Sennheiser Sound Control Protocol (SSC)

versatile command, control, and configuration for networked audio systems

Developer‘s guide for EW-DX

EW-DX EM 2 (Firmware 1.1.3)
CHG 70N (Firmware 1.0.6)

Sennheiser electronic GmbH & Co. KG

Am Labor 1, 30900 Wedemark, Germany, www.sennheiser.com
Publ. 03/2023

SENNHEISER

Table of Contents

1. Introduction

Modern professional audio devices are designed as building blocks for large, complex systems.

Whereas audio signal paths have converged to industry standards a long time ago, driven by practical necessities, and only recently challenged by new transport technologies like Ethernet, the professional audio markets have not evolved a similar technological convergence in the area of remote, centralised control of systems of audio equipment (the notable historical exception being MIDI, which but has a limited scope and extensibility).

In this heterogeneous environment of diverging standards proposed by individual vendors as well as open communities, there is no existing self-evident solution to be found for the needs raised by designing professional Sennheiser audio equipment.

As a consequence, communication protocols implemented in Sennheiser products have so far been designed on a single-product or product-family basis. This has worked sufficiently well, up to the point that separately developed protocols start to concur in nexus devices or applications, like:

- Wireless Systems Manager (PC-based control application for wireless transmission)
- Sennheiser Control Cockpit
- remote channel for Sennheiser microphones
- Media Control Systems (third party products, e.g., Crestron)
- A/V studio integration (third party products, e.g., Lawo)
- smartphone or tablet apps
- future centralised Sennheiser services

It has become evident that product-specific protocols fail to scale well in nexus products because of the added complexity in re-implementing the same remote control functionality from a customer point of view in a multitude of different backwards-compatible ways. It is not feasible to add more ever different technical solutions to the existing variety --- the aim must be to define a reasonably future-proof protocol suitable for existing as well as envisioned products, devices, and services.

A broad market evaluation of existing technical solutions was performed in a joint Sennheiser PRO division working group. As a result, it turns out that Open Sound Control comes closest to the specific needs for an extensible, future-proof command, control, metering, and configuration protocol for Sennheiser products.

This document describes the specific adaption of Open Sound Control to Sennheiser use, "Sennheiser Sound Control", SSC. The main other ingredient is JavaScript Object Notation (JSON), which enhances ease-of-use and the implementation complexity for small to smallest devices.

Note that the protocol is intended for command and control. Network audio streaming is entirely out of its scope.

2. Open Sound Control Overview

Open Sound Control (OSC) is a protocol developed at The Center For New Music and Audio Technology (CNMAT) at University of California, Berkeley.

The OSC specification Version 1.1 is available from the Open Sound Control website at:

http://www.opensoundcontrol.org/ .

The OSC Schema defined by MicroOSC at:

http://cnmat.berkeley.edu/library/uosc_project_documentation/osc_address_schema

was used as a starting point for some parts of the schema defined in this document.

OSC handles more advanced packet formats such as bundles of messