Oscillator Input 1 tags
/io/in/OSC/1/$mute	I	0..2		Oscillator Input 1 mute [RO]
/io/in/OSC/1/osc	N			Oscillator 1 source node
/io/in/OSC/1/osc/lvl	F	-40..6	69 steps	Oscillator 1 source level
/io/in/OSC/1/osc/mode	S		SINE, PINK, WHITE	Oscillator 1 source mode
/io/in/OSC/1/osc/f	F	20…20000	2323 steps	Oscillator 1 source frequency
/io/in/$BUS	N			Bus Input node
/io/in/$BUS/1	N	1..32		Bus Input 1 node
/io/in/$BUS/1/mode	S		M, ST, M/S	Bus Input 1 mode [RO]
/io/in/$BUS/1/col	I	1..12		Bus Input 1 color [RO]
/io/in/$BUS/1/name	S		16 chars max	Bus Input 1 name [RO]
/io/in/$BUS/1/icon	I	0..999		Bus Input 1 icon [RO]
/io/in/$BUS/1/tags	S		80 chars max	Bus Input 1 tags [RO]
/io/in/$MAIN	N			Main Input node
/io/in/$MAIN/1	N	1..8		Main Input 1 node
/io/in/$MAIN/1/mode	S		M, ST, M/S	Main Input 1 mode [RO]
/io/in/$MAIN/1/col	I	1..12		Main Input 1 color [RO]
/io/in/$MAIN/1/name	S		16 chars max	Main Input 1 name [RO]
/io/in/$MAIN/1/icon	I	0..999		Main Input 1 icon [RO]
/io/in/$MAIN/1/tags	S		80 chars max	Main Input 1 tags [RO]
/io/in/$MTX	N			Matrix Input node
/io/in/$MTX/1	N	1..16		Matrix Input 1 node
/io/in/$MTX/1/mode	S		M, ST, M/S	Matrix Input 1 mode [RO]
/io/in/$MTX/1/col	I	1..12		Matrix Input 1 color [RO]
/io/in/$MTX/1/name	S		16 chars max	Matrix Input 1 name [RO]
/io/in/$MTX/1/icon	I	0..999		Matrix Input 1 icon [RO]
/io/in/$MTX/1/tags	S		80 chars max	Matrix Input 1 tags [RO]
/io/in/$SEND	N			FX Send Input node
/io/in/$SEND/1	N	1..32		FX Send Input 1 node
/io/in/$SEND/1/mode	S		M, ST, M/S	FX Send Input 1 mode [RO]
/io/in/$SEND/1/col	I	1..12		FX Send Input 1 color [RO]

/io/in/$SEND/1/name	S		16 chars max	FX Send Input 1 name [RO]
/io/in/$SEND/1/icon	I	0..999		FX Send Input 1 icon [RO]
/io/in/$SEND/1/tags	S		80 chars max	FX Send Input 1 tags [RO]
/io/in/$MON	N			Monitor Input node
/io/in/$MON/1	N	1..4		Monitor Input 1 node
/io/in/$MON/1/mode	S		M, ST, M/S	Monitor Input 1 mode [RO]
/io/in/$MON/1/col	I	1..12		Monitor Input 1 color [RO]
/io/in/$MON/1/name	S		16 chars max	Monitor Input 1 name [RO]
/io/in/$MON/1/icon	I	0..999		Monitor Input 1 icon [RO]
/io/in/$MON/1/tags	S		80 chars max	Monitor Input 1 tags [RO]
/io/out	N			Output node
/io/out/LCL	N			Local Output node
/io/out/LCL/1	N	1..8		Local Output 1 node
/io/out/LCL/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	Local Output 1 group
/io/out/LCL/1/in	I	1..64		Local Output 1 input
/io/out/AUX	N			Aux Output node
/io/out/AUX/1	N	1..8		Aux Output 1 node
/io/out/AUX/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	Aux Output 1 group
/io/out/AUX/1/in	I	1..64		Aux Output 1 input
/io/out/A	N			AES50 A Output node
/io/out/A/1	N	1..48		AES50 A Output 1 node
/io/out/A/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	AES50 A Output 1 group
/io/out/A/1/in	I	1..64		AES50 A Output 1 input
/io/out/B	N			AES50 B Output node
/io/out/B/1	N	1..48		AES50 B Output 1 node
/io/out/B/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	AES50 B Output 1 group
/io/out/B/1/in	I	1..64		AES50 B Output 1 input
/io/out/C	N			AES50 C Output node
/io/out/C/1	N	1..48		AES50 C Output 1 node
/io/out/C/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	AES50 C Output 1 group
/io/out/C/1/in	I	1..64		AES50 C Output 1 input
/io/out/SC	N			StageConnect Output node
/io/out/SC/1	N	1..32		StageConnect Output 1 node
/io/out/SC/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	StageConnect Output 1 group
/io/out/SC/1/in	I	1..64		StageConnect Output 1 input

/io/out/USB	N			USB Output Audio node
/io/out/USB/1	N	1..48		USB Output Audio 1 node
/io/out/USB/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	USB Output Audio 1 group
/io/out/USB/1/in	I	1..64		USB Output Audio 1 input
/io/out/CRD	N			Card Output node
/io/out/CRD/1	N	1..64		Card Output 1 node
/io/out/CRD/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	Card Output 1 group
/io/out/CRD/1/in	I	1..64		Card Output 1 input
/io/out/MOD	N			Module Output node
/io/out/MOD/1	N	1..64		Module Output 1 node
/io/out/MOD/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	Module Output 1 group
/io/out/MOD/1/in	I	1..64		Module Output 1 input
/io/out/REC	N			USB Record Output node
/io/out/REC/1	N	1..4		USB Record Output 1 node
/io/out/REC/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	USB Record Output 1 group
/io/out/REC/1/in	I	1..64		USB Record Output 1 input
/io/out/AES	N			AES/EBU Output node
/io/out/AES/1	N	1..2		AES/EBU Output 1 node
/io/out/AES/1/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX, SEND, MON	AES/EBU Output 1 group
/io/out/AES/1/in	I	1..64		AES/EBU Output 1 input

Channel Settings

Command	Type	Range	Text	Description
/ch	N			Channel node
/ch/1	N	1..40		Channel 1 node
/ch/1/in	N			Channel 1 input node
/ch/1/in/set	N			Channel 1 input set node
/ch/1/in/set/$mode	S		M, ST, M/S	Channel 1 input mode [RO]
/ch/1/in/set/srcauto	I	0..1		Channel 1 input auto source switch
/ch/1/in/set/altsrc	I	0..1		Channel 1 input main/alt switch
/ch/1/in/set/inv	I	0..1		Channel 1 input phase invert switch
/ch/1/in/set/trim	F	-18..18	361 steps	Channel 1 input trim (dB)
/ch/1/in/set/bal	F	-9..9	181 steps	Channel 1 input balance (dB)
/ch/1/in/set/$g	F	-2.5..45 -3.0..45.5	20 steps (LCL) 98 steps (AES)	Channel 1 input gain (dB) – depends on source type
/ch/1/in/set/$vph	I	0..1		Channel 1 input phantom power – depends on source type
/ch/1/in/set/dlymode	S		M, MS, SMP	Channel 1 input delay mode (meters, ms, samples)
/ch/1/in/set/dly	F	0..150	1501 steps	Channel 1 input delay (meters)
/ch/1/in/set/dlyon	I	0..1		Channel 1 input delay
/ch/1/in/conn	N			Channel 1 input connection node
/ch/1/in/conn/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX	Channel 1 main input connection group
/ch/1/in/conn/in	I	1..64		Channel 1 main input connection group index
/ch/1/in/conn/altgrp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX	Channel 1 alt input connection group
/ch/1/in/conn/altin	I	1..64		Channel 1 alt input connection group index
/ch/1/flt	N			Channel 1 filter node
/ch/1/flt/lc	I	0..1		Channel 1 low cut switch
/ch/1/flt/lcf	F	20..2000	641 steps	Channel 1 low cut frequency (Hz)
/ch/1/flt/hc	I	0..1		Channel 1 high cut switch
/ch/1/flt/hcf	F	50..20000	833 steps	Channel 1 high cut frequency (Hz)
/ch/1/flt/tf	I	0..1		Channel 1 tool filter switch
/ch/1/flt/mdl	S		TILT, MAX, AP1, AP2	Channel 1 filter model (see Appendix on Filter plugins for parameters details, OSC patterns in italic below correspond to TILT)
/ch/1/flt/tilt	F	-6..6	49 steps	Channel 1 tilt level (dB)
/ch/1/clink	I	0..1		Channel 1 custom link
/ch/1/col	I	1..12		Channel 1 color
/ch/1/name	S		16 chars max	Channel 1 name
/ch/1/icon	I	0..999		Channel 1 icon
/ch/1/led	I	0..1		Channel 1 scribble light
/ch/1/mute	I	0..1		Channel 1 mute
/ch/1/fdr	F	-144..10	-oo..10 in 1024 steps	Channel 1 fader
/ch/1/pan	F	-100..100	201 steps	Channel 1 pan
/ch/1/wid	F	-150..150	61 steps	Channel 1 width (%)

/ch/1/$solo	I	0..1		Channel 1 solo switch
/ch/1/$sololed	I	0..2		Channel 1 solo LED [RO]
/ch/1/solosafe	I	0..1		Channel 1 solo safe
/ch/1/mon	S		A, B, A+B	Channel 1 monitor mode
/ch/1/proc	S		GEDI, GEID, GIED, IGED, GDEI, GDIE, GIDE, IGDE, EGDI, EGID, EIGD, IEGD, EDGI, EDIG, EIDG, IEDG, DEGI, DEIG, DIEG, IDEG, DGEI, DGIE, DIGE, IDGE	Channel 1 process order (G: gate, E: EQ, D: Dynamics, I: Insert
/ch/1/ptap	S		IN, FILT, 3, 4, 5, PFL, AFL, POST	Channel 1 pretap (to sends)
/ch/1/$presolo	I	0..1		Channel 1 presolo
/ch/1/peq	N			Channel 1 PreSend EQ node
/ch/1/peq/on	I	0..1		Channel 1 PEQ switch
/ch/1/peq/1g	F	-15..15	301 steps	Channel 1 PEQ band 1 gain (dB)
/ch/1/peq/1f	F	20..20000	960 steps	Channel 1 PEQ band 1 frequency (Hz)
/ch/1/peq/1q	F	0.44..10	181 steps	Channel 1 PEQ band 1 Q
/ch/1/peq/2g	F	-15..15	301 steps	Channel 1 PEQ band 2 gain 9dB)
/ch/1/peq/2f	F	20..20000	960 steps	Channel 1 PEQ band 2 frequency (Hz)
/ch/1/peq/2q	F	0.44..10	181 steps	Channel 1 PEQ band 2 Q
/ch/1/peq/3g	F	-15..15	301 steps	Channel 1 PEQ band 3 gain 9dB)
/ch/1/peq/3f	F	20..20000	960 steps	Channel 1 PEQ band 3 frequency (Hz)
/ch/1/peq/3q	F	0.44..10	181 steps	Channel 1 PEQ band 3 Q
/ch/1/gate	N			Channel 1 gate node
/ch/1/gate/on	I	0..1		Channel 1 gate switch
/ch/1/gate/mdl	S		GATE, DUCK, E88, 9000G, D241, DS902, WAVE, DEQ, WARM, 76LA, LA, RIDE, PSE	Channel 1 gate model (see Appendix on Gate plugins for parameters details, OSC patterns in italic below correspond to GATE)
/ch/1/gate/thr	F	-80..0	161 steps	Channel 1 gate threshold (dB)
/ch/1/gate/range	F	3..60	115 steps	Channel 1 gate range (dB)
/ch/1/gate/att	F	0..120	121 steps	Channel 1 gate attack (ms)
/ch/1/gate/hld	F	0..200	200 steps	Channel 1 gate hold (ms)
/ch/1/gate/rel	F	4..4000	130 steps	Channel 1 gate release(ms)
/ch/1/gate/acc	F	0..100	21 steps	Channel 1 gate accent (5)
/ch/1/gate/ratio	S		1:1.5, 1:2, 1:3, 1:4, GATE	Channel 1 gate ratio
/ch/1/gatesc	N			Channel 1 gate sidechain node
/ch/1/gatesc/type	S		Off, LP12, HP12, BP	Channel 1 gate sidechain type
/ch/1/gatesc/f	F	20..20000	961 steps	Channel 1 gate sidechain frequency (Hz)
/ch/1/gatesc/q	F	0.44..10	181 steps	Channel 1 gate sidechain Q
/ch/1/gatesc/src	S		SHELF, Ch.1..Ch.40	Channel 1 gate sidechain source
/ch/1/gatesc/tap	S		IN, FILT, 3, 4, 5, PFL, AFL, POST	Channel 1 gate sidechain tap
/ch/1/gatesc/$solo	I	0..1		Channel 1 gate sidechain solo
/ch/1/eq	N			Channel 1 EQ node
/ch/1/eq/on	I	0..1		Channel 1 EQ switch
/ch/1/eq/mdl	S		STD, SOUL, E88, E84, F110, PULSAR, MACH4	Channel 1 EQ model (see Appendix on EQ plugins for parameters details, OSC patterns in italic below correspond to STD)
/ch/1/eq/mix	F	0..125	126 steps	Channel 1 EQ mix (%)

/ch/1/eq/$solo	I	0..1		Channel 1 EQ solo
/ch/1/eq/$solobd	I	0..6		Channel 1 EQ solo band
/ch/1/eq/lg	F	-15..15	301 steps	Channel 1 EQ low gain (dB)
/ch/1/eq/lf	F	20..2000	641 steps	Channel 1 EQ low frequency (Hz)
/ch/1/eq/lq	F	0.44..10	181 steps	Channel 1 EQ low Q
/ch/1/eq/leq	S		PEQ, SHV	Channel 1 EQ low type
/ch/1/eq/1g	F	-15..15	301 steps	Channel 1 EQ band 1 gain (dB)
/ch/1/eq/1f	F	20..20000	961 steps	Channel 1 EQ band 1 frequency (Hz)
/ch/1/eq/1q	F	0.44..10	181 steps	Channel 1 EQ band 1 Q
/ch/1/eq/2g	F	-15..15	301 steps	Channel 1 EQ band 2 gain (dB)
/ch/1/eq/2f	F	20..20000	961 steps	Channel 1 EQ band 2 frequency (Hz)
/ch/1/eq/2q	F	0.44..10	181 steps	Channel 1 EQ band 2 Q
/ch/1/eq/3g	F	-15..15	301 steps	Channel 1 EQ band 3 gain (dB)
/ch/1/eq/3f	F	20..20000	961 steps	Channel 1 EQ band 3 frequency (Hz)
/ch/1/eq/3q	F	0.44..10	181 steps	Channel 1 EQ band 3 Q
/ch/1/eq/4g	F	-15..15	301 steps	Channel 1 EQ band 4 gain (dB)
/ch/1/eq/4f	F	20..20000	961 steps	Channel 1 EQ band 4 frequency (Hz)
/ch/1/eq/4q	F	0.44..10	181 steps	Channel 1 EQ band 4 Q
/ch/1/eq/hg	F	-15..15	301 steps	Channel 1 EQ high gain (dB)
/ch/1/eq/hf	F	50..20000	833 steps	Channel 1 EQ high frequency (Hz)
/ch/1/eq/hq	F	0.44..10	181 steps	Channel 1 EQ high Q
/ch/1/eq/heq	S		SHV, PEQ	Channel 1 EQ high type
/ch/1/dyn	N			Channel 1 dynamic (compressor) node
/ch/1/dyn/on	I	0..1		Channel 1 compressor switch
/ch/1/dyn/mdl	S		COMP, EXP, B160, B560, D241, ECL33, 9000C, SBUS, RED3, 76LA, LA, F670, BLISS, NSTR, WAVE, RIDE, 2250, L100	Channel 1 compressor model (see Appendix on Compressor plugins for parameters details, OSC patterns in italic below correspond to COMP)
/ch/1/dyn/mix	F	0..100	101 steps	Channel 1 compressor mix (%)
/ch/1/dyn/gain	F	-6..12	37 steps	Channel 1 compressor gain (dB)
/ch/1/dyn/thr	F	-60..0	121 steps	Channel 1 compressor threshold (dB)
/ch/1/dyn/ratio	S		1.1, 1.2, 1.3, 1.5, 1.7, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 6.0, 8.0, 10, 20, 50, 100	Channel 1 compressor ratio
/ch/1/dyn/knee	I	0..5		Channel 1 compressor knee
/ch/1/dyn/det	S		PEAK, RMS	Channel 1 compressor detect
/ch/1/dyn/att	F	0..120	121 steps	Channel 1 compressor attack (ms)
/ch/1/dyn/hld	F	1..200	200 steps	Channel 1 compressor hold (ms)
/ch/1/dyn/rel	F	4..4000	130 steps	Channel 1 compressor release (ms)
/ch/1/dyn/env	S		LIN, LOG	Channel 1 compressor envelope
/ch/1/dyn/auto	I	0..1		Channel 1 compressor auto switch
/ch/1/dynxo	N			Channel 1 compressor crossover node
/ch/1/dynxo/depth	F	0..20	41 steps	Channel 1 compressor crossover depth (dB)
/ch/1/dynxo/type	S		OFF, LO6, LO12, HI6, HI12, PC	Channel 1 compressor crossover type
/ch/1/dynxo/f	F	20..20000	901 steps	Channel 1 compressor crossover frequency (Hz)
/ch/1/dynxo/$solo	I	0..1		Channel 1 compressor crossover solo
/ch/1/dynsc	N			Channel 1 compressor sidechain node
/ch/1/dynsc/type	S		Off, LP12, HP12, BP	Channel 1 compressor sidechain type

/ch/1/dynsc/f	F	20..20000	901 steps	Channel 1 compressor sidechain frequency (Hz)
/ch/1/dynsc/q	F	0.44..10	181 steps	Channel 1 compressor sidechain Q
/ch/1/dynsc/src	S		SELF, CH.1..CH.40	Channel 1 compressor sidechain source
/ch/1/dynsc/tap	S		IN, FILT, 3, 4, 5, PFL, AFL, POST	Channel 1 compressor sidechain tap
/ch/1/dynsc/$solo	I	0..1		Channel 1 compressor sidechain solo
/ch/1/preins	N			Channel 1 pre-insert node
/ch/1/preins/on	I	0..1		Channel 1 pre-insert switch
/ch/1/preins/ins	S		NONE, FX1..FX16	Channel 1 pre-insert FX slot
/ch/1/preins/$stat	S		-, OK, N/A	Channel 1 pre-insert status [RO]
/ch/1/main	N			Channel 1 Main node
/ch/1/main/1	N	1..4		Channel 1 Main 1 node
/ch/1/main/1/on	I	0..1		Channel 1 Main 1 on switch
/ch/1/main/1/lvl	F	-144..10	-oo..10 in 1024 steps	Channel 1 Main 1 fader level (dB)
/ch/1/main/pre	I	0..1		Channel 1 sent pre fader to Main 1
/ch/1/send	N			Channel 1 sends node
/ch/1/send/1	N	1..16		Channel 1 sends 1 node
/ch/1/send/1/on	I	0..1		Channel 1 sends 1 on switch
/ch/1/send/1/lvl	F	-144..10	-oo..10 in 1024 steps	Channel 1 sends 1 fader level (dB)
/ch/1/send/1/pon	I	0..1		Channel 1 sends 1 pre always on switch
/ch/1/send/1/ind	I	0..1		Channel 1 sends 1 individual tap (0=link to bus)
/ch/1/send/1/mode	S		PRE, POST, GRP	Channel 1 sends 1 mode
/ch/1/send/1/plink	I	0..1		Channel 1 sends 1 pan link (0=individual)
/ch/1/send/1/pan	F	-100..100	201 steps	Channel 1 sends 1 pan
/ch/1/send/1/wid	F	-150..150	61 steps	Channel 1 sends 1 width (%)
/ch/1/postins	N			Channel 1 post insert node
/ch/1/postins/on	I	0..1		Channel 1 post insert on switch
/ch/1/postins/mode	S		FX, AUTO_X, AUTO_Y	Channel 1 post insert mode
/ch/1/postins/ins	S		NONE, FX1..FX16	Channel 1 post insert FX slot
/ch/1/postins/w	F	-12..12	241 steps	Channel 1 post insert autogain weight
/ch/1/postins/$stat	S		-, OK, N/A	Channel 1 post insert status [RO]
/ch/1/tags	S		80 chars max	Channel 1 tags
/ch/1/$fdr	F	-144..10	-oo..10 in 1024 steps	Channel 1 fader level as affected by dca (dB) [RO]
/ch/1/$mute	I	0..2		Channel 1 mute [RO]
/ch/1/$muteovr	I	0..1		Channel 1 mute override

Aux Settings

Command	Type	Range	Text	Description
/aux	N			Aux node
/aux/1	N	1..8		Aux 1 node
/aux/1/in	N			Aux 1 input node
/aux/1/in/set	N			Aux 1 input set node
/aux/1/in/set/$mode	S		M, ST, M/S	Aux 1 input mode [RO]
/aux/1/in/set/srcauto	I	0..1		Aux 1 input auto source switch
/aux/1/in/set/altsrc	I	0..1		Aux 1 input main/alt switch
/aux/1/in/set/inv	I	0..1		Aux 1 input phase invert switch
/aux/1/in/set/trim	F	-18..18	361 steps	Aux 1 input trim (dB)
/aux/1/in/set/bal	F	-9..9	181 steps	Aux 1 input balance (dB)
/aux/1/in/set/$g	F	-3..45	98 steps	Aux 1 input gain (dB)
/aux/1/in/set/$vph	I	0..1		Aux 1 input phantom power
/aux/1/in/conn	N			Aux 1 input connection node
/aux/1/in/conn/grp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX	Aux 1 input connection group
/aux/1/in/conn/in	I	1..64		Aux 1 input connection group index
/aux/1/in/conn/altgrp	S		OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES, USR, OSC, BUS, MAIN, MTX	Aux 1 alt input connection group
/aux/1/in/conn/altin	I	1..64		Aux 1 alt input connection group index
/aux/1/clink	I	0..1		Aux 1 custom link
/aux/1/col	I	1..12		Aux 1 color
/aux/1/name	S		16 chars max	Aux 1 name
/aux/1/icon	I	0..999		Aux 1 icon
/aux/1/led	I	0..1		Aux 1 scribble light
/aux/1/mute	I	0..1		Aux 1 mute
/aux/1/fdr	F	-144..10	-oo..10 in 1024 steps	Aux 1 fader level (dB)
/aux/1/pan	F	-100..100	201 steps	Aux 1 pan
/aux/1/wid	F	-150..150	61 steps	Aux 1 width (%)
/aux/1/$solo	I	0..1		Aux 1 solo
/aux/1/$sololed	I	0..2		Aux 1 solo LED [RO]
/aux/1/solosafe	I	0..1		Aux 1 solo safe
/aux/1/mon	S		A, B, A+B	Aux 1 monitor mode
/aux/1/eq	N			Aux 1 EQ node
/aux/1/eq/on	I	0..1		Aux 1 EQ switch
/aux/1/eq/mdl	S		STD, SOUL, E88, E84, F110, PULSAR	Aux 1 EQ model (see Appendix on EQ plugins for parameters details, OSC patterns in italic below correspond to STD)
/ aux /1/eq/mix	F	0..125	126 steps	Aux 1 EQ mix (%)
/ aux /1/eq/$solo	I	0..1		Aux 1 EQ solo
/ aux /1/eq/$solobd	I	0..6		Aux 1 EQ solo band
/ aux */1/eq/lg*	F	-15..15	301 steps	Aux 1 EQ low gain (dB)
/ aux */1/eq/lf*	F	20..2000	641 steps	Aux 1 EQ low frequency (Hz)
/ aux */1/eq/lq*	F	0.44..10	181 steps	Aux 1 EQ low Q
/ aux */1/eq/leq*	S		SHV, PEQ, CUT	Aux 1 EQ low type
/ aux */1/eq/1g*	F	-15..15	301 steps	Aux 1 EQ band 1 gain (dB)
/ aux */1/eq/1f*	F	20..20000	961 steps	Aux 1 EQ band 1 frequency (Hz)

/ aux /*1/eq/1q*	F	0.44..10	181 steps	Aux 1 EQ band 1 Q
/ aux /*1/eq/2g*	F	-15..15	301 steps	Aux 1 EQ band 2 gain (dB)
/ aux /*1/eq/2f*	F	20..20000	961 steps	Aux 1 EQ band 2 frequency (Hz)
/ aux /*1/eq/2q*	F	0.44..10	181 steps	Aux 1 EQ band 2 Q
/ aux /*1/eq/3g*	F	-15..15	301 steps	Aux 1 EQ band 3 gain (dB)
/ aux /*1/eq/3f*	F	20..20000	961 steps	Aux 1 EQ band 3 frequency (Hz)
/ aux /*1/eq/3q*	F	0.44..10	181 steps	Aux 1 EQ band 3 Q
/ aux /*1/eq/4g*	F	-15..15	301 steps	Aux 1 EQ band 4 gain (dB)
/ aux /*1/eq/4f*	F	20..20000	961 steps	Aux 1 EQ band 4 frequency (Hz)
/ aux /*1/eq/4q*	F	0.44..10	181 steps	Aux 1 EQ band 4 Q
/ aux /*1/eq/hg*	F	-15..15	301 steps	Aux 1 EQ high gain (dB)
/ aux /*1/eq/hf*	F	50..20000	833 steps	Aux 1 EQ high frequency (Hz)
/ aux /*1/eq/hq*	F	0.44..10	181 steps	Aux 1 EQ high Q
/ aux /*1/eq/heq*	S		SHV, PEQ, CUT	Aux 1 EQ high type
/aux/1/dyn	N			Aux 1 dynamic (compressor) node
/aux/1/dyn/on	I	0..1		Aux 1 compressor switch
/aux/1/dyn/thr	F	-36..12	97 steps	Aux 1 compressor threshold (dB)
/aux/1/dyn/depth	F	0..20	41 steps	Aux 1 compressor depth (dB)
/aux/1/dyn/fast	I	0..1		Aux 1 compressor fast switch
/aux/1/dyn/peak	I	0..1		Aux 1 compressor peak switch
/aux/1/dyn/ingain	F	0..100	101 steps	Aux 1 compressor input gain
/aux/1/dyn/cpeak	F	0..100	101 steps	Aux 1 compressor peak
/aux/1/dyn/cmode	S		COMP, LIM	Aux 1 compressor mode
/aux/1/preins	N			Aux 1 pre-insert node
/aux/1/preins/on	I	0..1		Aux 1 pre-insert switch
/aux/1/preins/ins	S		NONE, FX1..FX16	Aux 1 pre-insert FX slot
/aux/1/preins/$stat	S		-, OK, N/A	Aux 1 pre-insert status [RO]
/aux/1/main	N			Aux 1 Main node
/aux/1/main/1	N	1..4		Aux 1 Main 1 node
/aux/1/main/1/on	I	0..1		Aux 1 Main 1 on switch
/aux/1/main/1/lvl	F	-144..10	-oo..10 in 1024 steps	Aux 1 Main 1 fader level (dB)
/aux/1/main/1/pre	I	0..1		Aux 1 sent pre fader to Main 1
/aux/1/send	N			Aux 1 sends node
/aux/1/send/1	N	1..16		Aux 1 sends 1 node
/aux/1/send/1/on	I	0..1		Aux 1 sends 1 on switch
/aux/1/send/1/lvl	F	-144..10	-oo..10 in 1024 steps	Aux 1 sends 1 fader level (dB)
/aux/1/send/1/pon	I	0..1		Aux 1 sends 1 pre always on switch
/aux/1/send/1/ind	I	0..1		Aux 1 sends 1 individual link (0=link to bus)
/aux/1/send/1/mode	S		PRE, POST, GRP	Aux 1 sends 1 mode
/aux/1/send/1/plink	I	0..1		Aux 1 sends 1 pan link (0=individual)
/aux/1/send/1/pan	F	-100..100	201 steps	Aux 1 sends 1 pan
/aux/1/send/1/wid	F	-150..150	61 steps	Aux 1 sends 1 width (%)
/aux/1/tags	S	80 chars max		Aux 1 tags
/aux/1/$fdr	F	-144..10	-oo..10 in 1024 steps	Aux 1 fader level as affected by dca (dB)[RO]
/aux/1/$mute	I	0..2		Aux 1 mute {RO]
/aux/1/$muteovr	I	0..1		Aux 1 mute override

Bus Settings

Command	Type	Range	Text	Description
/bus	N			Bus node
/bus/1	N	1..16		Bus 1 node
/bus/1/in	N			Bus 1 input node
/bus/1/in/set	N			Bus 1 input set node
/bus/1/in/set/inv	I	0..1		Bus 1 input phase invert
/bus/1/in/set/trim	F	-18..18	361 steps	Bus 1 input trim (dB)
/bus/1/in/set/bal	F	-9..9	181 steps	Bus 1 input balance (dB)
/bus/1/col	I	1..12		Bus 1 color
/bus/1/name	S		16 chars max	Bus 1 name
/bus/1/icon	I	0..999		Bus 1 icon
/bus/1/led	I	0..1		Bus 1 scribble light
/bus/1/busmono	I	0..1		Bus 1 mono switch
/bus/1/mute	I	0..1		Bus 1 mute
/bus/1/fdr	F	-144..10	-oo..10 in 1024 steps	Bus 1 fader level (dB)
/bus/1/pan	F	-100..100	201 steps	Bus 1 pan
/bus/1/wid	F	-150..150	61 steps	Bus 1 width (%)
/bus/1/$solo	I	0..1		Bus 1 solo
/bus/1/$sololed	I	0..2		Bus 1 solo LED {RO]
/bus/1/mon	S		A, B, A+B	Bus 1 monitor mode
/bus/1/busmode	S		PRE, POST, GRP	Bus 1 mode
/bus/1/eq	N			Bus 1 EQ node
/bus/1/eq/on	I	0..1		Bus 1 EQ on switch
/bus/1/eq/mdl	S		STD, SOUL, E88, E84, F110, PULSAR, PIA	Bus 1 EQ model (see Appendix on EQ plugins for parameters details, OSC patterns in italic below correspond to STD)
/bus/1/eq/mix	F	0..100	126 steps	Bus 1 EQ mix
/bus/1/eq/$solo	I	0..1		Bus 1 EQ solo
/bus/1/eq/$solobd	I	0..1		Bus 1 EQ band solo
/bus/1/eq/lg	F	-15..15	301 steps	Bus 1 EQ low gain (dB)
/bus/1/eq/lf	F	20..2000	641 steps	Bus 1 EQ low frequency (Hz)
/bus/1/eq/lq	F	0.44..10	181 steps	Bus 1 EQ low Q
/bus/1/eq/leq	S		PEQ, SHV, CUT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48	Bus 1 EQ low type
/bus/1/eq/1g	F	-15..15	301 steps	Bus 1 EQ band 1 gain (dB)
/bus/1/eq/1f	F	20..20000	961 steps	Bus 1 EQ band 1 frequency (Hz)
/bus/1/eq/1q	F	0.44..10	181 steps	Bus 1 EQ band 1 Q
/bus/1/eq/2g	F	-15..15	301 steps	Bus 1 EQ band 2 gain (dB)
/bus/1/eq/2f	F	20..20000	961 steps	Bus 1 EQ band 2 frequency (Hz)
/bus/1/eq/2q	F	0.44..10	181 steps	Bus 1 EQ band 2 Q
/bus/1/eq/3g	F	-15..15	301 steps	Bus 1 EQ band 3 gain (dB)
/bus/1/eq/3f	F	20..20000	961 steps	Bus 1 EQ band 3 frequency (Hz)
/bus/1/eq/3q	F	0.44..10	181 steps	Bus 1 EQ band 3 Q
/bus/1/eq/4g	F	-15..15	301 steps	Bus 1 EQ band 4 gain (dB)
/bus/1/eq/4f	F	20..20000	961 steps	Bus 1 EQ band 4 frequency (Hz)
/bus/1/eq/4q	F	0.44..10	181 steps	Bus 1 EQ band 4 Q
/bus/1/eq/5g	F	-15..15	301 steps	Bus 1 EQ band 5 gain (dB)
/bus/1/eq/5f	F	20..20000	961 steps	Bus 1 EQ band 5 frequency (Hz)
/bus/1/eq/5q	F	0.44..10	181 steps	Bus 1 EQ band 5 Q
/bus/1/eq/6g	F	-15..15	301 steps	Bus 1 EQ band 6 gain (dB)

/bus/1/eq/6f	F	20..20000	961 steps	Bus 1 EQ band 6 frequency (Hz)
/bus/1/eq/6q	F	0.44..10	181 steps	Bus 1 EQ band 6 Q
/bus/1/eq/hg	F	-15..15	301 steps	Bus 1 EQ high gain (dB)
/bus/1/eq/hf	F	50..20000	833 steps	Bus 1 EQ high frequency (Hz)
/bus/1/eq/hq	F	0.44..10	181 steps	Bus 1 EQ high Q
/bus/1/eq/heq	S		PEQ, SHV, CUT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48	Bus 1 EQ high type
/bus/1/eq/tilt	F	-6..6	49 steps	Bus 1 EQ tilt level
/bus/1/dyn	N			Bus 1 dynamic (compressor) node
/bus/1/dyn/on	I	0..1		Bus 1 compressor switch
/bus/1/dyn/mdl	S		COMP, EXP, B160, B560, D241, ECL33, 9000C, SBUS, RED3, 76LA, LA, F670, BLISS, NSTR, WAVE, RIDE, 2250, L100	Bus 1 compressor model, (see Appendix on Compressor plugins for parameters details, OSC patterns in italic below correspond to COMP)
/bus/1/dyn/mix	F	0..100	101 steps	Bus 1 compressor mix (%)
/bus/1/dyn/gain	F	-6..12	37 steps	Bus 1 compressor gain (dB)
/bus/1/dyn/thr	F	-60..0	121 steps	Bus 1 compressor threshold (dB)
/bus/1/dyn/ratio	F	1.1..100		Bus 1 compressor ratio
/bus/1/dyn/knee	I	0..5		Bus 1 compressor knee
/bus/1/dyn/det	S		PEAK, RMS	Bus 1 compressor detect
/bus/1/dyn/att	F	0..120	121 steps	Bus 1 compressor attack (ms)
/bus/1/dyn/hld	F	1..200	200 steps	Bus 1 compressor hold (ms)
/bus/1/dyn/rel	F	4..4000	130 steps	Bus 1 compressor release (ms)
/bus/1/dyn/env	S		Lin, Log	Bus 1 compressor envelope
/bus/1/dyn/auto	I	0..1		Bus 1 compressor auto switch
/bus/1/dynxo	N			Bus 1 compressor crossover node
/bus/1/dynxo/depth	F	0..20	41 steps	Bus 1 compressor crossover depth (dB)
/bus/1/dynxo/type	S		OFF, LO6, LO12, HI6, HI12, PC	Bus 1 compressor crossover type
/bus/1/dynxo/f	F	20..20000	901 steps	Bus 1 compressor crossover frequency (Hz)
/bus/1/dynxo/$solo	I	0..1		Bus 1 compressor crossover solo
/bus/1/dynsc	N			Bus 1 compressor sidechain node
/bus/1/dynsc/type	S		OFF, LP12, HP12, BP	Bus 1 compressor sidechain type
/bus/1/dynsc/f	F	20..20000	901 steps	Bus 1 compressor sidechain frequency (Hz)
/bus/1/dynsc/q	F	0.44..10	181 steps	Bus 1 compressor sidechain Q
/bus/1/dynsc/src	S		SELF, BUS.1..BUS.16, MAIN.1..MAIN.4, MTX.1..MTX.8, AUX.1..AUX.8	Bus 1 compressor sidechain source
/bus/1/dynsc/tap	S		BUS, DYN, PFL, AFL, EQ, INS2	Bus 1 compressor sidechain tap
/bus/1/dynsc/$solo	I	0..1		Bus 1 compressor sidechain solo
/bus/1/preins	N			Bus 1 pre-insert node
/bus/1/preins/on	I	0..1		Bus 1 pre-insert switch
/bus/1/preins/ins	S		NONE, FX1..FX16	Bus 1 pre-insert FX slot
/bus/1/preins/$stat	S		-, OK, N/A	Bus 1 pre-insert status [RO]
/bus/1/main	N			Bus 1 Main node
/bus/1/main/1	N	1..4		Bus 1 Main 1 node
/bus/1/main/1/on	I	0..1		Bus 1 Main 1 on switch

/bus/1/main/1/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 Main 1 fader level (dB)
/bus/1/main/1/pre	I	0..1		Bus 1 sent pre fader to Main 1
/bus/1/send	N			Bus 1 sends node
/bus/1/send/1	N	1..8		Bus 1 sends 1 node
/bus/1/send/1/on	I	0..1		Bus 1 sends 1 on switch
/bus/1/send/1/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 sends 1 fader level (dB)
/bus/1/send/1/pre	I	0..1		Bus 1 sends 1 pre/post switch
/bus/1/send/MX1	N			Bus 1 matrix 1 sends node
/bus/1/send/MX1/on	I	0..1		Bus 1 matrix 1 on switch
/bus/1/send/MX1/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 1 fader level (dB)
/bus/1/send/MX1/pre	I	0..1		Bus 1 matrix 1 pre/post switch
/bus/1/send/MX2	N			Bus 1 matrix 2 sends node
/bus/1/send/MX2/on	I	0..1		Bus 1 matrix 2 on switch
/bus/1/send/MX2/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 2 fader level (dB)
/bus/1/send/MX2/pre	I	0..1		Bus 1 matrix 2 pre/post switch
/bus/1/send/MX3	N			Bus 1 matrix 3 sends node
/bus/1/send/MX3/on	I	0..1		Bus 1 matrix 3 on switch
/bus/1/send/MX3/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 3 fader level (dB)
/bus/1/send/MX3/pre	I	0..1		Bus 1 matrix 3 pre/post switch
/bus/1/send/MX4	N			Bus 1 matrix 4 sends node
/bus/1/send/MX4/on	I	0..1		Bus 1 matrix 4 on switch
/bus/1/send/MX4/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 4 fader level (dB)
/bus/1/send/MX4/pre	I	0..1		Bus 1 matrix 4 pre/post switch
/bus/1/send/MX5	N			Bus 1 matrix 5 sends node
/bus/1/send/MX5/on	I	0..1		Bus 1 matrix 5 on switch
/bus/1/send/MX5/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 5 fader level (dB)
/bus/1/send/MX5/pre	I	0..1		Bus 1 matrix 5 pre/post switch
/bus/1/send/MX6	N			Bus 1 matrix 6 sends node
/bus/1/send/MX6/on	I	0..1		Bus 1 matrix 6 on switch
/bus/1/send/MX6/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 6 fader level (dB)
/bus/1/send/MX6/pre	I	0..1		Bus 1 matrix 6 pre/post switch
/bus/1/send/MX7	N			Bus 1 matrix 7 sends node
/bus/1/send/MX7/on	I	0..1		Bus 1 matrix 7 on switch
/bus/1/send/MX7/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 7 fader level (dB)
/bus/1/send/MX7/pre	I	0..1		Bus 1 matrix 7 pre/post switch
/bus/1/send/MX8	N			Bus 1 matrix 8 sends node
/bus/1/send/MX8/on	I	0..1		Bus 1 matrix 8 on switch
/bus/1/send/MX8/lvl	F	-144..10	-oo..10 in 1024 steps	Bus 1 matrix 8 fader level (dB)
/bus/1/send/MX8/pre	I	0..1		Bus 1 matrix 8 pre/post switch
/bus/1/postins	N			Bus 1 post insert node
/bus/1/postins/on	I	0..1		Bus 1 post insert on switch
/bus/1/postins/ins	S		NONE, FX1..FX16	Bus 1 post insert mode
/bus/1/postins/$stat	S		-, OK, N/A	Bus 1 post insert status [RO]
/bus/1/tags	S		80 chars max	Bus 1 tags

/bus/1/$fdr	F	-144..10	-oo..10 in 1024 steps	Bus 1 fader level as affected by dca (dB)[RO]
/bus/1/$mute	I	0..2		Bus 1 mute [RO]
/bus/1/$muteovr	I	0..1		Bus 1 mute override

Mains Settings

Command	Type	Range	Text	Description
/main	N			Main node
/main/1	N	1..4		Main 1 node
/main/1/in	N			Main 1 input node
/main/1/in/set	N			Main 1 input set node
/main/1/in/set/inv	I	0..1		Main 1 input phase invert switch
/main/1/in/set/trim	F	-18..18	361 steps	Main 1 input trim
/main/1/in/set/bal	F	-9..9	181 steps	Main 1 input balance
/main/1/col	I	1..12		Main 1 color
/main/1/name	S		16 chars max	Main 1 name
/main/1/icon	I	0..999		Main 1 icon
/main/1/led	I	0..1		Main 1 scribble light
/main/1/busmono	I	0..1		Main 1 mono switch
/main/1/mute	I	0..1		Main 1 mute
/main/1/fdr	F	-144..10	-oo..10 in 1024 steps	Main 1 fader level (dB)
/main/1/pan	F	-100..100	201 steps	Main 1 pan
/main/1/wid	F	-150..150	61 steps	Main 1 width (%)
/main/1/$solo	I	0..1		Main 1 solo switch
/main/1/$sololed	I	0..2		Main 1 solo LED [RO]
/main/1/mon	S		A, B, A+B	Main 1 monitor mode
/main/1/eq	N			Main 1 EQ node
/main/1/eq/on	I	0..1		Main 1 EQ on switch
/main/1/eq/mdl	S		STD, SOUL, E88, E84, F110, PULSAR, PIA	Main 1 EQ model, (see Appendix on EQ plugins for parameters details, OSC patterns in italic below correspond to STD)
/main/1/eq/mix	F	0..100	126 steps	Main 1 EQ mix
/main/1/eq/$solo	I	0..1		Main 1 EQ solo
/main/1/eq/$solobd	I	0..1		Main 1 EQ band solo
/main/1/eq/lg	F	-15..15	301 steps	Main 1 EQ low gain (dB)
/main/1/eq/lf	F	20..2000	641 steps	Main 1 EQ low frequency (Hz)
/main/1/eq/lq	F	0.44..10	181 steps	Main 1 EQ low Q
/main/1/eq/leq	S		PEQ, SHV, CUT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48	Main 1 EQ low type
/main/1/eq/1g	F	-15..15	301 steps	Main 1 EQ band 1 gain (dB)
/main/1/eq/1f	F	20..20000	961 steps	Main 1 EQ band 1 frequency (Hz)
/main/1/eq/1q	F	0.44..10	181 steps	Main 1 EQ band 1 Q
/main/1/eq/2g	F	-15..15	301 steps	Main 1 EQ band 2 gain (dB)
/main/1/eq/2f	F	20..20000	961 steps	Main 1 EQ band 2 frequency (Hz)
/main/1/eq/2q	F	0.44..10	181 steps	Main 1 EQ band 2 Q
/main/1/eq/3g	F	-15..15	301 steps	Main 1 EQ band 3 gain (dB)
/main/1/eq/3f	F	20..20000	961 steps	Main 1 EQ band 3 frequency (Hz)
/main/1/eq/3q	F	0.44..10	181 steps	Main 1 EQ band 3 Q
/main/1/eq/4g	F	-15..15	301 steps	Main 1 EQ band 4 gain (dB)
/main/1/eq/4f	F	20..20000	961 steps	Main 1 EQ band 4 frequency (Hz)
/main/1/eq/4q	F	0.44..10	181 steps	Main 1 EQ band 4 Q

/main/1/eq/5g	F	-15..15	301 steps	Main 1 EQ band 5 gain (dB)
/main/1/eq/5f	F	20..20000	961 steps	Main 1 EQ band 5 frequency (Hz)
/main/1/eq/5q	F	0.44..10	181 steps	Main 1 EQ band 5 Q
/main/1/eq/6g	F	-15..15	301 steps	Main 1 EQ band 6 gain (dB)
/main/1/eq/6f	F	20..20000	961 steps	Main 1 EQ band 6 frequency (Hz)
/main/1/eq/6q	F	0.44..10	181 steps	Main 1 EQ band 6 Q
/main/1/eq/hg	F	-15..15	301 steps	Main 1 EQ high gain (dB)
/main/1/eq/hf	F	50..20000	833 steps	Main 1 EQ high frequency (Hz)
/main/1/eq/hq	F	0.44..10	181 steps	Main 1 EQ high Q
/main/1/eq/heq	S		PEQ, SHV, CUT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48	Main 1 EQ high type
/main/1/eq/tilt	F	-6..6	49 steps	Main 1 EQ tilt level
/main/1/dyn	N			Main 1 dynamic (compressor) node
/main/1/dyn/on	I	0..1		Main 1 compressor switch
/main/1/dyn/mdl	S		COMP, EXP, B160, B560, D241, ECL33, 9000C, SBUS, RED3, 76LA, LA, F670, BLISS, NSTR, WAVE, RIDE, 2250, L100	Main 1 compressor switch, (see Appendix on Compressor plugins for parameters details, OSC patterns in italic below correspond to COMP)
/main/1/dyn/mix	F	0..100	101 steps	Main 1 compressor mix (%)
/main/1/dyn/gain	F	-6..12	37 steps	Main 1 compressor gain (dB)
/main/1/dyn/thr	F	-60..0	121 steps	Main 1 compressor threshold (dB)
/main/1/dyn/ratio	F	1.1..100		Main 1 compressor ratio
/main/1/dyn/knee	I	0..5		Main 1 compressor knee
/main/1/dyn/det	S		PEAK, RMS	Main 1 compressor detect
/main/1/dyn/att	F	0..120	121 steps	Main 1 compressor attack (ms)
/main/1/dyn/hld	F	1..200	200 steps	Main 1 compressor hold (ms)
/main/1/dyn/rel	F	4..4000	130 steps	Main 1 compressor release (ms)
/main/1/dyn/env	S		LIN, LOG	Main 1 compressor envelope
/main/1/dyn/auto	I	0..1		Main 1 compressor auto switch
/main/1/dynxo	N			Main 1 compressor crossover node
/main/1/dynxo/depth	F	0..20	41 steps	Main 1 compressor crossover depth (dB)
/main/1/dynxo/type	S		OFF, LO6, LO12, HI6, HI12, PC	Main 1 compressor crossover type
/main/1/dynxo/f	F	20..20000	901 steps	Main 1 compressor crossover frequency (Hz)
/main/1/dynxo/$solo	I	0..1		Main 1 compressor crossover solo
/main/1/dynsc	N			Main 1 compressor sidechain node
/main/1/dynsc/type	S		OFF, LP12, HP12, BP	Main 1 compressor sidechain type
/main/1/dynsc/f	F	20..20000	901 steps	Main 1 compressor sidechain frequency (Hz)
/main/1/dynsc/q	F	0.44..10	181 steps	Main 1 compressor sidechain Q
/main/1/dynsc/src	S		SELF, BUS.1..BUS.16, MAIN.1..MAIN.4, MTX.1..MTX.8, AUX.1..AUX.8	Main 1 compressor sidechain source
/main/1/dynsc/tap	S		BUS, DYN, PFL, AFL, EQ, INS2	Main 1 compressor sidechain tap
/main/1/dynsc/$solo	I	0..1		Main 1 compressor sidechain solo
/main/1/preins	N			Main 1 pre-insert node
/main/1/preins/on	I	0..1		Main 1 pre-insert switch
/main/1/preins/ins	S		NONE, FX1..FX16	Main 1 pre-insert FX slot
/main/1/preins/$stat	S		-, OK, N/A	Main 1 pre-insert status [RO]

/main/1/send	N			Main 1 sends node
/main/1/send/MX1	N			Main 1 matrix sends node
/main/1/send/MX1/on	I	0..1		Main 1 matrix 1 on switch
/main/1/send/MX1/lvl	F	-144..10	-oo..10 in 1024 steps	Main 1 matrix 1 fader level (dB)
/main/1/send/MX1/pre	I	0..1		Main 1 matrix 1 pre/post switch
/main/1/send/MX2	N			Main 2 matrix sends node
/main/1/send/MX2/on	I	0..1		Main 2 matrix 1 on switch
/main/1/send/MX2/lvl	F	-144..10	-oo..10 in 1024 steps	Main 2 matrix 1 fader level (dB)
/main/1/send/MX2/pre	I	0..1		Main 2 matrix 1 pre/post switch
/main/1/send/MX3	N			Main 3 matrix sends node
/main/1/send/MX3/on	I	0..1		Main 3 matrix 1 on switch
/main/1/send/MX3/lvl	F	-144..10	-oo..10 in 1024 steps	Main 3 matrix 1 fader level (dB)
/main/1/send/MX3/pre	I	0..1		Main 3 matrix 1 pre/post switch
/main/1/send/MX4	N			Main 4 matrix sends node
/main/1/send/MX4/on	I	0..1		Main 4 matrix 1 on switch
/main/1/send/MX4/lvl	F	-144..10	-oo..10 in 1024 steps	Main 4 matrix 1 fader level (dB)
/main/1/send/MX4/pre	I	0..1		Main 4 matrix 1 pre/post switch
/main/1/send/MX5	N			Main 5 matrix sends node
/main/1/send/MX5/on	I	0..1		Main 5 matrix 1 on switch
/main/1/send/MX5/lvl	F	-144..10	-oo..10 in 1024 steps	Main 5 matrix 1 fader level (dB)
/main/1/send/MX5/pre	I	0..1		Main 5 matrix 1 pre/post switch
/main/1/send/MX6	N			Main 6 matrix sends node
/main/1/send/MX6/on	I	0..1		Main 6 matrix 1 on switch
/main/1/send/MX6/lvl	F	-144..10	-oo..10 in 1024 steps	Main 6 matrix 1 fader level (dB)
/main/1/send/MX6/pre	I	0..1		Main 6 matrix 1 pre/post switch
/main/1/send/MX7	N			Main 7 matrix sends node
/main/1/send/MX7/on	I	0..1		Main 7 matrix 1 on switch
/main/1/send/MX7/lvl	F	-144..10	-oo..10 in 1024 steps	Main 7 matrix 1 fader level (dB)
/main/1/send/MX7/pre	I	0..1		Main 7 matrix 1 pre/post switch
/main/1/send/MX8	N			Main 8 matrix sends node
/main/1/send/MX8/on	I	0..1		Main 8 matrix 1 on switch
/main/1/send/MX8/lvl	F	-144..10	-oo..10 in 1024 steps	Main 8 matrix 1 fader level (dB)
/main/1/send/MX8/pre	I	0..1		Main 8 matrix 1 pre/post switch
/main/1/postins	N			Main 1 post insert node
/main/1/postins/on	I	0..1		Main 1 post insert on switch
/main/1/postins/ins	S		NONE, FX1..FX16	Main 1 post insert mode
/main/1/postins/$stat	S		-, OK, N/A	Main 1 post insert status [RO]
/main/1/dly	N			Main 1 delay node
/main/1/dly/on	I	0..1		Main 1 delay on switch
/main/1/dly/m	F	0.1..100	1000 steps	Main 1 delay (meters)
/main/1/tags	S		80 chars max	Main 1 tags
/main/1/$fdr	F	-144..10	-oo..10 in 1024 steps	Main 1 fader level as affected by dca (dB)[RO]
/main/1/$mute	I	0..2		Main 1 mute [RO]

/main/1/$muteovr	I	0..1		Main 1 mute override

Matrix Settings

Command	Type	Range	Text	Description
/mtx	N			Matrix node
/mtx/1	N	1..8		Matrix 1 node
/mtx/1/in	N			Matrix 1 input node
/mtx/1/in/set	N			Matrix 1 input set node
/mtx/1/in/set/inv	I	0..1		Matrix 1 input phase invert
/mtx/1/in/set/trim	F	-18..18	361 steps	Matrix 1 input trim
/mtx/1/in/set/bal	F	-9..9	181 steps	Matrix 1 input balance
/mtx/1/dir	N			Matrix 1 direct input signal
/mtx/1/dir/1	N	1..2		Matrix 1 direct in 1 node
/mtx/1/dir/1/on	I	0..1		Matrix 1 direct in 1 on switch
/mtx/1/dir/1/lvl	F	-144..10	-oo..10 in 1024 steps	Matrix 1 direct in 1 fader level (dB)
/mtx/1/dir/1/inv	I	0..1		Matrix 1 direct in 1 invert
/mtx/1/dir/1/in	S		OFF, CH.1..CH.40, AUX.1..AUX.8, AES, MON.A, MON.B, MON.BUS	Matrix 1 direct in 1 input
/mtx/1/dir/1/tap	S		PRE, POST	Matrix 1 direct in 1 tap
/mtx/1/col	I	1..12		Matrix 1 color
/mtx/1/name	S		16 chars max	Matrix 1 name
/mtx/1/icon	I	0..999		Matrix 1 icon
/mtx/1/led	I	0..1		Matrix 1 scribble light
/mtx/1/busmono	I	0..1		Matrix 1 mono switch
/mtx/1/mute	I	0..1		Matrix 1 mute
/mtx/1/fdr	F	-144..10	-oo..10 in 1024 steps	Matrix 1 fader level (dB)
/mtx/1/pan	F	-100..100	201 steps	Matrix 1 pan
/mtx/1/wid	F	-150..150	61 steps	Matrix 1 width (%)
/mtx/1/$solo	I	0..1		Matrix 1 solo switch
/mtx/1/$sololed	I	0..2		Matrix 1 solo LED [RO]
/mtx/1/mon	S		A, B, A+B	Matrix 1 monitor mode
/mtx/1/eq	N			Matrix 1 EQ node
/mtx/1/eq/on	I	0..1		Matrix 1 EQ on switch
/mtx/1/eq/mdl	S		STD, SOUL, E88, E84, F110, PULSAR, PIA	Matrix 1 EQ model, (see Appendix on EQ plugins for parameters details, OSC patterns in italic below correspond to STD)
/mtx/1/eq/mix	F	0..125	126 steps	Matrix 1 EQ mix (%)
/mtx/1/eq/$solo	I	0..1		Matrix 1 EQ solo
/mtx/1/eq/$solobd	I	0..8		Matrix 1 EQ solo band
/mtx/1/eq/lg	F	-15..15	301 steps	Matrix 1 EQ low gain (dB)
/mtx/1/eq/lf	F	20..2000	641 steps	Matrix 1 EQ low frequency
/mtx/1/eq/lq	F	0.44..10	181 steps	Matrix 1 EQ low Q
/mtx/1/eq/leq	S		SHV, PEQ, CUT	Matrix 1 EQ low type
/mtx/1/eq/1g	F	-15..15	301 steps	Matrix 1 EQ band 1 gain (dB)
/mtx/1/eq/1f	F	20..20000	961 steps	Matrix 1 EQ band 1 frequency (Hz)
/mtx/1/eq/1q	F	0.44..10	181 steps	Matrix 1 EQ band 1 Q
/mtx/1/eq/2g	F	-15..15	301 steps	Matrix 1 EQ band 2 gain (dB)
/mtx/1/eq/2f	F	20..20000	961 steps	Matrix 1 EQ band 2 frequency (Hz)
/mtx/1/eq/2q	F	0.44..10	181 steps	Matrix 1 EQ band 2 Q
/mtx/1/eq/3g	F	-15..15	301 steps	Matrix 1 EQ band 3 gain (dB)
/mtx/1/eq/3f	F	20..20000	961 steps	Matrix 1 EQ band 3 frequency (Hz)
/mtx/1/eq/3q	F	0.44..10	181 steps	Matrix 1 EQ band 3 Q
/mtx/1/eq/4g	F	-15..15	301 steps	Matrix 1 EQ band 4 gain (dB)

/mtx/1/eq/4f	F	20..20000	961 steps	Matrix 1 EQ band 4 frequency (Hz)
/mtx/1/eq/4q	F	0.44..10	181 steps	Matrix 1 EQ band 4 Q
/mtx/1/eq/5g	F	-15..15	301 steps	Matrix 1 EQ band 5 gain (dB)
/mtx/1/eq/5f	F	20..20000	961 steps	Matrix 1 EQ band 5 frequency (Hz)
/mtx/1/eq/5q	F	0.44..10	181 steps	Matrix 1 EQ band 5 Q
/mtx/1/eq/6g	F	-15..15	301 steps	Matrix 1 EQ band 6 gain (dB)
/mtx/1/eq/6f	F	20..20000	961 steps	Matrix 1 EQ band 6 frequency (Hz)
/mtx/1/eq/6q	F	0.44..10	181 steps	Matrix 1 EQ band 6 Q
/mtx/1/eq/hg	F	-15..15	301 steps	Matrix 1 EQ high gain (dB)
/mtx/1/eq/hf	F	50..20000	833 steps	Matrix 1 EQ high frequency (Hz)
/mtx/1/eq/hq	F	0.44..10	181 steps	Matrix 1 EQ high Q
/mtx/1/eq/heq	S		SHV, PEQ, CUT	Matrix 1 EQ high type
/mtx/1/eq/tilt	F	-6..6	49 steps	Matrix 1 EQ tilt level (dB)
/mtx/1/dyn	N			Matrix 1 dynamic (compressor) node
/mtx/1/dyn/on	I	0..1		Matrix 1 compressor switch
/mtx/1/dyn/mdl	S		COMP, EXP, B160, B560, D241, ECL33, 9000C, SBUS, RED3, 76LA, LA, F670, BLISS, NSTR, WAVE, RIDE, 2250, L100	Matrix 1 compressor model, (see Appendix on Compressor plugins for parameters details, OSC patterns in italic below correspond to COMP)
/mtx/1/dyn/mix	F	0..100	101 steps	Matrix 1 compressor mix (%)
/mtx/1/dyn/gain	F	-6..12	37 steps	Matrix 1 compressor gain (dB)
/mtx/1/dyn/thr	F	-60..0	121 steps	Matrix 1 compressor threshold (dB)
/mtx/1/dyn/ratio	F	1.1..100		Matrix 1 compressor ratio
/mtx/1/dyn/knee	I	0..5		Matrix 1 compressor knee
/mtx/1/dyn/det	S		PEAK, RMS	Matrix 1 compressor detect
/mtx/1/dyn/att	F	0..120	121 steps	Matrix 1 compressor attack (ms)
/mtx/1/dyn/hld	F	1..200	200 steps	Matrix 1 compressor hold (ms)
/mtx/1/dyn/rel	F	4..4000	130 steps	Matrix 1 compressor release (ms)
/mtx/1/dyn/env	S		LIN, LOG	Matrix 1 compressor envelope
/mtx/1/dyn/auto	I	0..1		Matrix 1 compressor auto switch
/mtx/1/dynxo	N			Matrix 1 compressor crossover node
/mtx/1/dynxo/depth	F	0..20	41 steps	Matrix 1 compressor crossover depth (dB)
/mtx/1/dynxo/type	S		OFF, LO6, LO12, HI6, HI12, PC	Matrix 1 compressor crossover type
/mtx/1/dynxo/f	F	20..20000	901 steps	Matrix 1 compressor crossover frequency (Hz)
/mtx/1/dynxo/$solo	I	0..1		Matrix 1 compressor crossover solo
/mtx/1/dynsc	N			Matrix 1 compressor sidechain node
/mtx/1/dynsc/type	S		OFF, LP12, HP12, BP	Matrix 1 compressor sidechain type
/mtx/1/dynsc/f	F	20..20000	901 steps	Matrix 1 compressor sidechain frequency (Hz)
/mtx/1/dynsc/q	F	0.44..10	181 steps	Matrix 1 compressor sidechain Q
/mtx/1/dynsc/src	S		SELF, BUS.1..BUS.16, MAIN.1..MAIN.4, MTX.1..MTX.8, AUX.1..AUX.8	Matrix 1 compressor sidechain source
/mtx/1/dynsc/tap	S		BUS, DYN, PFL, AFL, EQ, INS2	Matrix 1 compressor sidechain tap
/mtx/1/dynsc/$solo	I	0..1		Matrix 1 compressor sidechain solo
/mtx/1/preins	N			Matrix 1 pre-insert node
/mtx/1/preins/on	I	0..1		Matrix 1 pre-insert switch
/mtx/1/preins/ins	S		NONE, FX1..FX16	Matrix 1 pre-insert FX slot
/mtx/1/preins/$stat	S		-,OK, N/A	Matrix 1 pre-insert status [RO]

/mtx/1/postins	N			Matrix 1 post insert node
/mtx/1/postins/on	I	0..1		Matrix 1 post insert on switch
/mtx/1/postins/ins	S		NONE, FX1..FX16	Matrix 1 post insert mode
/mtx/1/postins/$stat	S		-,OK, N/A	Matrix 1 post insert status [RO]
/mtx/1/dly	N			Matrix 1 delay node
/mtx/1/dly/on	I	0..1		Matrix 1 delay on switch
/mtx/1/dly/m	F	0.1..100	1000 steps	Maitrix1 delay (meters)
/mtx/1/tags	S		80 chars max	Matrix 1 tags
/mtx/1/$fdr	F	-144..10	-oo..10 in 1024 steps	Matrix 1 fader level as affected by dca (dB)[RO]
/mtx/1/$mute	I	0..2		Matrix 1 mute [RO]
/mtx/1/$muteovr	I	0..1		Matrix 1 mute override

DCA Settings

Command	Type	Range	Text	Description
/dca	N			DCA node
/dca/1	N	1..16		DCA 1 node
/dca/1/name	S		8 chars max	DCA 1 name
/dca/1/col	I	1..12		DCA 1 color
/dca/1/icon	I	0..999		DCA 1 icon
/dca/1/led	I	0..1		DCA 1 scribble light
/dca/1/mute	I	0..1		DCA 1 mute
/dca/1/fdr	F	-144..10	-oo..10 in 1024 steps	DCA 1 fader (dB)
/dca/1/$solo	I	0..1		DCA 1 solo
/dca/1/$sololed	I	0..1		DCA 1 solo LED [RO]
/dca/1/mon	S		A, B, A+B	DCA 1 monitor mode

Mutegroup Settings

Command	Type	Range	Text	Description
/mgrp	N			Mutegroup node
/mgrp/1	N	1..8		Mutegroup 1 node
/mgrp/1/name	S		8 chars max	Mutegroup 1 name
/mgrp/1/mute	I	0..1		Mutegroup 1 mute

Effects Settings

Command	Type	Range	Text	Description
/fx	N			FX node
/fx/1	N	1..16		FX 1 node
/fx/1/mdl	S		For /fx/1../fx/8: NONE, EXT, HALL, ROOM, CHAMBER, PLATE, CONCERT, AMBI, V-ROOM, V-REV, V-PLATE, GATED, REVERSE, DEL/REV, SHIMMER, SPRING, DIMCRS, CHORUS, FLANGER, ST-DL, TAP-DL, TAPE-DL, OILCAN, BBD-DL, PITCH, D-PITCH, VSS3, BPLATE, GEQ, PIA, DOUBLE, PCORR, LIMITER, DE-S2, ENHANCE, EXCITER, P-BASS, ROTARY, PHASER, PANNER, TAPE, MOOD, SUB, RACKAMP, UKROCK, ANGEL, JAZZC, DELUXE, BODY, SOUL, E88, E84, F110, PULSAR, MACH4, C5-CMB, SUB-M, V-IMG, SPKMAN, DEQ3, *EVEN*, *SOUL*, *VINTAGE*, *BUS*, *MASTER* For /fx/9../fx/16: NONE, EXT, GEQ, PIA, DOUBLE, PCORR, LIMITER, DE-S2, ENHANCE, EXCITER, P-BASS, ROTARY, PHASER, PANNER, TAPE, MOOD, SUB, RACKAMP, UKROCK, ANGEL, JAZZC, DELUXE, BODY, SOUL, E88, E84, F110, PULSAR, MACH4, C5-CMB, SUB-M, V-IMG, SPKMAN, DEQ3, *EVEN*, *SOUL*, *VINTAGE*, *BUS*, *MASTER*	FX 1 model (see Appendix for details, graphics and parameter values)
/fx/1/fxmix	F	0..100	101 steps	FX 1 mix % (depends on FX)
/fx/1/$esrc	I	0..400		FX 1 source [RO]
/fx/1/$emode	S		M, ST, M/S	FX 1 mode [RO]
/fx/1/$a_chn	I	0..76		FX 1 channel assigned to it [RO]
/fx/1/$a_pos	I	0..1		FX 1 channel insert (0=pre, 1=post) [RO]
/fx/1/…	/fx/1/… contains up to 64 parameters that depend on the selected model (/fx/1/mdl), as listed in the Appendix section			

Cards Settings

Command	Type	Range	Text	Description
/cards	N			Cards node
/cards/$type	S		NONE, WLIVE, WDANTE, WLINK	Cards type [RO]
/cards/$ver	S		32 chars max	Cards version [RO]
/cards/wlive	N			Cards W-Live node
/cards/wlive/$sdlink	S		IND, PAR	Cards W-Live SD parallel mode
/cards/wlive/$actlink	S		IND, PAR	Cards W-Live ACT link [RO]
/cards/wlive/$battstate	S		NONE, GOOD, LOW	Cards W-Live battery status [RO]
/cards/wlive/autoin	S		OFF, 1, 2	Cards W-Live auto input
/cards/wlive/meters	I	0..1		Cards show meters
/cards/wlive/1	N	1..2		Cards W-Live 1 node
/cards/wlive/1/$ctl	N			Cards W-Live 1 ctl node
/cards/wlive/1/$ctl/control	S		STOP, PPAUSE, PLAY, REC	Cards W-Live 1 control
/cards/wlive/1/$ctl/opensession	I	0..100		Cards W-Live 1 open session #
/cards/wlive/1/$ctl/editmarker	I	0..100		Cards W-Live 1 edit marker (set marker to current PAUSE time or last start PLAY time)
/cards/wlive/1/$ctl/gotomarker	I	0..101		Cards W-Live 1 goto marker # 101 is used to validate stime data
/cards/wlive/1/$ctl/deletemarker	I	0..100		Cards W-Live 1 delete marker #
/cards/wlive/1/$ctl/deletesession	I	0..100		Cards W-Live 1 delete session #
/cards/wlive/1/$ctl/stime	F	0..36000000	36000000 steps	Cards W-Live 1 time (ms). Must be followed by a $ctl/gotomarker 101 to be taken into account
/cards/wlive/1/$ctl/namesession	S		19 chars max	Cards W-Live 1 name session. Works only in STOP mode.
/cards/wlive/1/$ctl/setmarker	I	0..1		Cards W-Live 1 set marker
/cards/wlive/1/$ctl/formatsdcard	I	0..1		Cards W-Live 1 format SD card
/cards/wlive/1/cfg	N			Cards W-Live 1 cfg node
/cards/wlive/1/cfg/rectracks	S		32, 16, 8	Cards W-Live 1 rec tracks
/cards/wlive/1/cfg/playmode	S		PLAY, A->B, LOOP	Cards W-Live 1 play mode
/cards/wlive/1/$stat	N			Cards W-Live 1 status node
/cards/wlive/1/$stat/state	S		STOP, PPAUSE, PLAY, REC	Cards W-Live 1 state [RO]
/cards/wlive/1/$stat/etime	F	0..36000000	36000000 steps	Cards W-Live 1 etime (current time)
/cards/wlive/1/$stat/sdfree	F	0..36000000	36000000 steps	Cards W-Live 1 SD free space
/cards/wlive/1/$stat/sdsize	I	0..1024		Cards W-Live 1 SD size (Gb) [RO]
/cards/wlive/1/$stat/sdstate	S		NONE, READY, PROTECT, ERASE, ERROR	Cards W-Live 1 SD state [RO]
/cards/wlive/1/$stat/sessionlist	S		Ex: 2020-04-04 10:16:36, 2020-01-27 19:59:02, …	Cards W-Live 1 list of session recorded date and time
/cards/wlive/1/$stat/markerlist	S			Cards W-Live 1 current marker time

/cards/wlive/1/$stat/snamelist	S		Ex: CC Hard Candy Fi, CC Mr Jones	Cards W-Live 1 session names list[14] [RO]
/cards/wlive/1/$stat/sessions	I	0..100		Cards W-Live 1 total number of sessions [RO]
/cards/wlive/1/$stat/markers	I	0..100		Cards W-Live 1 total number of markers [RO]
/cards/wlive/1/$stat/sessionlen	F	0..36000000	36000000 steps	Cards W-Live 1 session length [RO]
/cards/wlive/1/$stat/sessionpos	I	0..100		Cards W-Live 1 session position
/cards/wlive/1/$stat/markerpos	I	0..100		Cards W-Live 1 marker position
/cards/wlive/1/$stat/tracks	S		32, 16, 8	Cards W-Live 1 track number in current session [RO]
/cards/wlive/1/$stat/rate	S		44.1, 48	Cards W-Live 1 sample rate [RO]
/cards/wlive/1/$stat/linkid	I	0..5		Cards W-Live 1 link state [RO]
/cards/wlive/1/$stat/start	F	0..36000000	36000000 steps	Cards W-Live 1 start
/cards/wlive/1/$stat/stop	F	0..36000000	36000000 steps	Cards W-Live 1 stop
/cards/wlive/1/$stat/errormessage	S		32 chars max	Cards W-Live 1 error message [RO]
/cards/wlive/1/$stat/errorcode	I	0..34		Cards W-Live 1 error code [RO]

[14] Only the first name of the list is returned by std OSC command. You must use the node definition command (OSC or native interface) to get the full contents.

USB Player Settings

Command	Type	Range	Text	Description
/play[15]	N			USB Player node
/play/$songs	S		List of strings	List of songs [RO][16]
/play/$actlist	S		256 chars max	Path to USB files [RO]
/play/$actidx	I		1..n	Current active entry in the playlist
/play/$actionidx	I		1..n	[RO]
/play/$playfile	S		256 chars max	Full path to a song to play using `/play/$action ,s PLAYFILE`
/play/$action	S		IDLE, STOP, PLAY, PAUSE, NEXT, PREV, PLAYFILE	USB Player action
/play/$actstate	S		STOP, PLAY, PAUSE, ERROR	USB Player active state [RO]
/play/$actfile	S		256 chars max	USB Player active file [RO]
/play/$song	S		64 chars max	USB Player song [RO]
/play/$album	S		64 chars max	USB Player album [RO]
/play/$artist	S		64 chars max	USB Player artist [RO]
/play/$pos	F	0..35999	36000 steps	USB Player position
/play/$total	F	0..35999	36000 steps	USB Player total time [RO]
/play/$resolution	S		16, 24	USB Player resolution [RO]
/play/$channels	S		1, 2, 3, 4	USB Player channels [RO]
/play/$rate	S		44.1, 48	USB Player sample rate [RO]
/play/$format	S		WAV, MP3, FLAC	USB Player format [RO]
/play/repeat	I	0..1		USB Player repeat
/rec	N			USB Recorder node
/rec/$actstate	S		STOP, REC, PAUSE, ERROR	USB Recorder active state
/rec/$actfile	S			USB Recorder active filename
/rec/$action	S		STOP, REC, PAUSE, NEWFILE	USB Recorder action
/rec/path	S			USB Recorder filename path
/rec/resolution	S			USB Recorder resolution
/rec/channels	S			USB Recorder channels
/rec/$time	F			USB Recorder time

Global Settings

/$globals	N			Global Settings node
/$globals/clkrate	F	44100, 48000		Master clock rate
/$globals/clksrc	S		INT, A, B, C, AES, CARD, MOD	Master clock source
/$globals/startmute	I	0..1		Mute outputs on startup
/$globals/usbacfg	S		2/2, 8/8, 16/16, 32/32, 48/48	USB Input/Output configuration
/$globals/sccfg	S		AUTO, 0/32, 1/31, 2/30, 3/29, 4/28, 5/27, 6/26, 7/25, 8/24, 9/23, 10/22,	SC Configuration

[15] These commands are valid only when a playlist is active, and opened.

[16] Will provide only the first element of the list. Use the node level request to get the full list of songs in the playlist, for ex: `/play~~~,s~~?~~~`

			11/21, 12/20, 13/19, 14/18, 15/17, 16/16, 17/15, 18/14, 19/13, 20/12, 21/11, 22/10, 23/9, 24/8, 25/7, 26/6, 27/5, 28/4, 29/3, 30/2, 31/1, 32/0	
/$globals/harmt	N			HA remote node
/$globals/harmt/a	I	0..1		Enable HA remote on AES-A
/$globals/harmt/b	I	0..1		Enable HA remote on AES-B
/$globals/harmt/c	I	0..1		Enable HA remote on AES-C
/$globals/custsync	N			Custom Sync node
/$globals/ custsync /a	I	0..1		Enable Cust Sync on AES-A
/$globals/ custsync /b	I	0..1		Enable Cust Sync on AES-B
/$globals/ custsync /c	I	0..1		Enable Cust Sync on AES-C

WING ce_data OSC commands list

Control Settings, listed below, form a large set of OSC commands and parameters, all[17] under the ***ce_data*** section in JSON snapshot files.

Control Settings

Command	Type	Range	Text	Description
/$ctl	N			Control node
/$ctl/$stat	N			Control status node
/$ctl/$stat/selidx	I	1..76		Channel strip selected ID[18]
/$ctl/$stat/pageidx	I	0..30		Channel page ID
/$ctl/$stat/bandidx	I	1..8		Channel EQ band ID
/$ctl/$stat/sof	I	-1..76		Sends on fader (SoF) status [-1 is the currently selected channel]
/$ctl/$stat/cnslock	S		19 chars when locked 0 chars if unlocked	Console lock [RO] – The console lock string is made of 19 characters **0** or **1** depending on which screen buttons were pressed to lock the console. Characters 1 to 7 map to the buttons on the screen left, starting with **HOME** [**ASSIGN** is char #7], and characters 18 & 19 map to the buttons on the right side of the screen. Other characters are always **0**. Ex: **1001001000000000000** - buttons **HOME**, **ROUTING** and **ASSIGN** have been used to lock the desk.
/$ctl/cfg	N			Control config node
/$ctl/cfg/lights	N			Lights node
/$ctl/cfg/lights/btns	I	0..100		Buttons backlight intensity
/$ctl/cfg/lights/leds	I	5..100		Buttons/LED light intensity
/$ctl/cfg/lights/meters	I	0..100		Meters intensity
/$ctl/cfg/lights/rgbleds	I	0..100		Color LED intensity (scribble lights)
/$ctl/cfg/lights/chlcds	I	5..100		Channel LCD intensity (scribble backlight)
/$ctl/cfg/lights/chlcdctr	I	0..100		Channel LCD contrast (scribble contrast)
/$ctl/cfg/lights/chedit	I	5..100		Channel strip intensity
/$ctl/cfg/lights/main	I	5..100		Touchscreen intensity
/$ctl/cfg/lights/glow	I	0..100		Under console light intensity
/$ctl/cfg/lights/patch	I	0..100		Patch panel light intensity
/$ctl/cfg/lights/lamp	I	0..100		Lamp light intensity
/$ctl/cfg/rta	N			RTA node
/$ctl/cfg/rta/homedisp	S		OFF, 1/3, FULL	RTA home size/mode
/$ctl/cfg/rta/homecol	S		RD25, RD50, RD75, AM25, AM50, AM75, BL25, BL50, BL75	RTA home color
/$ctl/cfg/rta/hometap	S		IN, EQ, POST	RTA home tap

[17] With the exception of the /$ctl/$stat, and parameters /$ctl/cfg/$noautosave and /$ctl/cfg/savenow

[18] The get command reports values between 0 and 75, but index 1 to 76 should be used when setting values.

/$ctl/cfg/rta/eqdisp	S		Off, 1/4, 1/3, 1/2, OVL/3, OVL	RTA EQ size/mode
/$ctl/cfg/rta/eqcol	S		RD25, RD50, RD75, AM25, AM50, AM75, BL25, BL50, BL75	RTA EQ color
/$ctl/cfg/rta/cheqtap	S		PRE, POST	RTA EQ tap
/$ctl/cfg/rta/chflttap	S		PRE, POST	RTA Channel filter tap
/$globals/muteovr	I	0..1		Chan strip mute overrides mute group
/$ctl/cfg/soloexcl	I	0..1		Solo exclusive
/$ctl/cfg/selfsolo	I	0..1		Select follows solo
/$ctl/cfg/solofsel	I	0..1		Solo follows select
/$ctl/cfg/sof2solo	I	0..1		Bus SOF activates solo
/$ctl/cfg/layerlinkl	I	0..1		User Layer link left/center
/$ctl/cfg/layerlinkr	I	0..1		User Layer link center/right
/$ctl/cfg/autoview	I	0..1		Screen follows channel strip
/$ctl/cfg/csctouch	I	0..1		Channel strip touch select
/$ctl/cfg/autosel_L	I	0..1		Channel auto select left
/$ctl/cfg/autosel_C	I	0..1		Channel auto select center
/$ctl/cfg/autosel_R	I	0..1		Channel auto select right
/$ctl/cfg/fdrbanking	I	0..1		Full fader paging
/$ctl/cfg/soffdr	S		L/C, All	SOF Faders (L/C: left/center)
/$ctl/cfg/sofbutton	S		AUTO, ON, FLASH	SOF button mode
/$ctl/cfg/sofframe	I	0..1		SOF frame
/$ctl/cfg/sofmode	I	0..1		Alternative SOF mode
/$ctl/cfg/seldblclick	I	0..1		Double click select takes you Home
/$ctl/cfg/usrmode	S		BUS, CC	Use F1-F3 as BUS or Custom Control
/$ctl/layer	N			Layer node
/$ctl/layer/L	N			Left layer node

/$ctl/layer/L/sel	I	1..19	1..7 settable
			8..19 fixed/pre-assigned
				Left layer select
				`1: Ch 1..Ch 12`
				`2: Ch 13..Ch 24`
				`3: Ch 25..Ch 36`
				`4: Ch 36..Ch 40 / Aux 1..Aux 8`
				`5: Bus 1..Bus 12`
				`6: User 1`
				`7: User 2`
				`8: Ch 1..Ch 8`
				`9: Ch 9..Ch 16`
				`10: Ch 17..Ch 24`
				`11: Ch 25..Ch 32`
				`12: Ch 33..Ch 40`
				`13: Aux 1..Aux 8`
				`14: Bus 1..Bus 8`
				`15: Bus 9..Bus 16`
				`16: Main 1..Main 4`
				`17: Matrix 1..Matrix 8`
				`18: DCA 1..DCA 8`
				`19: DCA 9..DCA 16`

/$ctl/layer/L/1	N	1..7		Left layer 1 node (see above)
/$ctl/layer/L/1/ofs	I	0..12		Left layer 1 offset (from <4 or4> keys for ex.)
/$ctl/layer/L/1/name	S		10 chars max CH1-12, CH13-24, CH25-36, CH37-AUX, BUSES, USER1, USER2	Left layer 1 name

/$ctl/layer/L/1/1	N	1..24		Left layer 1, node 1
/$ctl/layer/L/1/1/type	S		OFF, CH, BUS, DCA, MIDI, SEND, FX	Left layer 1, node 1 type (OSC patterns in italic below correspond to MIDI type)
/$ctl/layer/L/1/1/i	I	1..127		Left layer 1, node 1 index
/$ctl/layer/L/1/1/dst	I	1..16		Left layer 1, node 1 destination index (used for type SEND)
/$ctl/layer/L/1/1/val	*I*	*0..127*		*Left layer 1, node 1 value (when type MIDI)*
/$ctl/layer/C	N			Center layer node

/$ctl/layer/C/sel	I	1..19	1..6 settable
			7 not used
			8..19 fixed/pre-assigned
				Center layer select:
				`1: DCA`
				`2: Main/Matrix`
				`3: Aux/FX`
				`4: Bus/Master`
				`5: User 1`
				`6: User 2`
				`7: No-op`
				`8: Ch 1..Ch 8`
				`9: Ch 9..Ch 16`
				`10: Ch 17..Ch 24`
				`11: Ch 25..Ch 32`
				`12: Ch 33..Ch 40`
				`13: Aux 1..Aux 8`
				`14: Bus 1..Bus 8`
				`15: Bus 9..Bus 16`
				`16: Main 1..Main 4`
				`17: Matrix 1..Matrix 8`
				`18: DCA 1..DCA 8`
				`19: DCA 9..DCA 16`

/$ctl/layer/C/1	N	1..6		Center layer 1 node (see above)
/$ctl/layer/C/1/ofs	I	0..8		Center layer 1 offset
/$ctl/layer/C/1/name	S		10 chars max DCA, MAIN, AUX, BUSES, USER1, USER2	Center layer 1 name
/$ctl/layer/C/1/1	N	1..16		Center layer 1, node 1
/$ctl/layer/C/1/1/type	S		OFF, CH, BUS, DCA, MIDI, SEND, FX	Center layer 1, node 1 type (OSC patterns in italic below correspond to MIDI type)
/$ctl/layer/C/1/1/i	I	1..127		Center layer 1, node 1 index
/$ctl/layer/C/1/1/dst	I	1..16		Left layer 1, node 1 destination index (used for type SEND)
/$ctl/layer/C/1/1/val	*I*	*0..127*		*Center layer 1, node 1 value (when type MIDI)*
/$ctl/layer/R	N			Right layer node

/$ctl/layer/R/sel	I	1..19	1..7 settable
			8..19 fixed/pre-assigned
				Right layer select:
				`1: Main`
				`2: DCA`
				`3: Channels`
				`4: Aux/FX`
				`5: Bus/Master`
				`6: User 1`
				`7: User 2`
				`8: Ch 1..Ch 4`
				`9: Ch 9..Ch 12`
				`10: Ch 17..Ch 20`

				11: Ch 25..Ch 28 12: Ch 33..Ch 36 13: Aux 1..Aux 4 14: Bus 1..Bus 4 15: Bus 9..Bus 12 16: Main 1..Main 4 17: Matrix 1..Matrix 4 18: DCA 1..DCA 4 19: DCA 9..DCA 12
/$ctl/layer/R/1	N	1..7		Right layer 1 node (see above)
/$ctl/layer/R/1/ofs	I	0..15		Right layer 1 offset
/$ctl/layer/R/1/name	S		MAIN, DCA, CH1-40, AUX, BUSES, USER1, USER2	Right layer 1 name
/$ctl/layer/R/1/1	N	1..16 (40 for…/R/3…)		Right layer 1, node 1. 16 nodes except for type CH1-40: 40 nodes
/$ctl/layer/R/1/1/type	S		OFF, CH, BUS, DCA, MIDI, SEND, FX	Right layer 1, node 1 type (OSC patterns in italic below correspond to MIDI type)
/$ctl/layer/R/1/1/i	I	0..127		Right layer 1, node 1 index
/$ctl/layer/R/1/1/dst	I	1..16		Right layer 1, node 1 destination index (used for type SEND)
/$ctl/layer/R/1/1/val	*I*	*0..127*		*Right layer 1, node 1 value (when type MIDI)*
/$ctl/user	N			User node
/$ctl/user/sel	I	1..16		User select[19]
/$ctl/user/mode	S		USER, 2TRK, WLIVE, MGRP, SHOW	User button mode (5 buttons above the wheel)
/$ctl/user/cmode	S		HA, GATE, COMP, FLT, U1, U2, U3, PAN	User channel mode (8 buttons top right corner of the console)
/$ctl/user/gpio	N			User GPIO node
/$ctl/user/gpio/1	N	1..4		User GPIO 1 node
/$ctl/user/gpio/1/bu	N			User GPIO 1 up node
/$ctl/user/gpio/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User GPIO 1 up function (see appendix on buttons)
/$ctl/user/gpio/1/bu/name	S		16 chars max	User GPIO 1 up name (use a leading '|' to invert characters)
/$ctl/user/gpio/1/bu/$fname	S		16 chars max	User GPIO 1 up $fname [RO]
/$ctl/user/user	N			User Layer node (bottom with Link enabled)
/$ctl/user/user/1	N	1..4		User layer button 1 node
/$ctl/user/user/1/bu	N			User layer button 1 upper row node

[19] Setting values using the range 1..16, reported values are in the range 0..15

/$ctl/user/user/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User layer button1 upper row function (see appendix on buttons)
/$ctl/user/user/1/bu/name	S		16 chars max	User layer button 1 upper row name (use a leading '|' to invert characters)
/$ctl/user/user/1/bu/$fname	S		16 chars max	User layer button 1 upper row function name [RO]
/$ctl/user/user/1/bd	N			User layer button 1 lower row node
/$ctl/user/user/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User layer button 1 lower row function (see appendix on buttons)
/$ctl/user/user/1/bd/name	S		16 chars max	User layer button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/user/1/bd/$fname	S		16 chars max	User layer button 1 lower row function name [RO]
/$ctl/user/daw1	N			User DAW1 node
/$ctl/user/daw1/1	N	1..4		User DAW1 button 1 node
/$ctl/user/daw1/1/bu	N			User DAW1 button 1 upper row node
/$ctl/user/daw1/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW1 button 1 upper row function (see appendix on buttons)
/$ctl/user/daw1/1/bu/name	S		16 chars max	User DAW1 button 1 upper row name (use a leading '|' to invert characters)
/$ctl/user/daw1/1/bu/$fname	S		16 chars max	User DAW1 button 1 upper row function name [RO]
/$ctl/user/daw1/1/bd	N			User DAW1 button 1 lower row node
/$ctl/user/daw1/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP,	User DAW1 button 1 lower row function (see appendix on buttons)

			MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	
/$ctl/user/daw1/1/bd/name	S		16 chars max	User DAW1 button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/daw1/1/bd/$fname	S		16 chars max	User DAW1 button 1 lower row function name [RO]
/$ctl/user/daw2	N			User DAW2 node
/$ctl/user/daw2/1	N	1..4		User DAW2 button 1 node
/$ctl/user/daw2/1/bu	N			User DAW2 button 1 upper row node
/$ctl/user/daw2/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW2 button 1 upper row function (see appendix on buttons)
/$ctl/user/daw2/1/bu/name	S		16 chars max	User DAW2 button 1 upper row name (use a leading '|' to invert characters)
/$ctl/user/daw2/1/bu/$fname	S		16 chars max	User DAW2 button 1 upper row function name [RO]
/$ctl/user/daw2/1/bd	N			User DAW2 button 1 lower row node
/$ctl/user/daw2/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW2 button 1 lower row function (see appendix on buttons)
/$ctl/user/daw2/1/bd/name	S		16 chars max	User DAW2 button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/daw2/1/bd/$fname	S		16 chars max	User DAW2 button 1 lower row function name [RO]
/$ctl/user/daw3	N			User DAW3 node
/$ctl/user/daw3/1	N	1..4		User DAW3 button 1 node
/$ctl/user/daw3/1/bu	N			User DAW3 button 1 upper row node
/$ctl/user/daw3/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW3 button 1 upper row function (see appendix on buttons)

/$ctl/user/daw3/1/bu/name	S		16 chars max	User DAW3 button 1 upper row name (use a leading '|' to invert characters)
/$ctl/user/daw3/1/bu/$fname	S		16 chars max	User DAW3 button 1 upper row function name [RO]
/$ctl/user/daw3/1/bd	N			User DAW3 button 1 lower row node
/$ctl/user/daw3/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW3 button 1 lower row function (see appendix on buttons)
/$ctl/user/daw3/1/bd/name	S		16 chars max	User DAW3 button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/daw3/1/bd/$fname	S		16 chars max	User DAW3 button 1 lower row function name [RO]
/$ctl/user/daw4	N			User DAW4 node
/$ctl/user/daw4/1	N			User DAW4 button 1 node
/$ctl/user/daw4/1/bu	N	1..4		User DAW4 button 1 upper row node
/$ctl/user/daw4/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW4 button 1 upper row function (see appendix on buttons)
/$ctl/user/daw4/1/bu/name	S		16 chars max	User DAW4 button 1 upper row name (use a leading '|' to invert characters)
/$ctl/user/daw4/1/bu/$fname	S		16 chars max	User DAW4 button 1 upper row function name [RO]
/$ctl/user/daw4/1/bd	N			User DAW4 button 1 lower row node
/$ctl/user/daw4/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User DAW4 button 1 lower row function (see appendix on buttons)
/$ctl/user/daw4/1/bd/name	S		16 chars max	User DAW4 button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/daw4/1/bd/$fname	S		16 chars max	User DAW4 button 1 lower row function name [RO]
/$ctl/user/1	N	1..16		User 1 node

/$ctl/user/1/1	N	1..4		User 1 button/encoder 1 node
/$ctl/user/1/1/led	I	0..1		User 1 LED 1 off/on switch
/$ctl/user/1/1/col	I	1..12		User 1 LED 1 color
/$ctl/user/1/1/enc	N			User 1 encoder 1 node
/$ctl/user/1/1/enc/mode	S		OFF, FDR, PAN, DCA, SSND, FSND, FX, DAWMCU, MIDICC, SD A, SD B	User 1 encoder 1 function (see appendix on buttons)
/$ctl/user/1/1/enc/name	S		16 chars max	User 1 encoder 1 name (use a leading '|' to invert characters)
/$ctl/user/1/1/enc/$fname	S		16 chars max	User 1 encoder 1 function name [RO]
/$ctl/user/1/1/bu	N			User 1 button 1 upper row node
/$ctl/user/1/1/bu/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User 1 button 1 upper row function (see appendix on buttons)
/$ctl/user/1/1/bu/name	S		16 chars max	User 1 button 1 upper row name (use a leading bu/mode'|' to invert characters)
/$ctl/user/1/1/bu/$fname	S		16 chars max	User 1 button 1 upper row function name [RO]
/$ctl/user/1/1/bd	N			User 1 button 1 lower row node
/$ctl/user/1/1/bd/mode	S		OFF, MUTE, INS1, INS2, MGRP, DCAMUTE, SOF, FXPAR, DAWBTN, DAWENC, CHPAGE, PAGE, OTHER, GPIO, FSTART, MIDICCT, MIDICCP, MIDINT, MIDINP, MIDIPGM, USBPR, SDRECA, SESSIONA, MARKERA, SDRECB, SESSIONB, MARKERB	User 1 button 1 lower row function (see appendix on buttons)
/$ctl/user/1/1/bd/name	S		16 chars max	User 1 button 1 lower row name (use a leading '|' to invert characters)
/$ctl/user/1/1/bd/$fname	S		16 chars max	User 1 button 1 lower row function name [RO]
/$ctl/user/cuser	N	1..3		Cuser node
/$ctl/user/cuser/1	S		1, 2, 3, … , 15, 16	Cuser 1 rotary knob position; Keeps the bus send value assigned to respective channel for the F1..F3 section. Can be set/changed by holding the F1..F3 key and turning the knob.
/$ctl/gpio	N			GPIO node
/$ctl/gpio/1	N	1..4		GPIO 1 node
/$ctl/gpio/1/mode	S		TGLNO, TGLNC, INNO, INNC, OUTNO, OUTNC	GPIO 1 mode (TGL: toggle; NO: normally open; NC: normally closed)
/$ctl/gpio/1/$state	I	0..1		GPIO 1 state [RO]

/$ctl/gpio/1/gpstate	I	0..1		GPIO 1 gpio state
/$ctl/safes	N			Global Safes node
/$ctl/safes/ch	S		40 chars max	Ch safes switches (+ or space)
/$ctl/safes/aux	S		8 chars max	Aux safes switches (+ or space)
/$ctl/safes/bus	S		16 chars max	Bus safes switches (+ or space)
/$ctl/safes/main	S		4 chars max	Main safes switches (+ or space)
/$ctl/safes/mtx	S		8 chars max	Matrix safes switches (+ or space)
/$ctl/safes/dca	S		16 chars max	DCA safes switches (+ or space)
/$ctl/safes/mute	S		8 chars max	Mute safes switches (+ or space)
/$ctl/safes/fx	S		16 chars max	FX safes switches (+ or space)
/$ctl/safes/source	N			Source Safes node
/$ctl/safes/source/LCL	S		8 chars max	LCL source safes switches (+ or space)
/$ctl/safes/source/AUX	S		8 chars max	AUX source safes switches (+ or space)
/$ctl/safes/source/A	S		48 chars max	A source safes switches (+ or space)
/$ctl/safes/source/B	S		48 chars max	B source safes switches (+ or space)
/$ctl/safes/source/C	S		48 chars max	C source safes switches (+ or space)
/$ctl/safes/source/SC	S		32 chars max	SC source safes switches (+ or space)
/$ctl/safes/source/USB	S		48 chars max	USB source safes switches (+ or space)
/$ctl/safes/source/CRD	S		64 chars max	CRD source safes switches (+ or space)
/$ctl/safes/source/MOD	S		64 chars max	MOD source safes switches (+ or space)
/$ctl/safes/source/PLAY	S		4 chars max	REC source safes switches (+ or space)
/$ctl/safes/source/AES	S		2 chars max	AES source safes switches (+ or space)
/$ctl/safes/source/USR	S		24 chars max	USR source safes switches (+ or space)
/$ctl/safes/source/OSC	S		2 chars max	Osc source safes switches (+ or space)
/$ctl/safes/output	N			Output Safes node
/$ctl/safes/output/LCL	S		8 chars max	LCL out safes switches (+ or space)
/$ctl/safes/output/AUX	S		8 chars max	AUX out safes switches (+ or space)
/$ctl/safes/outpu/A	S		48 chars max	A out safes switches (+ or space)
/$ctl/safes/output/B	S		48 chars max	B out safes switches (+ or space)
/$ctl/safes/output/C	S		48 chars max	C out safes switches (+ or space)
/$ctl/safes/output/SC	S		32 chars max	SC out safes switches (+ or space)
/$ctl/safes/output/USB	S		48 chars max	USB out safes switches (+ or space)
/$ctl/safes/output/CRD	S		64 chars max	CRD out safes switches (+ or space)
/$ctl/safes/output/MOD	S		64 chars max	MOD out safes switches (+ or space)
/$ctl/safes/output/REC	S		4 chars max	REC out safes switches (+ or space)
/$ctl/safes/output/AES	S		2 chars max	AES out safes switches (+ or space)
/$ctl/safes/area	N			Area safes node
/$ctl/safes/area/L	S		7 chars max	Left area safes switches (+ or space)
/$ctl/safes/area/C	S		6 chars max	Center area safes switches (+ or space)
/$ctl/safes /area/R	S		7 chars max	Right area safes switches (+ or space)
/$ctl/safes/custom	S		22 chars max	Custom area safes switches (+ or space)
/$ctl/safes/setup	S		3 chars max	Setup area safes switches (+ or space)
/$ctl/daw	N			DAW node
/$ctl/daw/on	I	0..1		DAW enable
/$ctl/daw/conn	S		DIN, USB	DAW connection
/$ctl/daw/emul	S		MCU, HUI	DAW emulation
/$ctl/daw/config	S		CC, MSTR, MSTR1EXT, MSTR2EXT	DAW configuration
/$ctl/daw/ccup	I	0..1		DAW use upper cc

/$ctl/daw/disjog	I	0..1		DAW disable wheel during play
/$ctl/daw/preset	S		-, cubase, live, logicx, nuendo, protools, reaper, studioone	DAW last loaded preset
/$ctl/daw /$on	I	0..1		DAW enable switch
/$ctl/daw/$bpage	I	0..4		DAW on button page
/$ctl/daw/$btntouch	I	0..1		DAW on button sel fader touch
/$ctl/daw/$btnvpot	I	0..1		DAW on button sel vpot
/$ctl/daw/$btnrecrdy	I	0..1		DAW on button sel record ready
/$ctl/daw/$btnauto	I	0..1		DAW on button sel auto
/$ctl/daw/$btnvsel	I	0..1		DAW on button sel v-sel
/$ctl/daw/$btninsert	I	0..1		DAW on button sel insert
/$ctl/midi	N			Midi node
/$ctl/midi/enchctl	S		OFF, DIN, USB	Midi channel control
/$ctl/midi/enfxctl	S		OFF, DIN, USB	Midi FX control
/$ctl/midi/encustctl	S		OFF, DIN, USB	Midi custom control
/$ctl/midi/ensysex	S		OFF, DIN, USB	Midi SYSEX control
/$ctl/midi/enmidicc	S		OFF, DIN, USB	External Midi control
/$ctl/OSC	N			OSC node
/$ctl/OSC/ronly	I	0..1		Console OSC read only switch
/$ctl/lib	N			Scene/Snap/Show node
/$ctl/lib/$scenes	S		Ex: `scene_1, scene_2, scene_3`	List of scene/snap names [RO] [20] in the currently opened show
/$ctl/lib/$actidx	I	0..n		Scene/snap number currently loaded/active [RO][21]
/$ctl/lib/$active	S		256 chars max	Name of the active scene/snap [RO], ex: `I:SHOW2/scene_1.snap`
/$ctl/lib/$actshow	S		256 chars max	Name of the active show [RO], after having pressed on the "OPEN SHOW" icon, ex: `I:SHOW2`
/$ctl/lib/$action	S		IDLE, GOPREV, GONEXT, GO, PREV, NEXT	Show control actions, after having pressed on the "OPEN SHOW" icon
/$ctl/lib/$actionidx	I	1..n		Scene/snap index user selection, in the list of scenes/snaps. A show must be opened. Use `/$ctl/lib/$action ,s GO` to load the scene/snap `$actionidx` points to
/$ctl/$globals	N			CTL Global Settings node
/$ctl/$globals/fdrsel	I	0..1		Screen Touch Fader Select
/$ctl/$globals/fdrres	S		NORM, FINE, AUTO	Fader resolution
/$ctl/$globals/fdrspd	S		SLOW, MED, FAST	Fader speed
/$ctl/$globals/mousetchdis	I	0..1		Mouse disable touch
/$ctl/$globals/mousespd	F	0.1..2.0	191 steps	Mouse speed
/$ctl/$globals /tapflash	S		OFF, 8X, ON	Tap Tempo Flash
/$ctl/$globals /srcdisp	I	0..1		Show source on scribble
/$ctl/$globals /lockmtr	I	0..1		Show meter page when locked
/$ctl/$globals /_cf_load	I	0..1		Confirm Snapshot Load

[20] Only the first name of the list is returned by std OSC command. You must use the node definition command (OSC or native interface) to get the full contents, for ex: `/$ctl/lib~~~,s~~?~~~.` A show must be opened for the command to be active

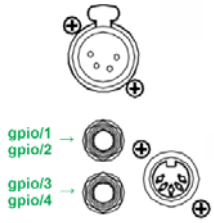
[21] A show must be opened for the command to be active. 0 means "no show scene/snap loaded"

/$ctl/$globals /_cf_upd	I	0..1		Confirm Snapshot Update
/$ctl/$globals /$filesort	S		A→Z, Z→A, 0→9, 9→0	File sort order
/$ctl/$globals /$noautosave	I	0..1		Auto save switch (0=autosave)
/$ctl/$globals /$savenow	I	0..1		Save console data now

Appendices

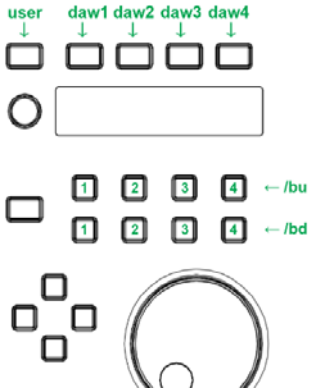
Appendix: Buttons (user/gpio, user/user, user/daw, user/)

WING includes a rather large set of buttons separated in different logical blocks: user/gpio, user/user, and user/daw and user. They are all managed under the `$ctl` subtree of commands. As in the case of effects where the effect model sets the type and number of OSC patterns available for supporting the functionality currently in effect, the associated JSON structure varies and adapts to the necessary sets of parameters.



user/gpio

This subsection covers the 4 possible GPIOs supported by WING; the actual set of usable OSC patterns available at a given time depends on the `mode` parameter value of the `/$ctl/user/gpio/1..4/bu/` OSC pattern represented below as `<OSC pattern>`.

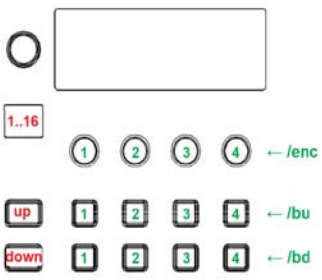


user/user

This subsection covers the 8 user buttons supported by WING; the actual set of usable OSC patterns available at a given time depends on the `mode` parameter value of the `/$ctl/user/user/1..4/bu/` and the `/$ctl/user/user/1..4/bd/` OSC patterns for the 4 buttons of the upper and lower row of the button section, and represented below as `<OSC pattern>`.

user/daw1..4

This subsection covers the 4 possible sets of 8 DAW buttons supported by WING; the actual set of usable OSC patterns available at a given time depends on the `mode` parameter value of the `/$ctl/user/daw1..daw4/1..4/bu/` and the `/$ctl/user/daw1..daw4/1..4/bd/` OSC patterns for the 4 buttons of the upper and lower row of the button section, and represented below as `<OSC pattern>`.



user/1..16/..4

This subsection covers the 16 possible sets of 8 user buttons and 4 user encoders supported by WING; the actual set of usable OSC patterns available at a given time depends on the `mode` parameter value of the `/$ctl/user/1..16/1..4/bu/`, `/$ctl/user/1..16/1..4/bd/`, and `/$ctl/user/1..16/1..4/enc/` OSC patterns for the 4 buttons of the upper and lower row of the button section, and the 4 encoders represented below as `<OSC pattern>` and `<OSC enc pattern>`, respectively.

The tables below list the different options for OSC patterns `<OSC pattern>`:

mode	Command	Type	Range / Text	Description
OFF	none			OFF
MUTE	<OSC pattern>/ch	I	1..76	Channel number
INS1	<OSC pattern>/ch	I	1..76	Channel number
INS2	<OSC pattern>/ch	I	1..76	Channel number
MGRP	<OSC pattern>/mgrp	S	MGRP.1, MGRP.2..MGRP.8	Mute group number
DCAMUTE	<OSC pattern>/dca	S	DCA.1, DCA.2..DCA.16	DCA fader number mute
SOF	<OSC pattern>/ch	I	1..76	Channel number
FXPAR	<OSC pattern>/fx	S	FX1..FX16	FX processor number
	<OSC pattern>/par	I	1..32	FX processor parameter number
DAWBTN	<OSC pattern>/btn	S	T1..T20, N1..N9, A1..A16, F1..F8, V1..V15, AU1..AU12, SY1..SY10, OT1..OT12, E1..E10, SP1.., SP6	MCU button[22]
DAWENC	<OSC pattern>/enc	S	M1P..M8P, E1P..E16P, M1..M8, E1..E16, JOG	DAW Rotary[23]
CHPAGE	<OSC pattern>/ch	I	1..76	Channel number
	<OSC pattern>/pg	S	HOME, INPUT, FILT, GATE, DYN, EQ, INS1, INS2, MAIN, SEND, SND.CFG, SND.EQ	Page name
PAGE	<OSC pattern>/pg	S	FX, MTRS, CHINS, SRC, OUTS, SETUP, LIB, CUSTCTL, MON, 2TRK, WLIVE, MIXV, FDRV, SENDV, MGRP, LAYER	Page name
OTHER	<OSC pattern>/other	S	TBA, TBB, ALTSRC, DAWSW, MONA, MONB, MONAB, MONDIM, MONMONO, MONSWAP, FDROFF, FDR0DB, AUTOX, AUTOY	Other functions
GPIO	<OSC pattern>/GPIO	S	A, B, C, D, A-P, B-P, C-P, D-P,	GPIO Toggle or Push
FSTART	<OSC pattern>/ch	I	1..76	Channel number
MIDICCT	<OSC pattern>/ch	I	1..16	MIDI channel (toggle)
	<OSC pattern>/cc	I	0..127	MIDI control change number
	<OSC pattern>/val	I	0..127	MIDI control value

[22] See MCU [DAW BUTTONS] commands list in Appendixes
[23] See MCU [DAW V-POTS] commands list in Appendixes

MIDICCP	<OSC pattern>/ch	I	1..16	MIDI channel (push)
	<OSC pattern>/cc	I	0..127	MIDI control change number
	<OSC pattern>/val	I	0..127	MIDI control value
MIDINT	<OSC pattern>/ch	I	1..16	MIDI channel (toggle)
	<OSC pattern>/note	I	0..127	MIDI note
	<OSC pattern>/val	I	0..127	MIDI note value
MIDINP	<OSC pattern>/ch	I	1..16	MIDI channel (push)
	<OSC pattern>/note	I	0..127	MIDI note
	<OSC pattern>/val	I	0..127	MIDI note value
MIDIPGM	<OSC pattern>/ch	I	1..16	MIDI channel
	<OSC pattern>/note	I	1..128	MIDI program value
USBPR	<OSC pattern>/usbpr	S	PSTOP, PLAY, PPAUSE, PNEXT, PPREV, RSTOP, RECORD, RPAUSE, RNEW	USB Play Rec
SDRECA	<OSC pattern>/sdrec	S	STOP, PLAY, REC, PAUSE, PLAYSTOP, PLAYPAUSE, ADD, PREV, NEXT, PLAYMARKER, GOMARKER, SELSESSION, PREV_S, NEXT_S	SD A recorder
SESSIONA	<OSC pattern>/session	S	S1..S20	SD A Session
MARKERA	<OSC pattern>/marker	S	M1..M20	SD A Marker
SDRECB	<OSC pattern>/sdrec	S	STOP, PLAY, REC, PAUSE, PLAYSTOP, PLAYPAUSE, ADD, PREV, NEXT, PLAYMARKER, GOMARKER, SELSESSION, PREV_S, NEXT_S	SD B recorder
SESSIONB	<OSC pattern>/session	S	S1..S20	SD B Session
MARKERB	<OSC pattern>/marker	S	M1..M20	SD B Marker

The table below lists the different options for the OSC encoder pattern `<OSC enc pattern>`:

mode	Command	Type	Range / Text	Description
OFF	none			OFF
FDR	<OSC enc pattern>/ch	I	1..76	Channel number
PAN	<OSC enc pattern>/ch	I	1..76	Channel number
DCA	<OSC enc pattern>/dca	S	DCA.1, DCA.2..DCA.16	DCA fader number
SSND	<OSC enc pattern>/send	S	BUS1, …, BUS16	Send to Bus number

FSND	<OSC enc pattern>/ch <OSC enc pattern>/send	I S	1..48 BUS1, …, BUS16	Channel number Send to Bus number
FX	<OSC enc pattern>/fx	S	FX1..FX16	FX processor number
	<OSC enc pattern>/par	I	1..40	FX processor parameter number
DAWMCU	<OSC enc pattern>/mcuenc	S	M1..M8, E1..E16, JOG	DAW Rotary[24]
MIDICC	<OSC enc pattern>/ch	I	1..16	MIDI channel
	<OSC enc pattern>/cc	I	0..127	MIDI control change number
	<OSC enc pattern>/val	I	0..127	MIDI control change value
SD A	<OSC enc pattern>/sdarec	S	POS, MARKER, SESSION	SD-A Recorder
SD B	<OSC enc pattern>/sdbrec	S	POS, MARKER, SESSION	SD-B Recorder

[24] See MCU [DAW REMOTE MCU] commands list in Appendixes

Appendix: Effects and Plugins' Parameters list

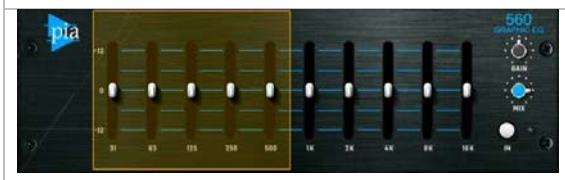
In the (long) tables below, we list all known/exposed effects and plugins available with the WING digital console, along with their name, type, and min/max/step/list values; We therefore present Standard Effects, Premium effects, Filter Plugins, Gate Plugins, EQ Plugins, and Compressor Plugins. All active effects and plugins modify the JSON tree and their respective OSC patterns. The tables below show all parameters associated to effects and plugins, including their name, type, and value range following the OSC pattern `/fx/1..16/`

On any channel, an insert will create a processing delay of 32 samples (i.e. around 0.66ms). This is because audio is routed to a different DSP for FX processing. It is important to take this into account when mixing, as phasing effects may result from the imposed delay.

Effects

Standard effects

	None *0 **"mdl"**: NONE*
	External *0 **"mdl"**: EXT* *1 **"egrp"**: str [OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES] ext grp* *2 **"ein"**: int [1…64] ext in* *3 **"emode"**: str [M, ST, M/S] ext mode* *4 **"lat"**: int [0…200] latency* *5 **"trim"**: linf [-18, 18, 361] dB, trim*
	Graphic EQ *0 **"mdl"**: GEQ* *1 **"type"**: str [STD, TRU] geq type* *2 **"20"**: linf [-15, 15, 121] dB* *3 **"25"**: linf [-15, 15, 121] dB* *4 **"31"**: linf [-15, 15, 121] dB* *5 **"40"**: linf [-15, 15, 121] dB* *6 **"50"**: linf [-15, 15, 121] dB* *7 **"63"**: linf [-15, 15, 121] dB* *8 **"80"**: linf [-15, 15, 121] dB* *9 **"100"**: linf [-15, 15, 121] dB* *10 **"125"**: linf [-15, 15, 121] dB* *11 **"160"**: linf [-15, 15, 121] dB* *12 **"200"**: linf [-15, 15, 121] dB* *13 **"250"**: linf [-15, 15, 121] dB* *14 **"315"**: linf [-15, 15, 121] dB* *15 **"400"**: linf [-15, 15, 121] dB* *16 **"500"**: linf [-15, 15, 121] dB* *17 **"630"**: linf [-15, 15, 121] dB* *18 **"800"**: linf [-15, 15, 121] dB* *19 **"1k"**: linf [-15, 15, 121] dB* *20 **"1k25"**: linf [-15, 15, 121] dB* *21 **"1k6"**: linf [-15, 15, 121] dB* *22 **"2k"**: linf [-15, 15, 121] dB* *23 **"2k5"**: linf [-15, 15, 121] dB* *24 **"3k15"**: linf [-15, 15, 121] dB* *25 **"4k"**: linf [-15, 15, 121] dB* *26 **"5k"**: linf [-15, 15, 121] dB* *27 **"6k3"**: linf [-15, 15, 121] dB* *28 **"8k"**: linf [-15, 15, 121] dB* *29 **"10k"**: linf [-15, 15, 121] dB*

	*30 **"12k5"**: linf [-15, 15, 121] dB* *31 **"16k"**: linf [-15, 15, 121] dB* *32 **"20k"**: linf [-15, 15, 121] dB*
	PIA 560 GEQ *0 **"mdl"**: PIA* *1 **"mix"**: linf [0, 125, 126] %, mix* *2 **"gain"**: linf [-12, 12, 241] dB* *3 **"31"**: linf [-12, 12, 241] dB* *4 **"63"**: linf [-12, 12, 241] dB* *5 **"125"**: linf [-12, 12, 241] dB* *6 **"250"**: linf [-12, 12, 241] dB* *7 **"500"**: linf [-12, 12, 241] dB* *8 **"1k"**: linf [-12, 12, 241] dB* *9 **"2k"**: linf [-12, 12, 241] dB* *10 **"4k"**: linf [-12, 12, 241] dB* *11 **"8k"**: linf [-12, 12, 241] dB* *12 **"16k"**: linf [-12, 12, 241] dB*
	Triple Dynamic EQ *0 **"mdl"**: DEQ3* *1 **"1-thr"**: linf [-60, 0, 121] dB, threshold 1* *2 **"1-ratio"**: str [1.20, 1.30, 1.50, 2.00, 3.00, 5.00, 10.00] ms, ratio 1* *3 **"1-att"**: linf [0.00, 200.00, 201] ms, att 1* *4 **"1-rel"**: logf [20.00, 4000.00, 130] ms, rel 1* *5 **"1-filt"**: str [OFF, BP, LP6, LP12, HP6, HP12]* *6 **"1-g"**: linf [-15.00, 15.00, 301] dB, gain 1* *7 **"1-f"**: logf [20, 20000, 961] Hz, freq 1* *8 **"1-q"**: logf [0.44, 10.00, 181] qual 1* *9 **"1-mode"**: str [low, high] mode 1* *10 **"2-thr"**: linf [-60, 0, 121] dB, threshold 2* *11 **"2-ratio"**: str [1.20, 1.30, 1.50, 2.00, 3.00, 5.00, 10.00] ms, ratio 2* *12 **"2-att"**: linf [0.00, 200.00, 201] ms, att 2* *13 **"2-rel"**: logf [20.00, 4000.00, 130] ms, rel 2* *14 **"2-filt"**: str [OFF, BP, LP6, LP12, HP6, HP12]* *15 **"2-g"**: linf [-15.00, 15.00, 301] dB, gain 2* *16 **"2-f"**: logf [20, 20000, 961] Hz, freq 2* *17 **"2-q"**: logf [0.44, 10.00, 181] qual 2* *18 **"2-mode"**: str [low, high] mode 2* *19 **"3-thr"**: linf [-60, 0, 121] dB, threshold 3* *20 **"3-ratio"**: str [1.20, 1.30, 1.50, 2.00, 3.00, 5.00, 10.00] ms, ratio 3* *21 **"3-att"**: linf [0.00, 200.00, 201] ms, att 3* *22 **"3-rel"**: logf [20.00, 4000.00, 130] ms, rel 3* *23 **"3-filt"**: str [OFF, BP, LP6, LP12, HP6, HP12]* *24 **"3-g"**: linf [-15.00, 15.00, 301] dB, gain 3* *25 **"3-f"**: logf [20, 20000, 961] Hz, freq 3* *26 **"3-q"**: logf [0.44, 10.00, 181] qual 3* *27 **"3-mode"**: str [low, high] mode 3*
	Combinator *0 **"mdl"**: C5-CMB* *1 **"thr"**: linf [-40, 0, 401] dB, threshold* *2 **"gain"**: linf [-10, 10, 201] dB, gain* *3 **"ratio"**: str [1.1, 1.2, 1.3, 1.5, 1.7, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0, 7.0, 10.0, 100.0] ms, ratio* *4 **"slope"**: str [24, 48] dB/Oct, slope* *5 **"bandse l"**: int [1..5] selected band* *6 **"att"**: linf [0, 20, 21] attack* *7 **"rel"**: logf [20, 3000, 201] ms, release* *8 **"arel"**: int [0, 1] auto release* *9 **"sbc"**: linf [1, 10, 10] sbc speed* *10 **"sbcon"**: int [0, 1] sbc on* *11 **"thr_1"**: linf [-10, 10, 201] dB, 1-THR* *12 **"thr_2"**: linf [-10, 10, 201] dB, 2-THR* *13 **"thr_3"**: linf [-10, 10, 201] dB, 3-THR* *14 **"thr_4"**: linf [-10, 10, 201] dB, 4-THR* *15 **"thr_5"**: linf [-10, 10, 201] dB, 5-THR*

	16 **"gain_1"**: linf [-10, 10, 201] dB, 1-GAIN 17 **"gain_2"**: linf [-10, 10, 201] dB, 2-GAIN 18 **"gain_3"**: linf [-10, 10, 201] dB, 3-GAIN 19 **"gain_4"**: linf [-10, 10, 201] dB, 4-GAIN 20 **"gain_5"**: linf [-10, 10, 201] dB, 5-GAIN 21 **"byp_1"**: int [0, 1], 1-BYP 22 **"byp_2"**: int [0, 1], 2-BYP 23 **"byp_3"**: int [0, 1], 3-BYP 24 **"byp_4"**: int [0, 1], 4-BYP 25 **"byp_5"**: int [0, 1], 5-BYP 26 **"width_1"**: linf [-50, 50, 101], 1-XOVER 27 **"width_2"**: linf [-50, 50, 101], 2-XOVER 28 **"width_3"**: linf [-50, 50, 101], 3-XOVER 29 **"width_4"**: linf [-50, 50, 101], 4-XOVER 30 **"width_5"**: linf [-50, 50, 101], 5-XOVER 31 **"mix"**: linf [0, 100, 101], mix 32 **"$bdsolo"**: int [0, 1] band solo
	Precision Limiter 0 **"mdl"**: LIMITER 1 **"gin"**: linf [0, 18, 73] dB, in gain 2 **"gout"**: linf [-18, 0, 73] dB out gain 3 **"sqz"**: int [0…100] sqeeze 4 **"knee"**: int [0…10] knee 5 **"again"**: int [0, 1] auto gain 6 **"att"**: linf [.05, 1, 95] ms, attack 7 **"rel"**: logf [20, 2000, 101] ms, release
	Speaker Manager 0 **"mdl"**: SPKMAN 1 **"hpf"**: logf [20.00, 20000.00, 961] Hz, high 2 **"hptype"**: str [FLAT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48] type 3 **"lpf"**: logf [20.00, 20000.00, 961] Hz, low 4 **"lptype"**: str [FLAT, BW6, BW12, BS12, LR12, BW18, BW24, BS24, LR24, BW48, LR48] type 5 **"tiltf"**: logf [100.00, 10000.00, 121] Hz, tilt 6 **"tiltg"**: linf [-6.00, 6.00, 121] dB, tilt gain 7 **"phase"**: linf [0.00, 180.00, 37] phase 8 **"invert "**: int [0, 1] invert 9 **"dist"**: linf [0.00, 5.00, 501] mtrs, distance 10 **"pos"**: linf [-5.00, 5.00, 1001] mtrs, pos. 11 **"dyneq"**: int [0, 1] deq 12 **"dynthr"**: linf [-60.00, 0.00, 121] dB, threshold 13 **"deqratio"**: str [1.20, 1.30, 1.50, 2.00, 3.00, 5.00, 10.00] ratio 14 **"deqatt"**: linf [0.00, 200.00, 201] ms, attack 15 **"deqrel"**: logf [20.00, 4000.00, 130] ms, release 16 **"deqfilt"**: str [OFF, BP, LP6, LP12, HP6, HP12] 17 **"deqg"**: linf [-15.00, 15.00, 301] dB, gain 18 **"deqf"**: logf [20.00, 20000.00, 961] Hz, freq 19 **"deqq"**: logf [0.44, 10.00, 181] qual 20 **"deqmode"**: str [Low, high] mode 21 **"lim"**: int [0, 1] limiter 22 **"limthr"**: linf [-24.00, 0.00, 241] dB, threshold 23 **"limrms"**: int [0, 1] rms 24 **"limrel"**: logf [50.00, 2000.00, 121] ms, release
	2-Band DeEsser 0 **"mdl"**: DE-S2 1 **"lo"**: linf [0, 50, 51] low 2 **"hi"**: linf [0, 50, 51] high 3 **"los"**: linf [0, 50, 51] low (s) 4 **"his"**: linf [0, 50, 51] high (s) 5 **"gdr"**: str [FEMALE, MALE] gender 6 **"mode"**: str [STEREO, MID/SIDE] mode

Ultra Enhancer

```
 0 "mdl":   ENHANCE
 1 "stlv":  linf [-100, 100, 201] %, st lvl
 2 "lmf":   linf [-100, 100, 201] %, lmf spread
 3 "lmvl":  linf [-100, 100, 201] %, mono lvl
 4 "st":    linf [-100, 100, 201] %, st pan
 5 "m":     linf [-100, 100, 201] %, mono pan
 6 "bass":  linf [0, 100, 101] %, bass gain
 7 "mid":   linf [0, 100, 101] %, mid gain
 8 "high":  linf [0, 100, 101] %, high gain
 9 "g":     linf [-112, 12, 241] dB, gain
10 "solo":  int [0, 1] solo
11 "bassf": linf [1, 50, 50] bass freq
12 "midq":  linf [1, 50, 50] mid Q
13 "highf": linf [1, 50, 50] high freq
```

Exciter

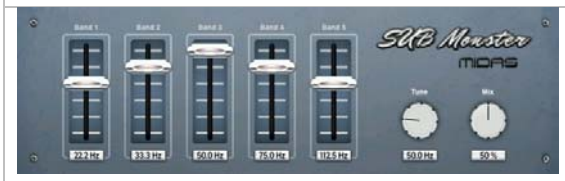
```
 0 "mdl":   EXCITER
 1 "tune":  logf [1000, 10000, 51] Hz, tune
 2 "peak":  linf [0, 100, 101] %, peak
 3 "zfill": linf [0, 100, 101] %, zfill
 4 "timbre":linf [-50, 50, 101] timbre
 5 "harm":  linf [0, 100, 101] %, harm
 6 "mix":   linf [0, 100, 101] %, mix
 7 "dry":   int [0, 1] dry
```

Psycho Bass

```
 0 "mdl":  P-BASS
 1 "int":  linf [-24, 6, 61] dB, intensity
 2 "bass": linf [-60, 0, 121] dB, bass gain
 3 "xf":   logf [32, 200. 51] Hz, X/O freq
 4 "solo": int [0, 1] solo
```

Sub Octaver

```
 0 "mdl":   SUB
 1 "rng":   str [LOW, MID, HIGH] range
 2 "oct1":  linf [0, 100, 101] %, octave 1
 3 "oct2":  linf [0, 100, 101] %, octave 2
```



Sub Monster

```
 0 "mdl":   SUB-M
 1 "mix":   [0, 100, 101] %mix
 2 "freq":  linf [45, 67.5, 226] Hz, freq
 3 "bd1":   linf [0, 100, 101] %, band 1
 4 "bd2":   linf [0, 100, 101] %, band 2
 5 "bd3":   linf [0, 100, 101] %, band 3
 6 "bd4":   linf [0, 100, 101] %, band 4
 7 "bd5":   linf [0, 100, 101] %, band 5
```

Velvet Imager

```
 0 "mdl":   V_IMG
 1 "wid":   linf> [-1.00, 1.00, 201] width
 2 "st":    linf [0.00, 100.00, 101] %, stereo
 3 "gain":  linf [-6.00, 6.00, 49] dB, gain
 4 "mode":  str [K, VELVET] mode
 5 "deep":  int [0,1] deep
```

Double Vocal

```
 0 "mdl":  DOUBLE
 1 "mode": str [TIGHT, LOOSE, GROUP,
                DETUNE, THICK] mode
 2 "mix":  linf [0, 100, 101] %, mix
 3 "sprd": linf [0, 100, 101] %, spread
```

Pitch Fix

```
 0 "mdl":  PCORR
 1 "spd":  linf [1, 100, 100] speed
 2 "amnt": linf [0, 50, 51] amount
 3 "a4":   linf [410, 470, 601] A4 pitch
 4 "_c":   int [0, 1]
 5 "_db":  int [0, 1]
 6 "_d":   int [0, 1]
 7 "_eb":  int [0, 1]
 8 "_e":   int [0, 1]
 9 "_f":   int [0, 1]
10 "_gb":  int [0, 1]
11 "_g":   int [0, 1]
12 "_ab":  int [0, 1]
13 "_a":   int [0, 1]
14 "_bb":  int [0, 1]
15 "_b":   int [0, 1]
```

Rotary Speaker

```
 0 "mdl":  ROTARY
 1 "sw":   str [STOP, SLOW, FAST]
 2 "lo":   logf [.1, 3.999, 51] Hz, lo speed
 3 "hi":   logf [4, 10, 51] Hz, hi speed
 4 "bal":  linf [-100, 100, 201] balance
 5 "mix":  linf [0, 100, 101] %, mix
 6 "dist": linf [0, 100, 101] distance
 7 "dac":  linf [0, 100, 101] %, drum accel
 8 "hac":  linf [0, 100, 101] %, horn accel
```

Phaser

```
 0 "mdl":    PHASER
 1 "spd":    logf [.05, 5, 201] Hz, speed
 2 "phase":  int [0…180] phase
 3 "wave":   int [-50…50] wave
 4 "range":  int [2…98] %, range
 5 "depth":  int [0…100] %, depth
 6 "emod":   int [-100, 100] % env mod
 7 "att":    logf [10, 1000, 201] ms, attack
 8 "hld":    logf [10, 2000, 201] ms, hold
 9 "rel":    logf [10, 1000, 201] ms, release
10 "mix":    int [0…100] %, mix
11 "stg":    int [2…12] stages
12 "reso":   int [0…80] %, reso
```

Tremolo Panner

```
 0 "mdl":    PANNER
 1 "att":    logf [10, 1000, 201] ms, attack
 2 "hld":    logf [10, 2000, 201] ms, hold
 3 "rel":    logf [10, 1000, 201] ms, release
 4 "espd":   int [0…100] %, env>depth
 5 "edep":   int [0…100] %, env>depth
 6 "spd":    logf [.05, 5, 201] Hz, speed
 7 "phase":  int [0…180] phase
 8 "wave":   int [-50…50] wave
 9 "depth":  int [0…100] %, depth
```

Tape Machine

```
 0 "mdl":    TAPE
 1 "drv":    linf [-12, 12, 97] dB, drive
 2 "spd":    logf [7.5, 30, 65]
 3 "low":    int [0, 1] low bump
 4 "hi":     int [0, 1] high shelv
 5 "out":    linf [-12, 12, 97] dB, out gains s
```

Mood Filter

```
 0 "mdl":    MOOD
 1 "fbase":  logf [20, 15000, 101] Hz, base
 2 "filt":   str [LP, HP, BP, NOTCH] type
 3 "slope":  str [12, 24] slope
 4 "reso":   linf [0, 10, 101] reso
 5 "drv":    linf [0, 10, 101] drive
 6 "env":    linf [-100, 100, 201] %, env
 7 "att":    logf [10, 250, 101] ms, attack
 8 "hld":    logf [1, 500, 101] ms, hold
 0 "rel":    logf [1, 500, 101] ms, release
 1 "mix":    linf [0, 10, 101] %, mix
 2 "lfo":    linf [linf [0, 10, 101] %, lfo
 6 "spd":    logf [.05, 20, 301] Hz, speed
 7 "phase":  int [0…180] phase
 8 "wave":   str [TRI, SIN, SAW+, SAW-,
                  RMP, SQU, RND] lfo wave
```

Bodyrez

```
 0 "mdl":    BODY
 1 "body":   linf [0, 100, 101] body
```

Rack Amp

```
 0 "mdl":    RACKAMP
 1 "pre":    linf [0, 10, 101] preamp
 2 "buzz":   linf [0, 10, 101] buzz
 3 "punch":  linf [0, 10, 101] punch
 4 "crunch": linf [0, 10, 101] crunch
 5 "drive    linf [0, 10, 101] drive
 6 "out":    linf [0, 10, 101] out gain
 7 "leq":    linf [0, 10, 101] low eq
 8 "heq"     linf [0, 10, 101] high eq
 9 "cab":    int [0, 1] cab sim
```

UK Rock Amp

```
 0 "mdl":    UKROCK
 1 "gain":   linf [0, 10, 101] gains
 2 "bass":   linf [0, 10, 101] bass
 3 "mid":    linf [0, 10, 101] middle
 4 "treb":   linf [0, 10, 101] trebble
 5 "pres     linf [0, 10, 101] presence
 6 "mstr":   linf [0, 10, 101] master
 7 "out":    linf [0, 10, 101] out gain
 8 "sag"     linf [0, 10, 101] sag
 9 "cab":    int [0, 1] cab sim
```

Angel Amp

```
 0 "mdl":    ANGEL
 1 "gain":   linf [0, 10, 101] gains
 2 "bass":   linf [0, 10, 101] bass
 3 "mid":    linf [0, 10, 101] middle
 4 "treb":   linf [0, 10, 101] trebble
 5 "pres     linf [0, 10, 101] presence
 6 "mstr":   linf [0, 10, 101] master
 7 "out":    linf [0, 10, 101] out gain
 8 "sag"     linf [0, 10, 101] sag
 9 "cab":    int [0, 1] cab sim
10 "midb":   int [0, 1] mid boost
11 "bri":    int [0, 1] bright
12 "bt":     int [0, 1] bottom
```

Jazz Clean Amp

```
 0 "mdl":   JAZZC
 1 "vol":   linf [0, 10, 101] volume
 2 "bass":  linf [0, 10, 101] bass
 3 "mid":   linf [0, 10, 101] middle
 4 "treb":  linf [0, 10, 101] trebble
 5 "out":   linf [0, 10, 101] out gain
 6 "bri":   int [0, 1] bright
 7 "cab":   int [0, 1] cab sim
```

Deluxe Amp

```
 0 "mdl":   DELUXE
 1 "vol":   linf [1, 10, 91] volume
 2 "bass":  linf [1, 10, 91] bass
 4 "treb":  linf [1, 10, 91] trebble
 5 "out":   linf [1, 10, 91] out gain
 6 "sag":   linf [1, 10, 91] sag
 7 "cab":   int [0, 1] cab sim
```

Soul Analogue

```
 0 "mdl":   SOUL
 1 "mix":   linf [0, 125, 126] %, mix
 2 "lf":    linf [0, 10, 101] lo freq
 3 "lg":    linf [-5, 5, 101] lo gain
 4 "lmf":   linf [0, 10, 101] lm freq
 5 "lmf3":  int [0, 1] lm /3
 6 "lmq":   linf [0, 10, 101] lm q
 7 "lmg":   linf [-5, 5, 101] lm gain
 8 "hmf":   linf [0, 10, 101] hm freq
 9 "hmf3":  int [0, 1] hm x3
10 "hmq":   linf [0, 10, 101] hm q
11 "hmg":   linf [-5, 5, 101] hm gain
12 "hf":    linf [0, 10, 101] hf freq
13 "hg":    linf [-5, 5, 101] hf gain
```

Even 88 Formant

```
 0 "mdl":   E88
 1 "mix":   linf [0, 125, 126] %, mix
 2 "lf":    linf [0, 10, 101] lf freq
 3 "lg":    linf [-5, 5, 101] lf gain
 4 "lq":    str [LOW, HIGH] lf q
 5 "lt":    str [BELL, SHELV] lf type
 6 "lmf":   linf [0, 10, 101] lm freq
 7 "lmg":   linf [-5, 5, 101] lm gain
 8 "lmq":   linf [0, 10, 101] lm q
 9 "hmf":   linf [0, 10, 101] hm freq
10 "hmg":   linf [-5, 5, 101] hm gain
11 "hmq":   linf [0, 10, 101] hm q
12 "hf":    linf [0, 10, 101] hm freq
13 "hg":    linf [-5, 5, 101] hf gain
14 "hq":    str [LOW, HIG] hf q
15 "ht":    str [BELL, SHELV] hf type
```

Even 84

```
 0 "mdl":   E84
 1 "mix":   linf [0, 125, 126] %, mix
 2 "g":     linf [-20, 20, 81] dB, gain
 3 "lf":    str [OFF, 35, 60, 110, 220] lf freq
 4 "lg":    linf [-5, 5, 101] lf gain
 5 "mf":    str [OFF, 350, 700, 1k6, 3k2,
                 4k8, 7k2] mid freq
 6 "mg":    linf [-5, 5, 101] mid gain
 7 "mq":    str [LOW, HIGH] mid q
 8 "hf":    str [10k, 12k, 16k, OFF] hf freq
 9 "hg":    linf [-5, 5, 101] hf gain
```

Fortissimo110

```
 0 "mdl":   F110
 1 "mix":   linf [0, 125, 126] %, mix
 2 "peq":   int [0, 1] peq on
 3 "lmf":   linf [0, 10, 101] lm freq
 4 "lmg":   linf [-5, 5, 101] lm gain
 5 "lmq":   linf [0, 10, 101] lm q
 6 "lmf3":  int [0, 1] lm /3
 7 "hmf":   linf [0, 10, 101] hm freq
 8 "hmg":   linf [-5, 5, 101] hm gain
 9 "hmq":   linf [0, 10, 101] hm q
10 "hmf3":  int [0, 1] hm x3
11 "shv":   inf [0, 1] shv on
12 "lf":    str [33, 56, 95, 160,
                 270, 460] lf freq
13 "lg":    linf [-5, 5, 101] lf gain
14 "hf":    str [3k3, 4k7, 6k8, 10k,
                 15k, 18k] hf freq
15 "hg":    linf [-5, 5, 101] hf q
16 "g":     linf [-18, 18, 73] gain
```

Pulsar

```
 0 "mdl":   PULSAR
 1 "mix":   linf [0, 125, 126] %, mix
 2 "eq1":   int [0, 1] eq1 on
 3 "1lb":   linf [0, 10, 101] lf boost
 4 "1latt": linf [0, 10, 101] lf att
 5 "1lf":   str [20, 30, 60, 100] Hz, lf freq
 6 "1hw":   linf [0, 10, 101] hf wid
 7 "1hb":   linf [0, 10, 101] hf boost
 8 "1hf":   str [3k, 4k, 5k, 8k, 10k,
                 12k, 16k] Hz, hf freq
 9 "1hatt": linf [0, 10, 101] hf att
10 "1hattf":str [5k, 10k, 20k] hf att
11 "eq5":   inf [0, 1] eq5 on
12 "5lb":   linf [0, 10, 101] lm boost
13 "5lf":   str [200, 300, 500, 700,
                 1k] Hz, lf freq
14 "5md":   linf [0, 10, 101] mid dip
15 "5mf":   str [200, 300, 500, 700, 1k, 1k5,
                 2k, 3k, 4k, 5k, 7k] Hz, mid freq
16 "5hb":   linf [0, 10, 101] HM boost
17 "5hf":   str [1k5, 2k, 3k, 4k,
                 5k] Hz, hf freq
```

Mach EQ4

```
 0 "mdl":   MACH4
 1 "mix":   linf [0, 125, 126] %, mix
 2 "sub":   linf [-5, 5, 101] sub
 3 "40":    linf [-5, 5, 101] 40
 4 "160":   linf [-5, 5, 101] 160
 5 "650":   linf [-5, 5, 101] 650
 6 "2k5":   linf [-5, 5, 101] 2k5
 7 "air":   linf [0, 10, 101] air
 8 "airm":  str [OFF, 2k5, 5k, 10k,
                 20k, 40k] air mode
 9 "again": int [0, 1] auto
```

Premium effects

	None *0 **"mdl"**: NONE*
	External *0 **"mdl"**: EXT* *1 **"egrp"**: str [OFF, LCL, AUX, A, B, C, SC, USB, CRD, MOD, PLAY, AES] ext grp* *2 **"ein"**: int [1…64] ext in* *3 **"emode"**: str [M, ST, M/S] ext mode* *4 **"lat"**: int [0…200] latency* *5 **"trim"**: linf [-18, 18, 361] dB, trim*
	Hall Reverb *0 **"mdl"**: HALL* *1 **"pdel"**: int [0…200] ms, pre-delay* *2 **"size"**: int [0…100] hall size* *3 **"dcy"**: logf [.2, 5, 101] s, decay* *4 **"mult"**: logf [.5, , 101] bass multiplier* *5 **"damp"**: logf [1k, 20k, 51] Hz, damping* *6 **"lc"**: logf [20, 400, 51] Hz, low cut* *7 **"hc"**: logf [200, 20k, 51] Hz, high cut* *8 **"shp"**: linf [0, 50, 51] shape* *9 **"sprd"**: int [0…50] spread* *10 **"diff"**: int [1…30] diffusion* *11 **"mspd"**: int [0…100] mod speed*
	Room Reverb *0 **"mdl"**: R-OOM* *1 **"pdel"**: int [0…200] ms, pre-delay* *2 **"size"**: linf [4, 76, 145] m, room size* *3 **"dcy"**: logf [.3, 25, 101] s, decay* *4 **"mult"**: logf [.25, 4, 101] bass multiplier* *5 **"damp"**: logf [1k, 20k, 51] Hz, damping* *6 **"lc"**: logf [20, 400, 51] Hz, low cut* *7 **"hc"**: logf [200, 20k, 51] Hz, high cut* *8 **"shp"**: linf [0, 250, 51] shape* *9 **"sprd"**: int [0…50] spread* *10 **"diff"**: int [0…100] diffusion* *11 **"spin"**: int [0…100] spin* *12 **"ecl"**: linf [0, 1200, 1201] ms, echo left* *13 **"ecr"**: linf [0, 1200, 1201] ms, echo right* *14 **"efl"**: linf [-100, 100, 201] %, feed left* *15 **"efr"**: linf [-100, 100, 201] %, feed right*
	Chamber Reverb *0 **"mdl"**: CHAMBER* *1 **"pdel"**: int [0…200] ms, pre-delay* *2 **"size"**: linf [4, 76, 145] m, room size* *3 **"dcy"**: logf [.3, 25, 101] s, decay* *4 **"mult"**: logf [.25, 4, 101] bass multiplier* *5 **"damp"**: logf [1k, 20k, 51] Hz, damping* *6 **"lc"**: logf [20, 400, 51] Hz, low cut* *7 **"hc"**: logf [200, 20k, 51] Hz, high cut* *8 **"shp"**: linf [0, 250, 51] shape* *9 **"sprd"**: int [0…50] spread* *10 **"diff"**: int [0…100] diffusion* *11 **"spin"**: int [0…100] spin* *12 **"ecl"**: linf [0, 300, 301] ms, echo left* *13 **"ecr"**: linf [0, 300, 301] ms, echo right* *14 **"ell"**: fader lvl dB, echo left* *15 **"elr"**: fader lvl dB, echo right*

Plate Reverb

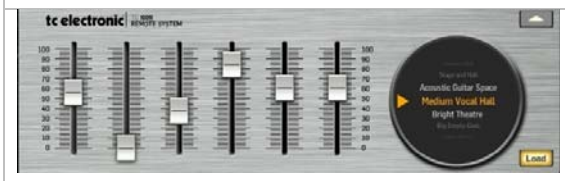
*0 **"mdl"**: PLATE*
*1 **"pdel"**: int [0…200] ms, pre-delay*
*2 **"size"**: linf [4, 76, 145] m, room size*
*3 **"dcy"**: logf [.3, 25, 101] s, decay*
*4 **"mult"**: logf [.25, 4, 101] bass multiplier*
*5 **"damp"**: logf [1k, 20k, 51] Hz, damping*
*6 **"lc"**: logf [20, 400, 51] Hz, low cut*
*7 **"hc"**: logf [200, 20k, 51] Hz, high cut*
*8 **"att"**: linf [0, 100, 101] attack*
*9 **"sprd"**: int [0…50] spread*
*10 **"diff"**: int [0…100] diffusion*
*11 **"spin"**: int [0…100] spin*
*12 **"ecl"**: linf [0, 1200, 1201] ms, echo left*
*13 **"ecr"**: linf [0, 1200, 1201] ms, echo right*
*14 **"efl"**: linf [-100, 100, 201] %, feed left*
*15 **"efr"**: linf [-100, 100, 201] %, feed right*

Concert Reverb

*0 **"mdl"**: CONCERT*
*1 **"pdel"**: int [0…200] ms, pre-delay*
*2 **"size"**: linf [20, 76, 113] m, room size*
*3 **"dcy"**: logf [.3, 29, 51] s, decay*
*4 **"mult"**: logf [.25, 4, 101] bass multiplier*
*5 **"damp"**: logf [1k, 20k, 51] Hz, damping*
*6 **"lc"**: logf [20, 400, 51] Hz, low cut*
*7 **"hc"**: logf [200, 20k, 51] Hz, high cut*
*8 **"shp"**: linf [0, 50, 51] shape*
*9 **"sprd"**: int [0…50] spread*
*10 **"diff"**: int [1…16] diffusion*
*11 **"depth"**: int [0, 100] depth*
*12 **"rfl"**: linf [0, 1200, 1201] ms, refl. left*
*13 **"rfr"**: linf [0, 1200, 1201] ms, refl. right*
*14 **"rfll"**: fader lvl dB, reflection left*
*15 **"rflr"**: fader lvl dB, reflection right*
*16 **"spin"**: int [0…100] spin*
*17 **"crs"**: int [1…100] chorus*

Ambiance

*0 **"mdl"**: AMBI*
*1 **"pdel"**: int [0…200] ms, pre-delay*
*2 **"size"**: linf [2, 100, 99] m, room size*
*3 **"dcy"**: logf [.2, 7.3, 101] s, decay*
*4 **"tail"**: int [0…100] tail gain*
*5 **"damp"**: logf [1k, 20k, 51] Hz, damping*
*6 **"diff"**: int [1…30] diffusion*
*7 **"mod"**: int [1…100] modulation speed*
*8 **"lc"**: logf [20, 400, 51] Hz, low cut*
*9 **"hc"**: logf [200, 20k, 51] Hz, high cut*



VSS3 Reverb

*0 **"mdl"**: VSS3*
*1 **"preset"**: str [Build, Small Booth, Home Room, Dialog Alley, Small Wood Room, A Small Room, Intimate Studio, Small Room, Tight & Natural, Room Conversation, Furnished Room 2, Studio 20x20 ft, Drew Room, Piano Close, Clear Guitar Room, Wide Ambient Chamber, Small Dense Hall, Slap Hall, Acoustic Gtr Ambience, Clear Room, Livingroom, Band Rehearsal Room, The Studio, In The Room, Studio 40x40 ft, Hit Room, Ambient Hall, Stage and Hall, Acoustic Guitar Space, Medium Vocal Hall, Bright Theatre , Big Empty Club, Venue Warm 1, Concert 1, Bright Guitar Hall, Concert Arena, Concert Piano, Piano Hall 1st Row, Empty Arena, Ballad Vocal*

	Hall, Grand Vocal Hall, Large Warm Hall, Back There, WoodHall, Church, Sound Col, 5000 Hall, Cathedral, Large Church, Medium Church, Warm Cathedral, Cologne Cathedral, Drum Plate Stuff, Drum Wood Plate, Piano Plate, Stairway Plate, Slapback Plate, Ambient Plate, Silky Gold Plate, Gold Plate, EMT 141, Leader Of The Band, VocalDry, Vocal Room, VocalWet, Slapback Vox 1, Vocal Hall 1, Vocal Chamber, Bright Male Vox, Vocal Bright, Vocal Deep, Vocal Female, Vocal Deep Male, Large Vocal Hall, Kick & Bass Ambience, Drum Room, Small Perc Room, Drum Room Xpander, Bright Shoe Gaze Snare, Snare Room Bright, Tom-Tom Reverb, Bossa Nova Perc Room, Hard Drum Space, Puk Drum Ambience, Overhead Mics, Dance Snare, Drum Perc Soft 1, Perc Straight Tail, Store Room, Studio Small, The Alley, Near The Wall, WoodFlr, Large Office, Conference Room, Dance Studio, Forest 2, StoneWall, Venue 1, Small Stairway, Forest 1, Airport PA, Small Tower Hall, On The Street, Dark Tunnel, Empty Nightclub, Parking Garage, Parking Distant, Long Swimmingpool] preset 2 **"load"**: int [0, 1] load 3 **"erpdly"**: linf [0, 100, 101] ms, er pdly 4 **"ertype"**: str [ROYAL, THEATRE, CHURCH, GAS, CONCERT, ROYAL2, V1-NEAR, V2-HARD, V3-SPREAD, V4-BUILD, V5-RANDOM, SLAP, CAR, PHONEBTH, BATHROOM, CONFRM9, CONFRM30, GARAGE, SWIMSTDM, AIRPORT, STREET, ALLEY, PIAZA, FOREST], er type 5 **"ersize"**: str [SML, MED, LRG] er size 6 **"erpos"**: str [NEAR, DIST] er position 7 **"erbal"**: linf [-100, 100, 201], er balance 8 **"erlc"**: logf [20, 400, 51] Hz, er low cut 9 **"ercol"**: linf [-40, 40, 81] er color 10 **"erlvl"**: fader lvl dB, er level 11 **"rvtype"**: str [SMOOTH, NATURAL, ALIVE, FAST, X-WIDE, ALIVE2] rev type 12 **"rvwide"**: str [NARROW, NORMAL, WIDE, X-WIDE] rev wide 13 **"rvpdly"**: linf [0, 200, 201] ms, rev pdly 14 **"dcy"**: linf [.1, 20, 280] s, decay 15 **"diff"**: linf [-50, 50, 101] diffuse 16 **"rvbal"**: linf [-100, 100, 201] balance 17 **"rvlvl"**: fader lvl dB, reverb level 18 **"ldcy"**: linf [.1, 2.5, 25] low decay 19 **"lmdcy"**: linf [.1, 2.5, 25] lowmid decay 20 **"hmdcy"**: linf [.1, 2.5, 25] mid decay 21 **"hdcy"**: linf [.1, 2.5, 25] high decay 22 **"hsoft"**: linf [-50, 50, 101] high soft 23 **"lxo"**: logf [20, 500, 113] Hz, low xover 24 **"mxo"**: logf [200, 2000, 81] Hz, mid xover 25 **"hxo"**: logf [500, 20000, 105] Hz, high xover 26 **"lshv"**: logf [20, 200, 81] Hz, low shelf 27 **"lsdmp"**: linf [0, -18, 37] dB, low damp 28 **"hcut"**: logf [20, 20000, 241] Hz, high cut 29 **"mtype"**: str [A, B, C, D, E, F] modulation type 30 **"mrate"**: linf [-100, 100, 201] modulation rate 31 **"mwid"**: linf [0, 200, 201] modulation width 32 **"view"**: int [0, 1] view

Vintage Room

```
 0 "mdl":   V-ROOM
 1 "pdel":  int [0…200] ms, pre-delay
 2 "size":  int [0…50] size
 3 "dcy":   logf [.1, 20, 101] s, decay
 4 "dens":  linf [1, 30, 30] density
 5 "erlvl": linf [0, 100, 101] %, Early level
 6 "lmult": logf [.1, 10, 101] low multiplier
 7 "hmult": logf [.1, 10, 101] high multiplier
 8 "lc":    logf [20, 400, 51] Hz, low cut
 9 "hc":    logf [200, 20k, 51] Hz, high cut
10 "frz":   int [0, 1] freeze
```

Vintage Reverb,

```
 0 "mdl":   V-REV
 1 "pdel":  int [0…120] ms, pre-delay
 2 "dcy":   linf [.4, 4.5, 83] s, decay
 3 "lmult": logf [.5, 2, 51] low multiplier
 4 "hmult": logf [.25, .67, 51] high multiplier
 5 "mod":   int [0…100] modulation speed
 6 "lc":    logf [20, 400, 51] Hz, low cut
 7 "hc":    logf [5000, 20k, 31] Hz, high cut
 8 "out":   str [FRONT, REAR] output
 9 "trans": int [0…1] transformer
```

Vintage Plate

```
 0 "mdl":   V-PLATE
 1 "pdel":  int [0…250] ms, pre-delay
 2 "dcy":   linf [1, 6, 101] s, decay
 3 "lc":    logf [20, 400, 51] Hz, low cut
 4 "col":   linf [-20, 20, 42] color
```

Blue Plate

```
 0 "mdl":   BPLATE
 1 "pdel":  int [0…200] ms, pre delay
 2 "size":  int [0…100] ms, size
 3 "dcy":   logf [0.2, 5, 101] s, decay
 4 "mult":  logf [0.5, 2, 51] bass multiplier
 5 "damp":  logf [1000, 20000, 51] Hz, damping
 6 "lc":    logf [20, 400, 51] Hz, low cut
 7 "hc":    logf [200, 20000, 51] Hz, high cut
 8 "xover": logf [20, 500, 51] Hz, xover
 9 "mdep":  linf [1, 50, 50] modulation depth
10 "msdp":  int [0…100] modulation speed
11 "diff":  int [1…30] diffusion
```

Gated Reverb

```
 0 "mdl":  GATED
 1 "pdel": int [0…200] ms, pre-delay
 2 "att":  int [4…30] attack
 3 "dcy":  logf [.14, 1, 101] s, decay
 4 "dens": int [0…100] density
 5 "diff": int [0…100] diffusion
 6 "sprd": int [0…50] spread
 7 "lc":   logf [20, 400, 51] Hz, low cut
 8 "hfs":  logf [200, 20k, 51] Hz, high freq
 9 "hsg":  linf [-30, 0, 61] dB, high gain
```

Reverse Reverb

```
 0 "mdl":  REVERSE
 1 "pdel": int [0…200] ms, pre-delay
 2 "rise": int [4…50] rise
 3 "dcy":  logf [.14, 1, 101] s, decay
 4 "diff": int [0…30] diffusion
 5 "sprd": int [0…100] spread
 6 "lc":   logf [20, 400, 51] Hz, low cut
 7 "hfs":  logf [200, 20k, 51] Hz, high freq
 8 "hsg":  linf [-30, 0, 61] dB, high gain
```

Delay/Reverb

```
 0 "mdl":   DEL/REV
 1 "time":  linf [0, 3000, 3000] ms, time
 2 "feed":  linf [0, 100, 101] %, feed
 3 "fhc":   logf [200, 2000, 51] Hz, feed HC
 4 "dly":   linf [0, 100, 101] %, delay
 5 "d2r":   linf [0, 100, 101] %, delay→rev
 6 "pdel":  int [0…200] ms, pre delay
 7 "size":  int [2…100] size
 8 "dcy":   logf [.1, 5, 51] s, decay
 9 "damp":  logf [1000, 20k, 51] Hz, damp
10 "rlc":   logf [20, 400, 51] Hz, rev LC
11 "i2r":   linf [0, 100, 101] %, in→rev
```

Shimmer Reverb

```
 0 "mdl":   SHIMMER
 1 "pdel":  int [0…250] ms, pre delay
 2 "size":  int [2…50] size
 3 "dcy":   logf [1, 20, 101] s, decay
 4 "lc":    logf [25, 250, 51] Hz, low cut
 5 "hc":    logf [500, 7000, 51] Hz, high cut
 6 "damp":  linf [0, 100, 101] %, damp
 7 "shim":  linf [0, 100, 101] %, shimmer
 8 "shine": linf [0, 100, 101] %, shine
```

Spring Reverb

```
 0 "mdl":   SPRING
 1 "dcy":   logf [1.5, 6, 101] s, decay
 2 "dens":  linf [1, 30, 30] density
 3 "low":   linf [1, 50, 50] bass
 4 "high":  linf [1, 50, 50] trebble
```

Dimension CRS

```
 0 "mdl":   DIMCRS
 1 "sw1":   int [0, 1] sw1
 2 "sw2":   int [0, 1] sw2
 3 "sw3":   int [0, 1] sw3
 4 "sw4":   int [0, 1] sw4
 5 "in":    str [MONO, STEREO] input
 6 "drysw": int [0, 1] dry
```

Stereo Chorus

```
 0 "mdl":   CHORUS
 1 "lc":    logf [20, 400, 51] Hz, LC
 2 "hc":    logf [200, 20000, 51] Hz, HC
 3 "wave":  linf [0, 100, 101] waveform
 4 "phase": linf [0, 100, 101] phase
 5 "mix":   linf [0, 100, 101] %, mix
 6 "dlyl":  linf [5, 50, 226] ms, dely l
 7 "dlyr":  linf [5, 50, 226] ms, dely r
 8 "depl":  linf [0, 100, 101] %, depth l
 9 "depr":  linf [0, 100, 101] %, depth r
10 "sprd":  linf [0, 100, 101] %, spread
11 "spd":   logf [.05, 5, 201] Hz, speed
```

Stereo Flanger

```
 0 "mdl":   CHORUS
 1 "lc":    logf [20, 400, 51] Hz, LC
 2 "hc":    logf [200, 20000, 51] Hz, HC
 3 "flc":   logf [20, 400, 51] Hz, feed LC
 4 "fhc":   logf [200, 20000, 51] Hz, feed HC
 5 "mix":   linf [0, 100, 101] %, mix
 6 "dlyl":  linf [5, 20, 196] ms, dely l
 7 "dlyr":  linf [5, 20, 196] ms, dely r
 8 "depl":  linf [0, 100, 101] %, depth l
 9 "depr":  linf [0, 100, 101] %, depth r
10 "phase":  linf [0, 180, 181] phase
```

	11 **"spd"**: logf [.05, 5, 201] Hz, speed 12 **"feed"**: linf [-90, 90, 181] %, feed
	Stereo Delay 0 **"mdl"**: ST-DL 1 **"time"**: linf [1, 3000, 3000] ms, time 2 **"mode"**: str [ST, X, M] mode 3 **"fact"**: str [1/4, 1/3, 1/2, 2/3, 3/4, 1, 3/8, 5/4, 4/3, 3/2, 2] factor 4 **"pat"**: str [1/2:1, 2/3:1, 3/4:1, 7/8:1, 1:1, 1:9/8, 1:5/4, 1:4/3, 1:3/2] pattern 5 **"offset"**:int [-50…50] ms, offset 6 **"feed"**: linf [0, 100, 101] %, feed 7 **"flc"**: logf [20, 400, 51] Hz, feed L cut 8 **"fhc"**: logf [200, 20000, 51] Hz, feed H cut 9 **"lc"**: logf [20, 400, 51] Hz, low cut 10 **"hc"**: logf [200, 20000, 51] Hz, high cut
	UltraTap Delay 0 **"mdl"**: TAP-DL 1 **"time"**: linf [1, 2000, 2000] ms, time 2 **"rep"**: int [1..16] repeat 3 **"slp"**: linf [-6, 6, 121] dB, slope 4 **"fact"**: str [1/3, 1/2, 2/3, 3/4, 1, 5/4, 4/3, 3/2, 2] factor 5 **"pdel"**: linf [0, 500, 501] ms, pre delay 6 **"mode"**: str [MOVE, JUMP, FOCUS, SPREAD] mode 7 **"wid"**: linf [-100, 100, 201] %, width 8 **"diff"**: linf [0, 100, 101] diffusion 9 **"lc"**: logf [20, 400, 51] Hz, low cut 10 **"hc"**: logf [200, 20000, 51] Hz, high cut
	Tape Delay 0 **"mdl"**: TAPE-DL 1 **"time"**: linf [60, 650, 591] ms, time 2 **"sust"**: linf [0, 100, 101] %, sustain 3 **"drv"**: linf [0, 100, 101] %, drive 4 **"wf"**: linf [0, 100, 101] %, flutter
	OilCan Delay 0 **"mdl"**: OILCAN 1 **"time"**: linf [0, 10, 1001] time 2 **"sust"**: linf [0, 10, 101] %, sustain 3 **"wb"**: linf [0, 10, 101] %, wobble 4 **"tone"**: linf [0, 10, 101] %, tone
	BBD Delay 0 **"mdl"**: BBD-DL 1 **"dly"**: linf [0, 100, 1001] time 2 **"feed"**: linf [0, 100, 101] %, feed
	Stereo Pitch 0 **"mdl"**: PITCH 1 **"semi"**: int [-12…12] semitones 2 **"cent"**: int [-50…50] cent 3 **"dly"**: linf [0, 500, 501] ms, delay 4 **"lc"**: logf [20, 400, 51] Hz, low cut 5 **"hc"**: logf [200, 20000, 51] Hz, high cut 6 **"mix"**: linf [0, 100, 101] %, mix

Dual Pitch

```
 0 "mdl":   D-PITCH
 1 "semi1": int [-12…12] semitones 1
 2 "cent1": int [-50…50] cent 1
 3 "dly1":  linf [0, 500, 501] ms, delay 1
 4 "pan1":  linf [-100, 100, 201] %, pan 1
 5 "lvl1":  fader lvl 1 dB
 6 "semi2": int [-12…12] semitones 2
 7 "cent2": int [-50…50] cent 2
 8 "dly2":  linf [0, 500, 501] ms, delay 2
 9 "pan2":  linf [-100, 100, 201] %, pan 2
10 "lvl2":  fader lvl 2 dB
11 "lc":    logf [20, 400, 51] Hz, low cut
12 "hc":    logf [200, 20000, 51] Hz, high cut
```

Channel effects

None

```
0 "mdl":    NONE
```

External

```
0 "mdl":    EXT
1 "egrp":   str [OFF, LCL, AUX, A, B, C, SC,
                 USB, CRD, MOD, PLAY, AES] ext grp
2 "ein":    int [1…64] ext in
3 "emode":  str [M, ST, M/S] ext mode
4 "lat":    int [0…200] latency
5 "trim":   linf [-18, 18, 361] dB, trim
```

Soul Analog EQ

```
0 "mdl":    SOUL
1 "mix":    linf [0, 125, 126] %, mix
2 "lf":     linf [0, 10, 101] lo freq
3 "lg":     linf [-5, 5, 101] lo gain
4 "lmf":    linf [0, 10, 101] lm freq
5 "lmf3":   int [0, 1] lm /3
6 "lmq":    linf [0, 10, 101] lm q
7 "lmg":    linf [-5, 5, 101] lm gain
8 "hmf":    linf [0, 10, 101] hm freq
9 "hmf3":   int [0, 1] hm x3
10 "hmq":   linf [0, 10, 101] hm q
11 "hmg":   linf [-5, 5, 101] hm gain
12 "hf":    linf [0, 10, 101] hf freq
13 "hg":    linf [-5, 5, 101] hf gain
```

Even 88-Formant EQ

```
0 "mdl":    E88
1 "mix":    linf [0, 125, 126] %, mix
2 "lf":     linf [0, 10, 101] lf freq
3 "lg":     linf [-5, 5, 101] lf gain
4 "lq":     str [LOW, HIGH] lf q
5 "lt":     str [BELL, SHELV] lf type
6 "lmf":    linf [0, 10, 101] lm freq
7 "lmg":    linf [-5, 5, 101] lm gain
8 "lmq":    linf [0, 10, 101] lm q
9 "hmf":    linf [0, 10, 101] hm freq
10 "hmg":   linf [-5, 5, 101] hm gain
11 "hmq":   linf [0, 10, 101] hm q
12 "hf":    linf [0, 10, 101] hm freq
13 "hg":    linf [-5, 5, 101] hf gain
14 "hq":    str [LOW, HIG] hf q
15 "ht":    str [BELL, SHELV] hf type
```

Even 84 EQ

```
0 "mdl":    E84
1 "mix":    linf [0, 125, 126] %, mix
2 "g":      linf [-20, 20, 81] dB, gain
3 "lf":     str [OFF, 35, 60, 110, 220] lf freq
4 "lg":     linf [-5, 5, 101] lf gain
5 "mf":     str [OFF, 350, 700, 1k6, 3k2,
                 4k8, 7k2] mid freq
6 "mg":     linf [-5, 5, 101] mid gain
7 "mq":     str [LOW, HIGH] mid q
8 "hf":     str [10k, 12k, 16k, OFF] hf freq
9 "hg":     linf [-5, 5, 101] hf gain
```

Focusrite ISA 110 EQ

```
0 "mdl":    F110
1 "mix":    linf [0, 125, 126] %, mix
2 "peq":    int [0, 1] peq on
3 "lmf":    linf [0, 10, 101] lm freq
4 "lmg":    linf [-5, 5, 101] lm gain
5 "lmq":    linf [0, 10, 101] lm q
```

	6 **"lmf3"**: int [0, 1] lm /3 7 **"hmf"**: linf [0, 10, 101] hm freq 8 **"hmg"**: linf [-5, 5, 101] hm gain 9 **"hmq"**: linf [0, 10, 101] hm q 10 **"hmf3"**: int [0, 1] hm x3 11 **"shv"**: inf [0, 1] shv on 12 **"lf"**: str [33, 56, 95, 160, 270, 460] lf freq 13 **"lg"**: linf [-5, 5, 101] lf gain 14 **"hf"**: str [3k3, 4k7, 6k8, 10k, 15k, 18k] hf freq 15 **"hg"**: linf [-5, 5, 101] hf q 16 **"g"**: linf [-18, 18, 73] gain
	Pulsar P1a/M5 EQ 0 **"mdl"**: PULSAR 1 **"mix"**: linf [0, 125, 126] %, mix 2 **"eq1"**: int [0, 1] eq1 on 3 **"1lb"**: linf [0, 10, 101] lf boost 4 **"1latt"**: linf [0, 10, 101] lf att 5 **"1lf"**: str [20, 30, 60, 100] Hz, lf freq 6 **"1hw"**: linf [0, 10, 101] hf wid 7 **"1hb"**: linf [0, 10, 101] hf boost 8 **"1hf"**: str [3k, 4k, , 5k, 8k, 10k, 12k, 16k] Hz, hf freq 9 **"1hatt"**: linf [0, 10, 101] hf att 10 **"1hattf"**:str [5k, 10k, 20k] hf att 11 **"eq5"**: int [0, 1] eq5 on 12 **"5lb"**: linf [0, 10, 101] lm boost 13 **"5lf"**: str [200, 300, 500, 700, 1k] Hz, lf freq 14 **"5md"**: linf [0, 10, 101] mid dip 15 **"5mf"**: str [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k] Hz, mid freq 16 **"5hb"**: linf [0, 10, 101] HM boost 17 **"5hf"**: str [1k5, 2k, 3k, 4k, 5k] Hz, hf freq
	Mach EQ4 0 **"mdl"**: MACH4 1 **"mix"**: linf [0, 125, 126] %, mix 2 **"sub"**: linf [-5, 5, 101] sub 3 **"40"**: linf [-5, 5, 101] 40 4 **"160"**: linf [-5, 5, 101] 160 5 **"650"**: linf [-5, 5, 101] 650 6 **"2k5"**: linf [-5, 5, 101] 2k5 7 **"air"**: linf [0, 10, 101] air 8 **"airm"**: str [OFF, 2k5, 5k, 10k, 20k, 40k] air mode 9 **"again"**: int [0, 1] auto
	Even Channel Even 88 Gate, Even 88 Formant EQ, Even Compressor/Limiter 0 **"mdl"**: *EVEN* 1 **"g_thr"**: Linf [-40.0, 0.0, 81] dB 2 **"g_hyst"**: Linf [0.0, 25.0, 51] dB 3 **"g_range"**: Linf [0, 60, 61] dB 4 **"g_rel"**: Logf [100, 3000, 130] ms 5 **"g_fast"**: int [0, 1] 6 **"g_m40"**: int [0, 1] 7 **"g_on"**: int [0, 1] 8 **"eq_on"**: int [0, 1] 9 **"lf"**: Linf [0.0, 10.0, 101] 10 **"lg"**: Linf [-5.0, 5.0, 101] 11 **"lq"**: Str [LOW, HIGH] 12 **"lt"**: Str [BELL, SHELV] 13 **"lmf"**: Linf [0.0, 10.0, 101] 14 **"lmg"**: Linf [-5.0, 5.0, 101] 15 **"lmq"**: Linf [0.0, 10.0, 101] 16 **"hmf"**: Linf [0.0, 10.0, 101]

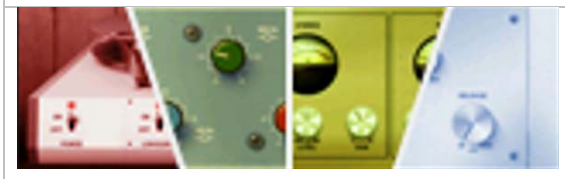
	17 “hmg”: Linf [-5.0, 5.0, 101] *18 “hmq”: Linf [0.0, 10.0, 101]* *19 “hf”: Linf [0.0, 10.0, 101]* *20 “hg”: Linf [-5.0, 5.0, 101]* *21 “hq”: Str [LOW, HIGH]* *22 “ht”: Str [BELL, SHELV]* *23 “mix”: Linf [0, 125 %, 126]* *24 “d_lon”: int [0, 1]* *25 “d_lthr”: Linf [-12.0, 0.0, 25] dB* *26 “d_lrec”: Str [50, 100, 200, 800, A1, A2]* *27 “d_lfast”: int [0, 1]* *28 “d_con”: int [0, 1]* *29 “d_cthr”: Linf [-35.0, -5.0 dB, 61]* *30 “d_ratio”: Str [1.5, 2.0, 3.0, 4.0, 6.0]* *31 “d_crec”: Str [100, 400, 800, 1500, A1, A2]* *32 “d_cfast”: int [0, 1]* *33 “d_gain”: Linf [-6, 12 dB, 7]*
	Soul Channel Soul 9000 Gate/Expander, Soul AnaLogue EQ, Soul 9000 Channel Compressor *0 “mdl”: *SOUL** *1 “g_thr”: Linf [-40.0, 0.0, 81] dB* *2 “g_range”: Linf [0, 40, 41] dB* *3 “g_hld”: Logf [10, 4000, 130] ms* *4 “g_rel”: Logf [100, 4000, 130] ms* *5 “g_fast”: int [0, 1]* *6 “g_mode”: Str [GATE, EXP]* *7 “g_on”: int [0, 1]* *8 “eq_on”: int [0, 1]* *9 “lf”: Linf [0.0, 10.0, 101]* *10 “lg”: Linf [-5.0, 5.0, 101]* *11 “lmf”: Linf [0.0, 10.0, 101]* *12 “lmf3”: int [0, 1]* *13 “lmq”: Linf [0.0, 10.0, 101]* *14 “lmg”: Linf [-5.0, 5.0, 101]* *15 “hmf”: Linf [0.0, 10.0, 101]* *16 “hmf3”: int [0, 1]* *17 “hmq”: Linf [0.0, 10.0, 101]* *18 “hmg”: Linf [-5.0, 5.0, 101]* *19 “hf”: Linf [0.0, 10.0, 101]* *20 “hg”: Linf [-5.0, 5.0, 101]* *21 “mix”: Linf [0, 125 %, 126]* *22 “d_on”: int [0, 1]* *23 “d_thr”: Linf [-30.0, 18.0, 97] dB* *24 “d_ratio”: Str [1.3, 1.4, 1.6, 1.8, 2.0, 2.5, 2.8, 3.3, 4.0, 5.0, 6.0, 7.0, 9.0, 12, 20, 50, 100]* *25 “d_fast”: int [0, 1]* *26 “d_rel”: Logf [100, 4000, 65] ms* *27 “d_peak”: int [0, 1]*
	Vintage Channel 76 Limiting Amplifier, Pulsar EQ P1A/m5, Model 2A Leveling Amplifier *0 “mdl”: *VINTAGE** *1 “d_in”: Linf [-48.0, 0.0, 97] dB* *2 “d_out”: Linf [-48.0, 0.0, 97] dB* *3 “d_att”: Linf [1.0, 7.0, 61]* *4 “d_rel”: Linf [1.0, 7.0, 61]* *5 “d_ratio”: Str [4, 8, 12, 20, ALL]* *6 “d_on”: int [0, 1]* *7 “eq1”: int [0, 1]* *8 “1lb”: Linf [0.0, 10.0, 101]* *9 “1latt”: Linf [0.0, 10.0, 101]* *10 “1lf”: Str [20, 30, 60, 100]* *11 “1hw”: Linf [0.0, 10.0, 101]* *12 “1hb”: Linf [0.0, 10.0, 101]* *13 “1hf”: Str [3k, 4k, 5k, 8k, 10k, 12k, 16k]* *14 “1hatt”: Linf [0.0, 10.0, 101]* *15 “1hattf”: Str [5k, 10k, 20k]*

```
16 "eq5":      int [0, 1]
17 "5lb":      Linf [0.0, 10.0, 101]
18 "5lf":      Str [200, 300, 500, 700, 1k]
19 "5md":      Linf [0.0, 10.0, 101]
20 "5mf":      Str [200, 300, 500, 700, 1k, 1k5,
                    2k, 3k, 4k, 5k, 7k]
21 "5hb":      Linf [0.0, 10.0, 101]
22 "5hf":      Str [1k5, 2k, 3k, 4k, 5k]
23 "l_ingain": Linf [0, 100, 101]
24 "l_peak":   Linf [0, 100, 101]
25 "l_mode":   Str [COMP, LIM]
26 "l_on":     int [0, 1]
```

Bus Channel

Soul Warmth, Even 84 EQ, Soul G Bus Compressor

```
 0 "mdl":     *BUS*
 1 "w_drv":    Linf [10, 125, 116] %
 2 "w_hrm":    Linf [-100, 100, 201]
 3 "w_col":    Linf [-1.00, +1.00, 41]
 4 "w_trim":   Linf [-18.0, +6.0, 49] dB
 5 "w_mix":    Linf [0, 100, 101] %
 6 "w_on":     int [0, 1]
 7 "eq_on":    int [0, 1]
 9 "g":        Linf [-20.0, 20.0, 81] dB
10 "lf":       Str [OFF, 35, 60, 110, 220]
11 "lg":       Linf [-5.0, 5.0, 101]
12 "mf":       Str [OFF, 350, 700, 1k6, 3k2,
                    4k8, 7k2]
13 "mg":       Linf [-5.0, 5.0, 101]
14 "mq":       Str [LOW, HIGH]
15 "hf":       Str [10k, 12k, 16k, OFF]
16 "hg":       Linf [-5.0, 5.0, 101]
17 "mix":      Linf [0, 125 %, 126]
18 "d_thr":    Linf [-40.0, 0.0, 81] dB
19 "d_ratio":  Str [1.5, 2.0, 3.0, 4.0, 5.0, 10]
20 "d_att":    Str [0.1, 0.3, 1.0, 3.0, 10.0, 30.0]
21 "d_rel":    Str [0.1, 0.2, 0.4, 0.8, 1.6, AUTO]
22 "d_gain":   Linf [-6.0, 12.0, 37] dB
23 "d_on":     int [0, 1]
```



Mastering

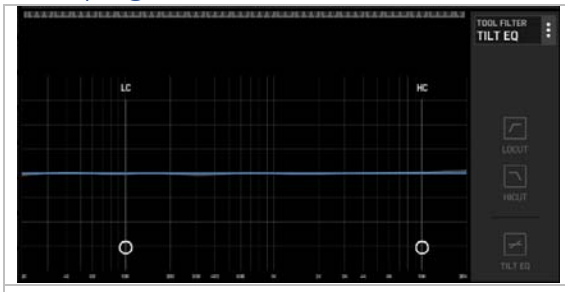
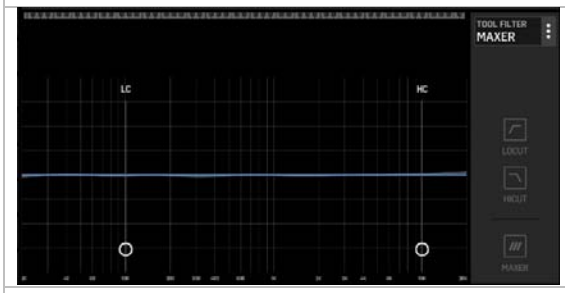
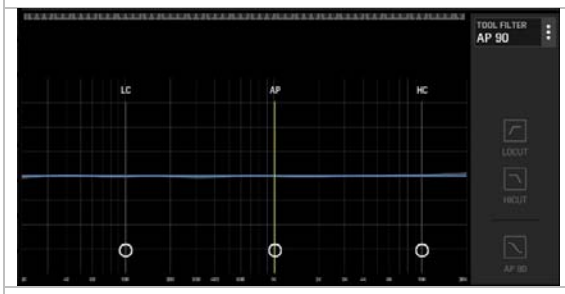
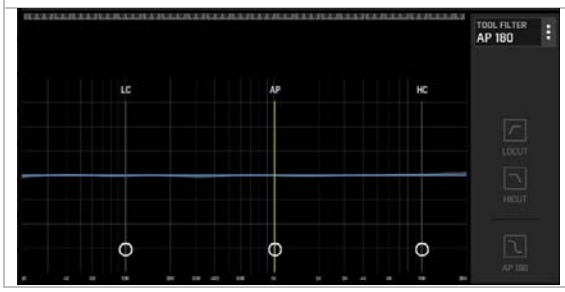
Tape, Mach EQ4 EQ, Stereo Enhancer, Precision Limiter

```
 0 "mdl":     *MASTER*
 1 "t_drv":    Linf [-5.0, 25.0, 61] dB
 2 "t_spd":    Logf [7.5, 30.0, 65]
 3 "t_low":    int [0, 1]
 4 "t_hi":     int [0, 1]
 5 "t_on":     int [0, 1]
 6 "sub":      Linf [-5.0, 5.0, 201]
 7 "40":       Linf [-5.0, 5.0, 201]
 8 "160":      Linf [-5.0, 5.0, 201]
 9 "650":      Linf [-5.0, 5.0, 201]
10 "2k5":      Linf [-5.0, 5.0, 201]
11 "air":      Linf [0.0, 10.0, 201]
12 "airm":     Str [OFF, 2k5, 5k, 10k, 20k, 40k]
13 "eq_on":    int [0, 1]
14 "e_stlvl":  Linf [-100, +100, 201] %
15 "e_lmf":    Linf [-100, +100, 201] %
16 "e_mlvl":   Linf [-100, +100, 201] %
17 "e_st":     Linf [-100, 100, 201] %
18 "e_m":      Linf [-100, 100, 201] %
19 "e_bass":   Linf [0, 100, 101] %
20 "e_mid":    Linf [0, 100, 101] %
21 "e_high":   Linf [0, 100, 101] %
22 "e_bassf":  Linf [1, 50, 50]
23 "e_midq":   Linf [1, 50, 50]
24 "e_highf":  Linf [1, 50, 50]
25 "e_on":     int [0, 1]
26 "l_gin":    Linf [0.00, 18.00, 73] dB
27 "l_gout":   Linf [-18.00, 0.00, 73] dB
28 "l_sqz":    int [0, 100]
```

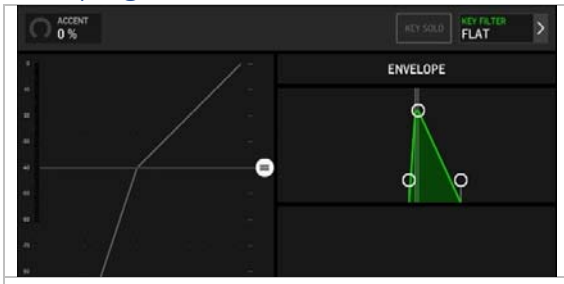
	29 “L_knee”: int [0, 10] 30 “L_again”: int [0, 1] 31 “L_att”: Linf [0.05, 1.00, 96] ms 32 “L_rel”: Logf [20, 2000, 101] ms

Plugins

Filter plugins

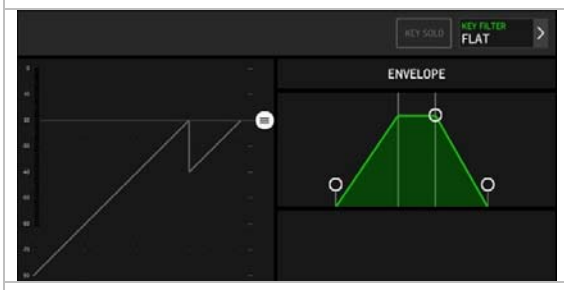
	Tilt Filter *0 **"mdl"**: TILT* *1 **"tilt"**: linf [-6, 6, 49] tilt*
	Maxer Filter *0 **"mdl"**: MAX* *1 **"low"**: linfplugins [0, 100, 101] %, low cont* *2 **"proc"**: linf [0, 100, 101] %, high proc*
	AP90 Filter (all pass) *0 **"mdl"**: AP1* *1 **"freq"**: logf [100, 10000, 100] Hz, freq*
	AP180 Filter (all pass) *0 **"mdl"**: AP2* *1 **"f"**: logf [100, 10000, 100] Hz, freq* *2 **"q"**: logf [.442, 10, 181] q*

Gate plugins



Standard Gate/Expander

```
0 "mdl":   GATE
1 "thr":   linf [-80, 0, 161] dB, thr
2 "range": linf [3, 60, 115] dB, range
3 "att":   linf [0, 120, 121] ms, attack
4 "hold":  linf [1, 200, 200] ms, hold
5 "rel":   logf [4, 4000, 130] ms, release
6 "acc":   linf [0, 100, 21] %, accent
7 "ratio": str [1:1.5, 1:2, 1:3, 1:4,
                gate] ratio
```



Standard Ducker

```
0 "mdl":   DUCK
1 "thr":   linf [-80, 0, 161] dB, thr
2 "range": linf [3, 60, 115] dB, range
3 "att":   linf [0, 120, 121] ms, attack
4 "hold":  linf [1, 200, 200] ms, hold
5 "rel":   linf [20, 4000, 130] ms, release
```

SSL 9000 Gate/Expander

```
0 "mdl":    9000G
1 "thr":    linf [-40, 0 81] dB, input
2 "range":  linf [-0, 40, 41] dB
3 "hld":    logf [10, 4000, 130] ms, hold
4 "rel":    logf [100, 4000, 130] ms, release
5 "fast":   int [0, 1] fast
6 "mode":   str [GATE, EXP] mode
```

Even 88-Gate

```
0 "mdl":   E88
1 "thr":   linf [-40, 0, 81] dB, thr
2 "hyst":  linf [0, 25, 51] dB, hyst
3 "range": linf [0, 60, 61] dB, range
4 "rel":   logf [100, 3000, 130] ms, release
5 "fast":  int [0, 1] fast
6 "m40":   int [0, 1] thr
```

DrawMore Expander Gate 241

```
0 "mdl":   DUCK
1 "thr":   linf [-80, 0, 161] dB, thr
2 "slow":  int [0, 1] slow
```

DBX 902 De-Esser

```
0 "mdl":   DS902
1 "f":     logf [800, 8000, 130] Hz, freq
2 "range": linf [3, 12, 25] dB, range
3 "mode":  str [FULL, HF] mode
```

76 Limiting Amp

```
0 "mdl":   76LA
1 "in":    linf [-48, 0, 97] dB, input
2 "out":   linf [-48, 0, 97] dB
3 "att":   linf [1, 7, 61] attack
4 "rel":   linf [1, 7, 61] release
5 "ratio": str [4, 8, 12, 20, ALL] ratio
```

Leveling Amplifier 2A

*0 **"mdl"**: LA*
*1 **"ingain"**:linf [0, 100, 101] gain*
*2 **"peak"**: linf [0, 100, 101] peak*
*3 **"mode"**: str [comp, lim] mode*

Source Extractor

*0 **"mdl"**: PSE*
*1 **"thr"**: linf [-36, 12, 97] dB, threahold*
*2 **"depth"**: linf [0, 20, 41] dB, depth*
*3 **"fast"**: int [0, 1] fast*
*4 **"peak"**: int [0, 1] peak*

Wave Designer

*0 **"mdl"**: WAVE*
*1 **"att"**: linf [-15, 15, 61] dB, attack*
*2 **"sust"**: linf [-24, 24, 97] dB, sustain*
*3 **"g"**: linf [-18, 9, 55] dB, gain*

Auto Rider Dynamics

*0 **"mdl"**: RIDE*
*1 **"thr"**: linf [-54, 18, 73] dB, thr*
*2 **"tgt"**: linf [-48, 0, 97] dB, target*
*3 **"spd"**: int [1…50] speed*
*4 **"ratio"**: flt [2.0, 4.0, 8.0, 20.0, 100.0] ratio*
*5 **"hld"**: logf [.1, 10, 65] s, hold*
*6 **"range"**: linf [1, 15, 29] dB, range*

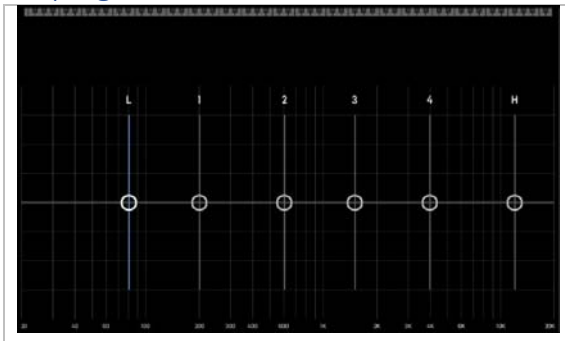
Soul Warmth Preamp

*0 **"mdl"**: WARM*
*1 **"drv"**: linf [10, 100, 91] %, drive*
*2 **"hrm"**: linf [-100, 100, 201] harm*
*3 **"col"**: linf [-1, 1, 41] color*
*3 **"trim"**: linf [-18, 6, 49] dB, trim*
*4 **"mix"**: linf [0, 100, 101] dB, mix*

Dynamic EQ

*0 **"mdl"**: DEQ*
*1 **"thr"**: linf [-60, 0, 121] dB, thr*
*2 **"ratio"**: flt [1.2, 1.3, 1.5, 2.0, 3.0, 5.0, 10.0] ratio*
*3 **"att"**: linf [0, 200, 201] ms, attack*
*4 **"rel"**: logf [20, 4000, 130] ms, release*
*5 **"filt"**: str [OFF, BP, LP6, LP12, HP6, HP12] filter*
*6 **"g"**: linf [-15, 15, 301] dB, gain*
*7 **"f"**: logf [20, 20000, 961] Hz, freq*
*8 **"q"**: logf [.442, 10, 181] q*
*9 **"mode"**: str [low, high] mode*

EQ plugins



Standard EQ

Channel:

```
 0 "mdl":  STD
 1 "lg":   linf [-15, 15, 301] dB, gain l
 2 "lf":   logf [20, 2000, 641] Hz, freq l
 3 "lq":   logf [0.442, 10, 181] q l
 4 "leq":  str [SHV, PEQ] eq l
 5 "1g":   linf [-15, 15, 301] dB, gain 1
 6 "1f":   logf [20, 20000, 961] Hz, freq 1
 7 "1q":   logf [0.442, 10, 181] q 1
 8 "2g":   linf [-15, 15, 301] dB, gain 2
 9 "2f":   logf [20, 20000, 961] Hz, freq 2
10 "2q":   logf [0.442, 10, 181] q 2
11 "3g":   linf [-15, 15, 301] dB, gain 3
12 "3f":   logf [20, 20000, 961] Hz, freq 3
13 "3q":   logf [0.442, 10, 181] q 3
14 "4g":   linf [-15, 15, 301] dB, gain 4
15 "4f":   logf [20, 20000, 961] Hz, freq 4
16 "4q":   logf [0.442, 10, 181] q 4
17 "hg":   linf [-15, 15, 301] dB, gain h
18 "hf":   logf [50, 20000, 833] Hz, freq h
19 "hq":   logf [0.442, 10, 181] q h
20 "heq":  str [SHV, PEQ] eq h
```

Bus, mtx, main:

```
 0 "mdl":  STD
 1 "lg":   linf [-15, 15, 301] dB, gain l
 2 "lf":   logf [20, 2000, 641] Hz, freq l
 3 "lq":   logf [0.442, 10, 181] q l
 4 "leq":  str [SHV, PEQ, CUT] eq l
 5 "1g":   linf [-15, 15, 301] dB, gain 1
 6 "1f":   logf [20, 20000, 961] Hz, freq 1
 7 "1q":   logf [0.442, 10, 181] q 1
 8 "2g":   linf [-15, 15, 301] dB, gain 2
 9 "2f":   logf [20, 20000, 961] Hz, freq 2
10 "2q":   logf [0.442, 10, 181] q 2
11 "3g":   linf [-15, 15, 301] dB, gain 3
12 "3f":   logf [20, 20000, 961] Hz, freq 3
13 "3q":   logf [0.442, 10, 181] q 3
14 "4g":   linf [-15, 15, 301] dB, gain 4
15 "4f":   logf [20, 20000, 961] Hz, freq 4
16 "4q":   logf [0.442, 10, 181] q 4
17 "5g":   linf [-15, 15, 301] dB, gain 5
18 "5f":   logf [20, 20000, 961] Hz, freq 5
19 "5q":   logf [0.442, 10, 181] q 5
20 "6g":   linf [-15, 15, 301] dB, gain 6
21 "6f":   logf [20, 20000, 961] Hz, freq 6
22 "7q":   logf [0.442, 10, 181] q 6
23 "hg":   linf [-15, 15, 301] dB, gain h
24 "hf":   logf [50, 20000, 833] Hz, freq h
25 "hq":   logf [0.442, 10, 181] q h
26 "heq":  str [SHV, PEQ, CUT] eq h
27 "tilt": linf [-6, 6, 49] dB, tilt
```

Soul Analog EQ

```
 0 "mdl":   SOUL
 1 "mix":   linf [0, 125, 126] %, mix
 2 "lf":    linf [0, 10, 101] lo freq
 3 "lg":    linf [-5, 5, 101] lo gain
 4 "lmf":   linf [0, 10, 101] lm freq
 5 "lmf3":  int [0, 1] lm /3
 6 "lmq":   linf [0, 10, 101] lm q
 7 "lmg":   linf [-5, 5, 101] lm gain
 8 "hmf":   linf [0, 10, 101] hm freq
 9 "hmf3":  int [0, 1] hm x3
10 "hmq":   linf [0, 10, 101] hm q
11 "hmg":   linf [-5, 5, 101] hm gain
12 "hf":    linf [0, 10, 101] hf freq
13 "hg":    linf [-5, 5, 101] hf gain
```

Even 88-Formant EQ

```
 0 "mdl":   E88
 1 "mix":   linf [0, 125, 126] %, mix
 2 "lf":    linf [0, 10, 101] lf freq
 3 "lg":    linf [-5, 5, 101] lf gain
 4 "lq":    str [LOW, HIGH] lf q
 5 "lt":    str [BELL, SHELV] lf type
 6 "lmf":   linf [0, 10, 101] lm freq
 7 "lmg":   linf [-5, 5, 101] lm gain
 8 "lmq":   linf [0, 10, 101] lm q
 9 "hmf":   linf [0, 10, 101] hm freq
10 "hmg":   linf [-5, 5, 101] hm gain
11 "hmq":   linf [0, 10, 101] hm q
12 "hf":    linf [0, 10, 101] hm freq
13 "hg":    linf [-5, 5, 101] hf gain
14 "hq":    str [LOW, HIG] hf q
15 "ht":    str [BELL, SHELV] hf type
```

Even 84 EQ

```
 0 "mdl":   E84
 1 "mix":   linf [0, 125, 126] %, mix
 2 "g":     linf [-20, 20, 81] dB, gain
 3 "lf":    str [OFF, 35, 60, 110, 220] lf freq
 4 "lg":    linf [-5, 5, 101] lf gain
 5 "mf":    str [OFF, 350, 700, 1k6, 3k2,
                 4k8, 7k2] mid freq
 6 "mg":    linf [-5, 5, 101] mid gain
 7 "mq":    str [LOW, HIGH] mid q
 8 "hf":    str [10k, 12k, 16k, OFF] hf freq
 9 "hg":    linf [-5, 5, 101] hf gain
```

Focusrite ISA 110 EQ

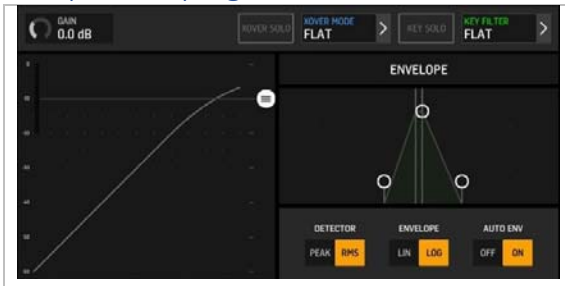
```
 0 "mdl":   F110
 1 "mix":   linf [0, 125, 126] %, mix
 2 "peq":   int [0, 1] peq on
 3 "lmf":   linf [0, 10, 101] lm freq
 4 "lmg":   linf [-5, 5, 101] lm gain
 5 "lmq":   linf [0, 10, 101] lm q
 6 "lmf3":  int [0, 1] lm /3
 7 "hmf":   linf [0, 10, 101] hm freq
 8 "hmg":   linf [-5, 5, 101] hm gain
 9 "hmq":   linf [0, 10, 101] hm q
10 "hmf3":  int [0, 1] hm x3
11 "shv":   inf [0, 1] shv on
12 "lf":    str [33, 56, 95, 160,
                270, 460] lf freq
13 "lg":    linf [-5, 5, 101] lf gain
14 "hf":    str [3k3, 4k7, 6k8, 10k,
                 15k, 18k] hf freq
15 "hg":    linf [-5, 5, 101] hf q
16 "g":     linf [-18, 18, 73] gain
```

Pulsar P1a/M5 EQ

```
 0 "mdl":   PULSAR
 1 "mix":   linf [0, 125, 126] %, mix
 2 "eq1":   int [0, 1] eq1 on
 3 "1lb":   linf [0, 10, 101] lf boost
 4 "1latt": linf [0, 10, 101] lf att
 5 "1lf":   str [20, 30, 60, 100] Hz, lf freq
 6 "1hw":   linf [0, 10, 101] hf wid
 7 "1hb":   linf [0, 10, 101] hf boost
 8 "1hf":   str [3k, 4k, , 5k, 8k, 10k,
                 12k, 16k] Hz, hf freq
 9 "1hatt": linf [0, 10, 101] hf att
10 "1hattf":str [5k, 10k, 20k] hf att
11 "eq5":   int [0, 1] eq5 on
12 "5lb":   linf [0, 10, 101] lm boost
13 "5lf":   str [200, 300, 500, 700,
                 1k] Hz, lf freq
```

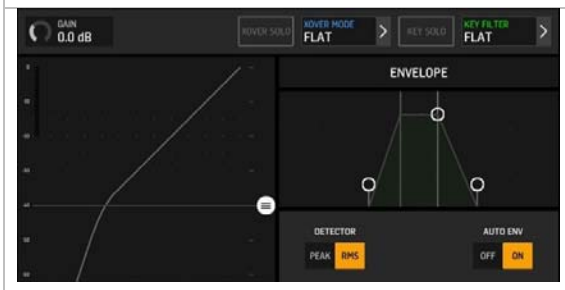
	*14 **"5md"**: linf [0, 10, 101] mid dip* *15 **"5mf"**: str [200, 300, 500, 700, 1k, 1k5, 2k, 3k, 4k, 5k, 7k] Hz, mid freq* *16 **"5hb"**: linf [0, 10, 101] HM boost* *17 **"5hf"**: str [1k5, 2k, 3k, 4k, 5k] Hz, hf freq*
	Mach EQ4 *0 **"mdl"**: MACH4* *1 **"mix"**: linf [0, 125, 126] %, mix* *2 **"sub"**: linf [-5, 5, 101] sub* *3 **"40"**: linf [-5, 5, 101] 40* *4 **"160"**: linf [-5, 5, 101] 160* *5 **"650"**: linf [-5, 5, 101] 650* *6 **"2k5"**: linf [-5, 5, 101] 2k5* *7 **"air"**: linf [0, 10, 101] air* *8 **"airm"**: str [OFF, 2k5, 5k, 10k, 20k, 40k] air mode* *9 **"again"**: int [0, 1] auto*

Compressor plugins



Standard compressor

```
 0 "mdl":   COMP
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-60, 0, 121] dB, thr
 4 "ratio": flt [1.1, 1.2, 1.3, 1.5, 1.7, 2.0,
                 2.5, 3.0, 3.5, 4.0, 5.0, 6.0,
                 8.0, 10., 20., 50., 100.] ratio
 5 "knee":  int [0…5] knee
 6 "det":   str [PEAK, RMS] detector
 7 "att":   linf [0, 120, 121] ms, attack
 8 "hld":   linf [1, 200, 200] ms, hold
 9 "rel":   logf [4, 4000, 130] ms release
10 "env":   str [LIN, LOG] envelope
11 "auto":  int [0, 1] auto
```



Standard expander

```
 0 "mdl":   EXP
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-60, 0, 121] dB, thr
 4 "ratio": flt [1.1, 1.2, 1.3, 1.5, 1.7, 2.0,
                 2.5, 3.0, 3.5, 4.0, 5.0, 6.0,
                 8.0, 10., 20., 50., 100.] ratio
 5 "knee":  int [0…5] knee
 6 "det":   str [PEAK, RMS] detector
 7 "att":   linf [0, 120, 121] ms, attack
 8 "hld":   linf [1, 200, 200] ms, hold
 9 "rel":   logf [4, 4000, 130] ms release
10 "env":   str [LIN, LOG] envelope
11 "auto":  int [0, 1] auto
```

BDX 160 Compressor/Limiter

```
 0 "mdl":   B160
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   logf [.01, 5, 65] thr
 4 "ratio": flt [1.1, 1.2, 1.3, 1.5, 1.7, 2.0,
                 2.5, 3.0, 3.5, 4.0, 5.0, 6.0,
                 8.0, 10., 20., 50.] ratio
```

BDX 560 Easy Compressor

```
 0 "mdl":   B560
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-40, 20, 121] dB, thr
 4 "ratio": flt [1.1, 1.2, 1.5, 2.0, 3.0, 4.0,
                 5.0, 7.0, 10., 50., 999.,
                 -5.0, -3.0, -2.0, -1.0] ratio
 5 "auto":  int [0, 1] auto
```

Draw More Compressor

```
 0 "mdl":   D241
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [0, -60, 121] dB, thr
 4 "ratio": flt [1.1, 1.2, 1.3, 1.5,
                 1.7, 2.0, 3.0, 3.5,
                 4.0, 5.0, 6.0, 8.0,
                 10.0, 20.0, 50.0,
                 100.0] ratio
 5 "att":   linf [.5, 100, 65] ms, attack
 6 "rel":   logf [50, 5000, 130] ms release
 7 "lim":   linf [-20, 0, 41] dB, lim thr
 8 "lrel":  logf [50, 5000, 130] ms, lim rel
 9 "auto":  int [0, 1] auto
```

Red Compressor

```
 0 "mdl":   RED3
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-48, 0, 97] dB, thr
 4 "ratio": flt [1.1, 1.2, 1.3, 1.5, 2.0,
                 2.5, 3.0, 3.5, 4.0, 5.0,
                 6.0, 8.0, 10.] ratio
 5 "att":   linf [1, 50, 65] ms, attack
 7 "rel":   logf [100, 4000, 65] ms release
 8 "auto":  int [0, 1] auto
```

Soul 9000 Channel Compressor

```
 0 "mdl":   9000C
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-48, 0, 97] dB, thr
 4 "ratio": flt [1.3, 1.43, 1.57, 1.8, 2.0,
                 2.8, 3.3, 4.0, 5.0 , 6.0,
                 7.0, 9.0, 12.0, 20.0, 50.0,
                 100.0] ratio
 5 "fast":  int [0, 1] fast att
 6 "rel":   logf [100, 4000, 65] ms release
 7 "peak":  int [0, 1] peak
```

Soul G Buss Compressor

```
 0 "mdl":   SBUS
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-48, 0, 81] dB, thr
 4 "ratio": flt [1.5, 2.0, 3.0, 4.0, 5.0,
                 10.0] ratio
 5 "att":   flt [0.1, 0.3, 1.0, 3.0, 10.0,
                 30.0] ratio
 6 "rel":   str [0.1, 0.2, 0.4, 0.8, 1.6,
                 AUTO] release
```

Even Compressor/Limiter

```
 0 "mdl":   ECL33
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "lon":   int [0, 1] lim on
 4 "lthr":  linf [-12, 0, 25] dB, lim thr
 5 "lrec":  str [50, 100, 200, 800,
                 A1, A2] lim rec
 6 "lfast": int [0, 1] lim fast
 7 "con":   int [0, 1] comp on
 8 "cthr":  linf [-35, -5, 61] dB, comp thr
 9 "ratio": str [1.5, 2.0. 3.0, 4.0, 6.0] ratio
10 "crec":  str [100, 400, 800, 1500
                 A1, A2] comp rec
11 "cfast": int [0, 1] comp fast
```

Eternal Bliss

```
 0 "mdl":   BLISS
 1 "mix":   linf [0, 100, 101] %, mix
 2 "gain":  linf [-6, 12, 37] dB, gain
 3 "thr":   linf [-50, 0, 101] dB, thr
 4 "ratio": flt [1.2, 1.3, 1.6, 2.0, 3.0,
                 -1.0, -2.0, -3.0,  -4.0] ratio
 5 "att":   linf [.4, 150, 65] ms, attack
 6 "rel":   logf [5, 1200, 65] ms release
 7 "afast": int [0, 1] auto fast
 8 "alog":  int [0, 1] anti log
 9 "glon":  int [0, 1] gr limit on
10 "glim":  linf [-21, 0, 43] gr limit
```

Amplifier76 Limiting Amplifier

```
0 "mdl":    76LA
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "in":     linf [-48, 0, 97] dB, input
4 "out":    linf [-48, 0, 97] dB
5 "att":    linf [1, 7, 61] attack
6 "rel":    linf [1, 7, 61] release
7 "ratio":  str [4, 8, 12, 20, ALL] ratio
```

Leveling Amplifier 2A

```
0 "mdl":    LA
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "ingain": linf [0, 100, 101] gain
4 "peak":   linf [0, 100, 101] peak
5 "mode":   str [comp, lim] mode
```

Fairkid Model 670

```
0 "mdl":    F670
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "in":     linf [-20, 0, 81] dB, input
4 "thr":    linf [0, 10, 41] thr
5 "time":   int [1…6] time
6 "bias":   linf [0, 1, 101] bias
```

No Stressor

```
0 "mdl":    NSTR
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "in":     linf [0, 10, 101] input
4 "ou":     linf [0, 10, 101] output
5 "att":    linf [0, 10, 101] attack
6 "rel":    linf [0, 10, 101] release
7 "ratio":  str [1.5:1, 2:1, 3:1, 4:1, 6:1,
                 10:1, 20:1, NUKE] ratio
```

PIA 2250

```
0 "mdl":    2250
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "thr":    linf [0, 10, 101] threashold
4 "ratio":  linf [0, 10, 101] output
5 "att":    str [FAST, MED, SLOW] attack
6 "rel":    logf [50, 3000, 130] ms, release
7 "knee":   str [HARD, SOFT] knee
8 "Type":   str [OLD, NEW] type
```

LTA100 Leveler

```
0 "mdl":    L100
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "ingain": linf [0, 10, 101] gain
4 "gr":     linf [0, 10, 101] gain reduction
5 "att":    str [FAST, MED, SLOW] attack
6 "rel":    str [FAST, MED, SLOW] release
```

Wave Designer

```
0 "mdl":    WAVE
1 "mix":    linf [0, 100, 101] %, mix
2 "gain":   linf [-6, 12, 37] dB, gain
3 "att":    linf [-15, 15, 61] dB, attack
4 "sust":   linf [-24, 24, 97] dB, sustain
5 "g":      linf [-16, 9, 55] dB, gain
```

Auto Rider Dynamics

```
0 "mdl":   RIDE
1 "mix":   linf [0, 100, 101] %, mix
2 "gain":  linf [-6, 12, 37] dB, gain
3 "thr":   linf [-54, 18, 73] dB, thr
4 "tgt":   linf [-48, 0, 97] dB, target
5 "spd":   int [1…50] speed
6 "ratio": flt [2.0, 4.0, 8.0,
                20.0, 100.0] ratio
7 "hld":   logf [.1, 10, 65] s, hold
8 "range": linf [1, 15, 29] dB, range
```

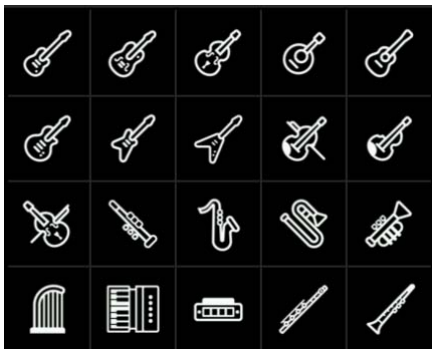
Appendix: WING Icons

The table below gives the list of icons available with WING. Icon number ranges are listed to the right of the icons.

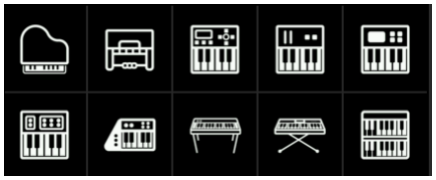
General:
[0…14]

Vocals and Mics:
[100…114]

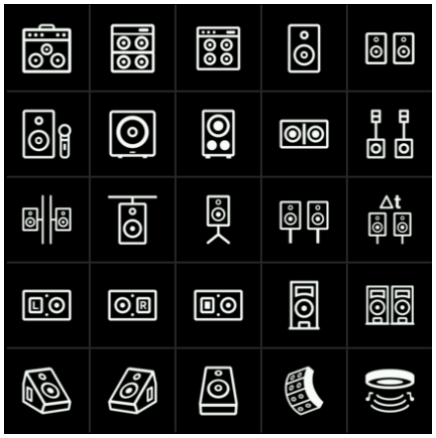
Drums and Percussions:
[200…224]



Strings and Winds:
[300…319]



Keys:
[400…409]

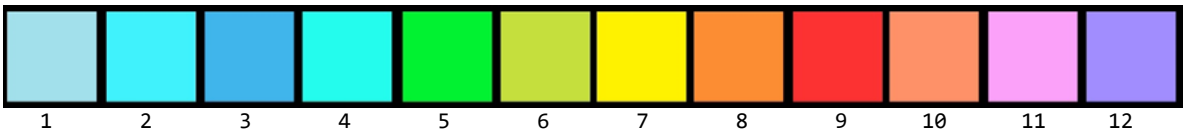


Speakers:
[500…524]

Specials:
[600…614]

Appendix: WING Colors

WING colors are used in several areas such as channel strip color, scribble color, etc. The known colors are shown below and indexed as values 1 to 12:

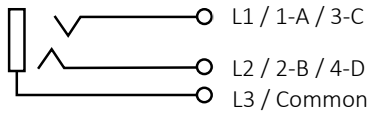


1	gray blue
2	medium blue
3	dark blue
4	turquoise
5	green
6	olive green
7	yellow
8	orange
9	red
10	coral
11	pink
12	mauve

Appendix: WING GPIOs:

The WING digital mixing console is offering 4 GPIOs (General Purpose Input/Output) which can be very useful in the studio or live situations. This paragraph shows how to use them in different modes. Let's look at what GPIOs can offer.

At the rear of the console, two TRS jack sockets provide connections to 4 GPIOS. Each of the TRS sockets is depicted below. Lug L3 is common to the 2 GPIOs supported by each socket. Lugs L1 and L2 are respectively used for GPIO 1 or A, 2 or B or 3 or C, 4 or D, depending on the socket used.



WING GPIO 'mode' settings can be any of the following: `TGLNO, TGLNC, INNO, INNC, OUTNO, OUTNC`. These are represented by OSC patterns `/$ctl/gpio/1..4/mode`, and correspond to:

`TGLNO`	Toggle, Normally Opened
`TGLNC`	Toggle, Normally Closed
`INNO`	Input, Normally Opened
`INNC`	Input, Normally Closed
`OUTNO`	Output, Normally Opened
`OUTNC`	Output, Normally Closed

WING GPIO 'state' values can be 0 for `open/OFF` (light off), or 1 for `close/ON` (light on). These correspond to OSC patterns `/$ctl/gpio/1..4/gpstate`.

Electrical connections:

- `INNO / INNC`: The console provides approx. 5V between A/B/C/D and Common. The application of a short, dropping voltage to 0V will change the state of the respective GPIO between `open` and `close`, depending on the `NO/NC` mode.

- `OUTNO / OUTNC`: The console provides approx. 5V between A/B/C/D and Common; The voltage presented by the console goes from near 5V to 0V depending on the state (`open` or `close`) and the `NO/NC` mode of the respective GPIO.

- `TGLNO / TGLNC`: This is to toggle the internal state of the GPIO. The console provides approx.. 5V between A/B/C/D and Common; changing the state of the respective GPIO does not change the voltage provided by the console.

Appendix: MCU [DAW BUTTONS] commands list

OSC	MCU action	MIDI (port 4)		OSC	MCU action	MIDI (port 4)
T1	STOP	90, 5D, 7F/00		V7	BUSES (VIEW)	90, 43, 7F/00
T2	PLAY	90, 5E, 7F/00		V8	OUTPUTS (VIEW)	90, 44, 7F/00
T3	RECORD	90, 5F, 7F/00		V9	USER (VIEW)	90, 45, 7F/00
T4	REWIND	90, 5B, 7F/00		V10	MIX (VIEW)	
T5	FAST FWD	90, 5C, 7F/00		V11	EDIT (VIEW)	
T6	MARKER	90, 54, 7F/00		V12	TRANSPORT (VIEW)	
T7	NUDGE	90, 55, 7F/00		V13	MEM/LOC (VIEW)	
T8	CYCLE	90, 56, 7F/00		V14	STATUS (VIEW)	
T9	DROP	90, 57, 7F/00		V15	ALT (VIEW)	
T10	REPLACE	90, 58, 7F/00		AU1	READ/OFF (AUTOM)	90, 4A, 7F/00
T11	SCRUB	90, 65, 7F/00		AU2	WRITE (AUTOM)	90, 4B, 7F/00
T12	SHUTTLE			AU3	TRIM (AUTOM)	90, 4C, 7F/00
T13	RETURN TO ZERO			AU4	TOUCH (AUTOM)	90, 4D, 7F/00
T14	GO TO END			AU5	LATCH (AUTOM)	90, 4E, 7F/00
T15	IN			AU6	OFF (AUTOM)	
T16	OUT			AU7	FADER (AUTOM)	
T17	PRE			AU8	PAN (AUTOM)	
T18	POST			AU9	MUTE (AUTOM)	
T19	ONLINE			AU10	SEND (AUTOM)	
T20	QUICK PUNCH			AU11	SEND MUTE (AUTOM)	
N1	UP (NAV)	90, 60, 7F/00		AU12	PLUG-IN (AUTOM)	
N2	DOWN (NAV)	90, 61, 7F/00		SY1	SHIFT	90, 46, 7F/00
N3	LEFT (NAV)	90, 62, 7F/00		SY2	OPTION	90, 47, 7F/00
N4	RIGHT (NAV)	90, 63, 7F/00		SY3	CTRL	90, 48, 7F/00
N5	ZOOM	90, 64, 7F/00		SY4	ALT	90, 49, 7F/00
N6	BK <	90, 2E, 7F/00		SY5	SAVE	90, 50, 7F/00
N7	BK >	90, 2F, 7F/00		SY6	UNDO	90, 51, 7F/00
N8	CH <	90, 30, 7F/00		SY7	CANCEL	90, 52, 7F/00
N9	CH >	90, 31, 7F/00		SY8	ENTER	90, 53, 7F/00
A1	TRACK (ASSIGN)	90, 28, 7F/00		SY9	EDIT MODE	
A2	SEND (ASSIGN)	90, 29, 7F/00		SY10	EDIT TOOL	
A3	PAN (ASSIGN)	90, 2A, 7F/00		OT1	FLIP	90, 32, 7F/00
A4	PLUG-IN (ASSIGN)	90, 2B, 7F/00		OT2	GROUP	90, 4F, 7F/00
A5	EQ (ASSIGN)	90, 2C, 7F/00		OT3	NAME/VALUE	90, 34, 7F/00
A6	INST (ASSIGN)	90, 2D, 7F/00		OT4	TIME/BEATS	90, 35, 7F/00
A7	SEND A (ASSIGN)			OT5	CLICK	90, 59, 7F/00
A8	SEND B (ASSIGN)			OT6	SOLO	90, 5A, 7F/00
A9	SEND C (ASSIGN)			OT7	FOOTSW A	90, 66, 7F/00
A10	SEND D (ASSIGN)			OT8	FOOTSW B	90, 67, 7F/00
A11	SEND E (ASSIGN)			OT9	DEFAULT	
A12	INPUT (ASSIGN)			OT10	SUSPEND	
A13	OUTPUT (ASSIGN)			OT11	BYPASS	
A14	ASSIGN (ASSIGN)			OT12	RECRDY ALL	
A15	SHIFT (ASSIGN)			E1	CUT (EDIT)	
A16	MUTE (ASSIGN)			E2	COPY (EDIT)	
F1	F1	90, 36, 7F/00		E3	PASTE (EDIT)	
F2	F2	90, 37, 7F/00		E4	SEPARATE (EDIT)	
F3	F3	90, 38, 7F/00		E5	CAPTURE (EDIT)	
F4	F4	90, 39, 7F/00		E6	DELETE (EDIT)	
F5	F5	90, 3A, 7F/00		E7	ASSIGN (EDIT)	
F6	F6	90, 3B, 7F/00		E8	COMPARE (EDIT)	
F7	F7	90, 3C, 7F/00		E9	BYPASS (EDIT)	
F8	F8	90, 3D, 7F/00		E10	INS/PARAM (EDIT)	
V1	GLOBAL (VIEW)	90, 33, 7F/00		SP1	FADER TOUCH [MUTE]	
V2	MIDI (VIEW)	90, 3E, 7F/00		SP2	V-POT CTRL [SEL/SOLO]	
V3	INPUTS (VIEW)	90, 3F, 7F/00		SP3	RECRDY CTRL [SEL]	
V4	AUDIO TRACKS (VIEW)	90, 40, 7F/00		SP4	AUTO [SEL]	
V5	INSTRUMENT (VIEW)	90, 41, 7F/00		SP5	V-SEL [SEL]	
V6	AUX (VIEW)	90, 42, 7F/00		SP6	INSERT [SEL]	

Appendix: MCU [DAW V-POTS] commands list

OSC	MCU action	MIDI (port 4)		OSC	MCU action	MIDI (port 4)
M1P	V-POT M1 Push	90, 20, 7F/00		M1	V-POT M1	B0, 10, 01/41
M2P	V-POT M2 Push	90, 21, 7F/00		M2	V-POT M2	B0, 11, 01/41
M3P	V-POT M3 Push	90, 22, 7F/00		M3	V-POT M3	B0, 12, 01/41
M4P	V-POT M4 Push	90, 23, 7F/00		M4	V-POT M4	B0, 13, 01/41
M5P	V-POT M5 Push	90, 24, 7F/00		M5	V-POT M5	B0, 14, 01/41
M6P	V-POT M6 Push	90, 25, 7F/00		M6	V-POT M6	B0, 15, 01/41
M7P	V-POT M7 Push	90, 26, 7F/00		M7	V-POT M7	B0, 16, 01/41
M8P	V-POT M8 Push	90, 27, 7F/00		M8	V-POT M8	B0, 17, 01/41
E1P	V-POT EXT1 Push			E1	V-POT EXT1	
E2P	V-POT EXT2 Push			E2	V-POT EXT2	
E3P	V-POT EXT3 Push			E3	V-POT EXT3	
E4P	V-POT EXT4 Push			E4	V-POT EXT4	
E5P	V-POT EXT5 Push			E5	V-POT EXT5	
E6P	V-POT EXT6 Push			E6	V-POT EXT6	
E7P	V-POT EXT7 Push			E7	V-POT EXT7	
E8P	V-POT EXT8 Push			E8	V-POT EXT8	
E9P	V-POT EXT9 Push			E9	V-POT EXT9	
E10P	V-POT EXT10 Push			E10	V-POT EXT10	
E11P	V-POT EXT11 Push			E11	V-POT EXT11	
E12P	V-POT EXT12 Push			E12	V-POT EXT12	
E13P	V-POT EXT13 Push			E13	V-POT EXT13	
E14P	V-POT EXT14 Push			E14	V-POT EXT14	
E15P	V-POT EXT15 Push			E15	V-POT EXT15	
E16P	V-POT EXT16 Push			E16	V-POT EXT16	
				JOG	JOG WHEEL	B0, 3C, 01/41

Appendix: MCU [DAW REMOTE MCU] commands list

OSC	MCU action	MIDI (port 4)
M1	V-POT M1	B0, 10, 01/41
M2	V-POT M2	B0, 11, 01/41
M3	V-POT M3	B0, 12, 01/41
M4	V-POT M4	B0, 13, 01/41
M5	V-POT M5	B0, 14, 01/41
M6	V-POT M6	B0, 15, 01/41
M7	V-POT M7	B0, 16, 01/41
M8	V-POT M8	B0, 17, 01/41
E1	V-POT EXT1	
E2	V-POT EXT2	
E3	V-POT EXT3	
E4	V-POT EXT4	
E5	V-POT EXT5	
E6	V-POT EXT6	
E7	V-POT EXT7	
E8	V-POT EXT8	
E9	V-POT EXT9	
E10	V-POT EXT10	
E11	V-POT EXT11	
E12	V-POT EXT12	
E13	V-POT EXT13	
E14	V-POT EXT14	
E15	V-POT EXT15	
E16	V-POT EXT16	
JOG	JOG WHEEL	B0, 3C, 01/41

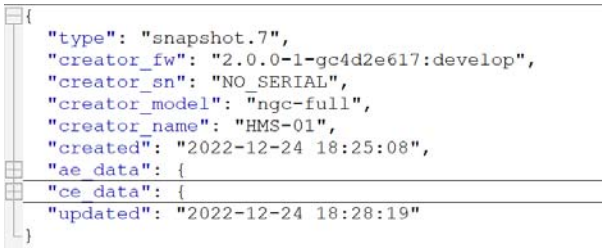
Appendix: WING Snapshot and JSON Data Structure:

A WING snapshot (also called Snapfile when saved to a file) is organized as a collection of classes, sub-classes and objects regrouping attributes and values in logical groups. These can be represented as a hierarchical tree. A JSON[25] notation is used to describe and store the hierarchical tree.
A complete WING snapfile is close to 460000 bytes and 28800 lines, containing a rather complex hierarchical list of object identifiers and their associated values.
A WING snapfile does not contain read-only objects; i.e. there are more elements available than the one saved in a snapfile!

Global Snapfile

A snapfile is divided in either:

3 sections: `description`, `ae_data` and `ce_data`, as shown below:



4 sections: `description`, `ae_data` and `ce_data`, `scopes`, as shown below:

Description

description: This small section contains (as its name suggests) a description for the snapshot, including name, and elements corresponding to the WING that generated the snapshot. “created” lists the date and time of the creation of the snapfile, while “updated” will retain the date and time of the most recent update made to the file.

```
“type”: string,
“creator_fw”: string,
“creator_sn”: string,
“creator_model”: string,
“creator_name”: string,
“created”: date time,
“updated”: date time,
```

[25] JavaScript Object Notation: an efficient way to represent structured objects. Also used as a data-interchange format.

scopes

scopes: A large set of *BooLean* {*'+',' '*} values to list what has been 'marked' at snapshot time. This can be used as a reminder of the initial purpose of the snapshot.
The scopes class contains the following objects:

ch, aux, bus, main, mtx, dca, mute, fx, source, output, area, custom, setup, mainflt, sendflt;

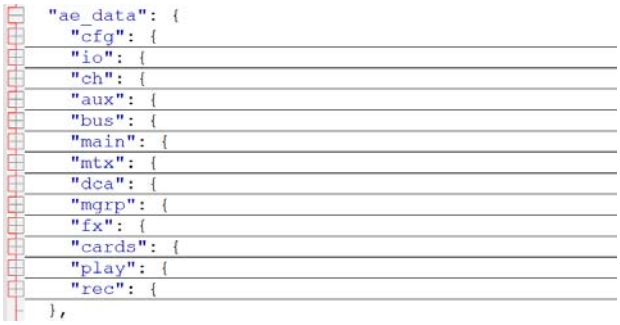
For example:

```
"scopes": {
  "ch": "++++++++++++++++++++++++++++++++++++++++",
  "aux": "++++++++",
  "bus": "++++++++++++++++",
  "main": "++++",
  "mtx": "++++++++",
  "dca": "++++++++++++++++",
  "mute": "++++++++",
  "fx": "++++++++++++++++",
  "source": {
    "LCL": "++++++++",
    "AUX": "++++++++",
    "A": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "B": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "C": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "SC": "++++++++++++++++++++++++++++++++",
    "USB": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "CRD": "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++",
    "MOD": "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++",
    "PLAY": "++++",
    "AES": "++",
    "USR": "++++++++++++++++++++++++",
    "OSC": "++"
  },
  "output": {
    "LCL": "++++++++",
    "AUX": "++++++++",
    "A": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "B": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "C": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "SC": "++++++++++++++++++++++++++++++++++",
    "USB": "++++++++++++++++++++++++++++++++++++++++++++++++++",
    "CRD": "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++",
    "MOD": "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++",
    "REC": "++++",
    "AES": "++"
  },
  "area": {
    "L": "+++++++",
    "C": "++++++",
    "R": "+++++++"
  },
  "custom": "++++++++++++++++++++++",
  "setup": "+++",
  "filters": "               ",
  "mainflt": "    ",
  "sendflt": "                         "
}
```

Scopes are not elements that can be programmatically changed. They are only set at snapshot time using the console main LCD. As mentioned above, they are optionally saved at save time to notify what was targeted for save/update.

ae_data

ae_data stands for "Audio Engine", and regroups a rather large set of attributes and values aimed at registering all main settings of the WING audio engine, such as Routing, Channel EQ settings, FX parameter values, etc., as shown in the figure below:



In the next pages, we present the structure, 1 block of parameters at a time. Understanding what parameters are present in each block is a good way to better grasp and understand the vast range of capabilities WING offers. It is also a good way to envision the parameter list one can get and set using **wapi** (described later in the wapi documentation) as the JSON structure parameters matches the tokens used by the API for **wapi** `get()` and `set()` functions.

Indeed, all tokens related to the audio engine can be directly coded from the JSON description, for example, the C-like token notation for the JSON *cfg.mon.1.pan* element is named `CFG_MON_1_PAN`.

We show in the following pages, the contents of the JSON tree structure after a console reset, so default values are listed. In order to reduce the number of pages the JSON structure description would take; the following notation is used:

"abc": {}, means that *"abc"* uses the same structure definition as the previous member in the JSON file, and:

"2":{}…"n": {}, means that objects *"2"* to *"n"* use the same structure definition as the previous member in the JSON file.

The ae_data class contains the following objects: ***cfg, io, ch, aux, bus, main, mtx, dca, mgrp, fx, cards, play, rec***, shown in the following pages using the notation conventions above.

```
"ae_data": {
        "cfg": {
            "clkrate": 48000,
            "clksrc": "INT",
            "mainlink": "OFF",
            "dcamgrp": true,
            "muteovr": true,
            "startmute": false,
            "usbacfg": "48/48",
            "sccfg": "AUTO",
            "mon": {
                "1": {
                    "inv": false,
                    "pan": 0,
                    "wid": 100,
                    "eq": {
                        "on": false,
```

```
                "lsg": 0,
                "lsf": 60.13884,
                "1g": 0,
                "1f": 129.8763,
                "1q": 1.995882,
                "2g": 0,
                "2f": 299.2472,
                "2q": 1.995882,
                "3g": 0,
                "3f": 699.4875,
                "3q": 1.995882,
                "4g": 0,
                "4f": 1499.788,
                "4q": 1.995882,
                "5g": 0,
                "5f": 2992.471,
                "5q": 1.995882,
                "6g": 0,
                "6f": 6013.884,
                "6q": 1.995882,
                "hsg": 0,
                "hsf": 11994.42
            },
            "lim": 0,
            "dly": {
                "on": false,
                "m": 0.1
            },
            "dim": 20,
            "pfldim": 12,
            "eqbdtrim": 0,
            "srclvl": 0,
            "srcmix": -144,
            "src": "MAIN.1",
            "tags": ""
        },
        "2": {}
    },
    "solo": {
        "mode": "LIVE",
        "mon": "A",
        "mute": false,
        "chtap": "PFL",
        "bustap": "AFL",
        "maintap": "PFL",
        "mtxtap": "PFL",
        "srcsolo": "OFF"
    },
    "rta": {
        "rtasrc": 0,
        "rtatap": "IN",
        "rtadecay": "MED",
        "rtadet": "PEAK",
        "rtarange": 30,
        "rtagain": 0,
        "rtaauto": true
    },
    "mtr": {
        "scopesrc": 0,
        "scopetap": "IN"
    },
    "talk": {
        "assign": "OFF",
        "indiv": false,
        "A": {
            "mode": "AUTO",
            "mondim": false,
            "busdim": 0,
            "B1": false,
            "B2": false,
            "B3": false,
            "B4": false,
            "B5": false,
            "B6": false,
            "B7": false,
            "B8": false,
            "B9": false,
```

```
                "B10": false,
                "B11": false,
                "B12": false,
                "B13": false,
                "B14": false,
                "B15": false,
                "B16": false,
                "M1": false,
                "M2": false,
                "M3": false,
                "M4": false
            },
            "B": {}
        },
        "harmt": {
            "a": false,
            "b": false,
            "c": false
        },
        "custsync": {
            "a": false,
            "b": false,
            "c": false
        },
        "amix": {
            "x": true,
            "y": true
        }
    },
    "io": {
        "altsw": false,
        "in": {
            "LCL": {
                "1": {
                    "mode": "M",
                    "g": 0,
                    "vph": false,
                    "mute": false,
                    "pol": false,
                    "col": 1,
                    "name": "",
                    "icon": 1,
                    "tags": "",
                    "rmt": "OFF",
                    "rcvc": false
                },
                "2": {},..."8": {}
            },
            "AUX": {
                "1": {
                    "mode": "M",
                    "mute": false,
                    "pol": false,
                    "col": 1,
                    "name": "",
                    "icon": 2,
                    "tags": ""
                },
                "2": {},..."8": {}
            },
            "A": {
                "1": {
                    "mode": "M",
                    "g": 0,
                    "vph": false,
                    "mute": false,
                    "pol": false,
                    "col": 1,
                    "name": "",
                    "icon": 0,
                    "tags": "",
                    "rmt": "OFF",
                    "rcvc": false
                },
                "2": {},..."48": {}
            },
            "B": {},
```

```
"C": {},
"SC": {
    "1": {
        "mode": "M",
        "mute": false,
        "pol": false,
        "col": 1,
        "name": "",
        "icon": 0,
        "tags": ""
    },
    "2": {},..."32": {}
},
"USB": {
    "1": {
        "mode": "ST",
        "mute": false,
        "pol": false,
        "col": 8,
        "name": "USB 1/2",
        "icon": 605,
        "tags": ""
    },
    "2": {},..."48": {}
},
"CRD": {
    "1": {
        "mode": "M",
        "mute": false,
        "pol": false,
        "col": 1,
        "name": "",
        "icon": 0,
        "tags": ""
    },
    "2": {},..."64": {}
},
"MOD": {
    "1": {
        "mode": "M",
        "mute": false,
        "pol": false,
        "col": 1,
        "name": "",
        "icon": 0,
        "tags": ""
    },
    "2": {},..."64": {}
},
"PLAY": {
    "1": {
        "mode": "ST",
        "mute": false,
        "pol": false,
        "col": 8,
        "name": "2TR",
        "icon": 608,
        "tags": ""
    },
    "2": {},..."4": {}
},
"AES": {
    "1": {
        "mode": "M",
        "mute": false,
        "pol": false,
        "col": 1,
        "name": "",
        "icon": 0,
        "tags": ""
    },
    "2": {}
},
"USR": {
    "1": {
        "mode": "M",
        "mute": false,
```

```
                "pol": false,
                "col": 1,
                "name": "",
                "icon": 0,
                "tags": "",
                "user": {
                    "grp": "OFF",
                    "in": 1,
                    "tap": "PRE",
                    "lr": "L+R"
                }
            },
            "2": {},..."24": {}
        },
        "OSC": {
            "1": {
                "mode": "M",
                "mute": false,
                "col": 1,
                "name": "",
                "icon": 0,
                "tags": "",
                "osc": {
                    "lvl": -6,
                    "mode": "SINE",
                    "f": 999.992
                }
            },
            "2": {}
            }
        }
    },
    "out": {
        "LCL": {
            "1": {
                "grp": "BUS",
                "in": 1
            },
            "2": {},..."8": {}
        },
        "AUX": {
            "1": {
                "grp": "OFF",
                "in": 1
            },
            "2": {},..."8": {}
        },
        "A": {
            "1": {
                "grp": "OFF",
                "in": 1
            },
            "2": {},..."48": {}
        },
        "B": {},
        "C": {},
        "SC": {
            "1": {
                "grp": "OFF",
                "in": 1
            },
            "2": {},..."32": {}
        },
        "USB": {
            "1": {
                "grp": "OFF",
                "in": 1
            },
            "2": {},..."48": {}
        },
        "CRD": {
            "1": {
                "grp": "OFF",
                "in": 1
            },
            "2": {},..."64": {}
        },
```

```
            "MOD": {
                "1": {
                    "grp": "OFF",
                    "in": 1
                },
                "2": {},..."64": {}
            },
            "REC": {
                "1": {
                    "grp": "OFF",
                    "in": 1
                },
                "2": {},..."4": {}
            },
            "AES": {
                "1": {
                    "grp": "OFF",
                    "in": 1
                },
                "2": {}
            }
        }
    },
    "ch": {
        "1": {
            "in": {
                "set": {
                    "srcauto": false,
                    "altsrc": false,
                    "inv": false,
                    "trim": 0,
                    "bal": 0,
                    "dlymode": "M",
                    "dly": 0,
                    "dlyon": false
                },
                "conn": {
                    "grp": "LCL",
                    "in": 1,
                    "altgrp": "OFF",
                    "altin": 1
                }
            },
            "flt": {
                "lc": false,
                "lcf": 100.2375,
                "hc": false,
                "hcf": 10018.26,
                "tf": false,
                "mdl": "TILT",
                "tilt": 0
            },
            "clink": true,
            "col": 1,
            "name": "",
            "icon": 1,
            "led": true,
            "mute": false,
            "fdr": -144,
            "pan": 0,
            "wid": 100,
            "solosafe": false,
            "mon": "A",
            "proc": "GEDI",
            "ptap": "5",
            "peq": {
                "on": false,
                "1g": 0,
                "1f": 99.68543,
                "1q": 0.99797,
                "2g": 0,
                "2f": 999.2505,
                "2q": 0.99797,
                "3g": 0,
                "3f": 10016.53,
                "3q": 0.99797
            },
```

```
"gate": {
    "on": false,
    "mdl": "GATE",
    "thr": -40,
    "range": 40,
    "att": 10,
    "hld": 10,
    "rel": 199.4043,
    "acc": 0,
    "ratio": "1:3"
},
"gatesc": {
    "type": "OFF",
    "f": 1002.374,
    "q": 1.995882,
    "src": "SELF",
    "tap": "IN"
},
"eq": {
    "on": false,
    "mdl": "STD",
    "mix": 100,
    "lg": 0,
    "lf": 80.19642,
    "lq": 0.99797,
    "leq": "SHV",
    "1g": 0,
    "1f": 200,
    "1q": 0.99797,
    "2g": 0,
    "2f": 601.3884,
    "2q": 0.99797,
    "3g": 0,
    "3f": 1499.788,
    "3q": 0.99797,
    "4g": 0,
    "4f": 3990.524,
    "4q": 0.99797,
    "hg": 0,
    "hf": 11994.42,
    "hq": 0.99797,
    "heq": "SHV"
},
"dyn": {
    "on": false,
    "mdl": "COMP",
    "mix": 100,
    "gain": 0,
    "thr": -10,
    "ratio": 3,
    "knee": 3,
    "det": "RMS",
    "att": 50,
    "hld": 20,
    "rel": 152.5652,
    "env": "LOG",
    "auto": true
},
"dynxo": {
    "depth": 6,
    "type": "OFF",
    "f": 1002.374
},
"dynsc": {
    "type": "OFF",
    "f": 1002.374,
    "q": 1.995882,
    "src": "SELF",
    "tap": "IN"
},
"preins": {
    "on": false,
    "ins": "NONE"
},
"main": {
    "1": {
        "on": true,
```

```
                "lvl": 0,
                "pre": false
            },
            "2": {},..."4": {}
        },
        "send": {
            "1": {
                "on": false,
                "lvl": -144,
                "pon": false,
                "ind": false,
                "mode": "PRE",
                "plink": false,
                "pan": 0,
                "wid": 100
            },
            "2": {},..."16": {}
        },
        "postins": {
            "on": false,
            "mode": "FX",
            "ins": "NONE",
            "w": 0
        },
        "tags": ""
    },
    "2": {},..."40": {}
},
"aux": {
    "1": {
        "in": {
            "set": {
                "srcauto": false,
                "altsrc": false,
                "inv": false,
                "trim": 0,
                "bal": 0
            },
            "conn": {
                "grp": "USB",
                "in": 1,
                "altgrp": "OFF",
                "altin": 1
            }
        },
        "clink": true,
        "col": 8,
        "name": "USB 1/2",
        "icon": 605,
        "led": true,
        "mute": false,
        "fdr": -144,
        "pan": 0,
        "wid": 100,
        "solosafe": false,
        "mon": "A",
        "eq": {
            "on": false,
            "mdl": "STD",
            "mix": 100,
            "lg": 0,
            "lf": 80.19642,
            "lq": 0.99797,
            "leq": "SHV",
            "1g": 0,
            "1f": 200,
            "1q": 0.99797,
            "2g": 0,
            "2f": 601.3884,
            "2q": 0.99797,
            "3g": 0,
            "3f": 1499.788,
            "3q": 0.99797,
            "4g": 0,
            "4f": 3990.524,
            "4q": 0.99797,
            "hg": 0,
```

```
                "hf": 11994.42,
                "hq": 0.99797,
                "heq": "SHV"
            },
            "dyn": {
                "on": false,
                "thr": -36,
                "depth": 12,
                "fast": false,
                "peak": false,
                "ingain": 40,
                "cpeak": 0,
                "cmode": "COMP"
            },
            "preins": {
                "on": false,
                "ins": "NONE"
            },
            "main": {
                "1": {
                    "on": true,
                    "lvl": 0,
                    "pre": false
                },
                "2": {},..."4": {}
            },
            "send": {
                "1": {
                    "on": false,
                    "lvl": -144,
                    "pon": false,
                    "ind": false,
                    "mode": "PRE",
                    "plink": false,
                    "pan": 0,
                    "wid": 100
                },
                "2": {},..."16": {}
            },
            "tags": ""
        },
        "2": {},..."8": {}
    },
    "bus": {
        "1": {
            "in": {
                "set": {
                    "inv": false,
                    "trim": 0,
                    "bal": 0
                }
            },
            "col": 1,
            "name": "",
            "icon": 0,
            "led": true,
            "busmono": false,
            "mute": false,
            "fdr": -144,
            "pan": 0,
            "wid": 100,
            "mon": "A",
            "busmode": "PRE",
            "eq": {
                "on": false,
                "mdl": "STD",
                "mix": 100,
                "lg": 0,
                "lf": 60.13884,
                "lq": 0.99797,
                "leq": "SHV",
                "1g": 0,
                "1f": 129.8763,
                "1q": 0.99797,
                "2g": 0,
                "2f": 299.2472,
                "2q": 0.99797,
```

```
            "3g": 0,
            "3f": 699.4875,
            "3q": 0.99797,
            "4g": 0,
            "4f": 1499.788,
            "4q": 0.99797,
            "5g": 0,
            "5f": 2992.471,
            "5q": 0.99797,
            "6g": 0,
            "6f": 6013.884,
            "6q": 0.99797,
            "hg": 0,
            "hf": 11994.42,
            "hq": 0.99797,
            "heq": "SHV",
            "tilt": 0
        },
        "dyn": {
            "on": false,
            "mdl": "COMP",
            "mix": 100,
            "gain": 0,
            "thr": -10,
            "ratio": 3,
            "knee": 3,
            "det": "RMS",
            "att": 50,
            "hld": 20,
            "rel": 152.5652,
            "env": "LOG",
            "auto": true
        },
        "dynxo": {
            "depth": 6,
            "type": "OFF",
            "f": 1002.374
        },
        "dynsc": {
            "type": "OFF",
            "f": 1002.374,
            "q": 1.995882,
            "src": "SELF",
            "tap": "BUS"
        },
        "preins": {
            "on": false,
            "ins": "NONE"
        },
        "main": {
            "1": {
                "on": false,
                "lvl": 0,
                "pre": false
            },
            "2": {},..."4": {}
        },
        "send": {
            "1": {
                "on": false,
                "lvl": -144,
                "pre": false
            },
            "2": {},..."8": {},
            "MX1": {
                "on": false,
                "lvl": -144,
                "pre": false
            },
            "MX2": {},..."MX8": {}
        },
        "postins": {
            "on": false,
            "ins": "NONE"
        },
        "tags": ""
    },
```

```
    "2": {},..."16": {}
},
"main": {
    "1": {
        "in": {
            "set": {
                "inv": false,
                "trim": 0,
                "bal": 0
            }
        },
        "col": 1,
        "name": "",
        "icon": 0,
        "led": true,
        "busmono": false,
        "mute": false,
        "fdr": -144,
        "pan": 0,
        "wid": 100,
        "mon": "A",
        "eq": {
            "on": false,
            "mdl": "STD",
            "mix": 100,
            "lg": 0,
            "lf": 60.13884,
            "lq": 0.99797,
            "leq": "SHV",
            "1g": 0,
            "1f": 129.8763,
            "1q": 0.99797,
            "2g": 0,
            "2f": 299.2472,
            "2q": 0.99797,
            "3g": 0,
            "3f": 699.4875,
            "3q": 0.99797,
            "4g": 0,
            "4f": 1499.788,
            "4q": 0.99797,
            "5g": 0,
            "5f": 2992.471,
            "5q": 0.99797,
            "6g": 0,
            "6f": 6013.884,
            "6q": 0.99797,
            "hg": 0,
            "hf": 11994.42,
            "hq": 0.99797,
            "heq": "SHV",
            "tilt": 0
        },
        "dyn": {
            "on": false,
            "mdl": "COMP",
            "mix": 100,
            "gain": 0,
            "thr": -10,
            "ratio": 3,
            "knee": 3,
            "det": "RMS",
            "att": 50,
            "hld": 20,
            "rel": 152.5652,
            "env": "LOG",
            "auto": true
        },
        "dynxo": {
            "depth": 6,
            "type": "OFF",
            "f": 1002.374
        },
        "dynsc": {
            "type": "OFF",
            "f": 1002.374,
            "q": 1.995882,
```

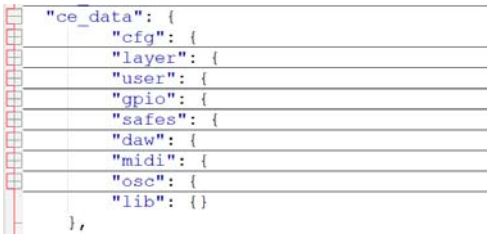
```
            "src": "SELF",
            "tap": "BUS"
        },
        "preins": {
            "on": false,
            "ins": "NONE"
        },
        "send": {
            "MX1": {
                "on": false,
                "lvl": -144,
                "pre": false
            },
            "MX2": {},..."MX8": {}
        },
        "postins": {
            "on": false,
            "ins": "NONE"
        },
        "dly": {
            "on": false,
            "m": 0.1
        },
        "tags": ""
    },
    "2": {},..."4": {}
},
"mtx": {
    "1": {
        "in": {
            "set": {
                "inv": false,
                "trim": 0,
                "bal": 0
            }
        },
        "dir": {
            "1": {
                "on": false,
                "lvl": -144,
                "inv": false,
                "in": "OFF",
                "tap": "PRE"
            },
            "2": {}
        },
        "col": 1,
        "name": "",
        "icon": 0,
        "led": true,
        "busmono": false,
        "mute": false,
        "fdr": -144,
        "pan": 0,
        "wid": 100,
        "mon": "A",
        "eq": {
            "on": false,
            "mdl": "STD",
            "mix": 100,
            "lg": 0,
            "lf": 60.13884,
            "lq": 0.99797,
            "leq": "SHV",
            "1g": 0,
            "1f": 129.8763,
            "1q": 0.99797,
            "2g": 0,
            "2f": 299.2472,
            "2q": 0.99797,
            "3g": 0,
            "3f": 699.4875,
            "3q": 0.99797,
            "4g": 0,
            "4f": 1499.788,
            "4q": 0.99797,
            "5g": 0,
```

```
            "5f": 2992.471,
            "5q": 0.99797,
            "6g": 0,
            "6f": 6013.884,
            "6q": 0.99797,
            "hg": 0,
            "hf": 11994.42,
            "hq": 0.99797,
            "heq": "SHV",
            "tilt": 0
        },
        "dyn": {
            "on": false,
            "mdl": "COMP",
            "mix": 100,
            "gain": 0,
            "thr": -10,
            "ratio": 3,
            "knee": 3,
            "det": "RMS",
            "att": 50,
            "hld": 20,
            "rel": 152.5652,
            "env": "LOG",
            "auto": true
        },
        "dynxo": {
            "depth": 6,
            "type": "OFF",
            "f": 1002.374
        },
        "dynsc": {
            "type": "OFF",
            "f": 1002.374,
            "q": 1.995882,
            "src": "SELF",
            "tap": "BUS"
        },
        "preins": {
            "on": false,
            "ins": "NONE"
        },
        "postins": {
            "on": false,
            "ins": "NONE"
        },
        "dly": {
            "on": false,
            "m": 0.1
        },
        "tags": ""
    },
    "2": {},..."8": {}
},
"dca": {
    "1": {
        "name": "",
        "col": 1,
        "icon": 0,
        "led": false,
        "mute": false,
        "fdr": -144,
        "mon": "A"
    },
    "2": {},..."16": {}
},
"mgrp": {
    "1": {
        "name": "MGRP.1",
        "mute": false
    },
    "2": {},..."8": {}
},
"fx": {
    "1": {
        "mdl": "NONE",
        "fxmix": 100
```

```
            },
            "2": {},..."16": {}
        },
        "cards": {
            "wlive": {
                "autoin": "OFF",
                "meters": true,
                "1": {
                    "cfg": {
                        "rectracks": "32",
                        "playmode": "PLAY"
                    }
                },
                "2": {}
            }
        },
        "play": {
            "playall": true,
            "repeat": false
        },
        "rec": {
            "path": "WINGREC",
            "resolution": "24",
            "channels": "2"
        }
    },
```

ce_data

ce_data contains all JSON structure elements representing the "Control Engine" settings for WING.
The ce_data class contains the objects: ***cfg, layer, user, gpio, safes, daw, midi, osc, lib,*** as shown below:



Note that for ease of access and ***programming*** using the native interface or OSC remote protocol, the ce_data JSON tree structure is appended to the ae_data tree structure.

```
"ce_data": {
        "cfg": {
            "lights": {
                "btns": 25,
                "leds": 90,
                "meters": 40,
                "rgbleds": 25,
                "chlcds": 60,
                "chlcdctr": 50,
                "chedit": 80,
                "main": 80,
                "glow": 0,
                "patch": 0,
                "lamp": 0
            },
            "rta": {
                "homedisp": "1/3",
                "homecol": "BL50",
                "hometap": "IN",
                "eqdisp": "1/4",
                "eqcol": "BL75",
                "cheqtap": "PRE",
                "chflttap": "PRE",
                "eqdecay": "MED",
                "eqdet": "PEAK",
                "eqrange": 30,
                "eqgain": 0,
                "eqauto": true
            },
            "mtrsfc": {
                "in": "PRE",
                "bus": "POST",
                "main": "POST",
                "mtx": "POST",
                "dca": "PRE"
            },
            "mtrpage": {
                "in": "PRE",
                "bus": "POST",
                "main": "POST",
                "mtx": "POST",
                "dca": "PRE"
            },
            "mainmtr": "MAIN.1",
            "mainpos": "AUTO",
            "soloexcl": true,
            "selfsolo": true,
            "solofsel": false,
            "sof2solo": false,
            "layerlinkl": false,
            "layerlinkr": false,
            "autoview": false,
            "csctouch": true,
```

```
        "autosel_L": false,
        "autosel_C": false,
        "autosel_R": false,
        "fdrsel": false,
        "fdrres": "AUTO",
        "fdrspd": "MED",
        "fdrbanking": true,
        "soffdr": "L/C",
        "sofbutton": "AUTO",
        "sofframe": true,
        "sofmode": true,
        "seldblclick": true,
        "mousetchdis": false,
        "mousespd": 1.77,
        "usrmode": "CC",
        "tapflash": "ON",
        "srcdisp": true,
        "lockmtr": false,
        "timefmt": "24H",
        "datefmt": "YMD",
        "filesort": "A->Z"
    },
    "layer": {
        "L": {
            "sel": 5,
            "1": {
                "ofs": 0,
                "name": "CH1-12",
                "1": {
                    "type": "CH",
                    "i": 1,
                    "dst": 1
                },
                "2": {},..."24": {}
            },
            "2": {}..."7": {}
            }
        },
        "C": {
            "sel": 2,
            "1": {
                "ofs": 0,
                "name": "DCA",
                "1": {
                    "type": "DCA",
                    "i": 1,
                    "dst": 1
                },
                "2": {},..."16": {}
            },
            "2": {}..."6": {}
        },
        "R": {
            "sel": 1,
            "1": {
                "ofs": 0,
                "name": "MAIN",
                "1": {
                    "type": "BUS",
                    "i": 17,
                    "dst": 1
                },
                "2": {},..."16": {}
            },
            "2": {}..."7": {}
        }
    },
    "user": {
        "sel": 1,
        "mode": "USER",
        "cmode": "HA",
        "gpio": {
            "1": {
                "bu": {
                    "mode": "OFF",
                    "name": "GPIO 1"
                }
```

```
            },
            "2": {},..."4": {}
        },
        "user": {
            "1": {
                "bu": {
                    "mode": "OFF",
                    "name": ""
                },
                "bd": {
                    "mode": "OFF",
                    "name": ""
                }
            },
            "2": {},..."4": {}
        },
        "daw1": {
            "1": {
                "bu": {
                    "mode": "DAWBTN",
                    "name": "STOP",
                    "btn": "T1"
                },
                "bd": {
                    "mode": "DAWBTN",
                    "name": "REWIND",
                    "btn": "T4"
                }
            },
            "2": {},..."4": {}
        },
        "daw2": {},..."daw4": {}
        },
        "1": {
            "1": {
                "led": false,
                "col": 1,
                "enc": {
                    "mode": "OFF",
                    "name": ""
                },
                "bu": {
                    "mode": "OFF",
                    "name": ""
                },
                "bd": {
                    "mode": "OFF",
                    "name": ""
                }
            },
            "2": {},..."4": {}
        },
        "2": {},..."16": {},
        "cuser": {
            "1": 1,
            "2": 1,
            "3": 1
        }
    },
    "gpio": {
        "1": {
            "mode": "TGLNO",
            "gpstate": false
        },
        "2": {},..."4": {}
    },
    "safes": {
        "ch": "",
        "aux": "",
        "bus": "",
        "main": "",
        "mtx": "",
        "dca": "",
        "mute": "",
        "fx": "",
        "source": {
            "LCL": "",
```

```
            "AUX": "",
            "A": "",
            "B": "",
            "C": "",
            "SC": "",
            "USB": "",
            "CRD": "",
            "MOD": "",
            "PLAY": "",
            "AES": "",
            "USR": "",
            "OSC": ""
        },
        "output": {
            "LCL": "",
            "AUX": "",
            "A": "",
            "B": "",
            "C": "",
            "SC": "",
            "USB": "",
            "CRD": "",
            "MOD": "",
            "REC": "",
            "AES": ""
        },
        "area": {
            "L": "0",
            "C": "0",
            "R": "0"
        },
        "custom": "",
        "setup": ""
    },
    "daw": {
        "on": true,
        "conn": "USB",
        "emul": "MCU",
        "config": "CC",
        "ccup": false,
        "disjog": false,
        "preset": "-"
    },
    "midi": {
        "enchctl": "OFF",
        "enfxctl": "OFF",
        "encustctl": "OFF",
        "ensysex": "OFF",
        "enmidicc": "OFF"
    },
    "osc": {
        "ronly": false
    },
    "lib": {}
},
```


More JSON files

WING desk provides more JSON files. Indeed, JSON format is also used to save/store channel, library, and effect presets. These files are created as you save presets and libraries that help you setup your system faster down the road.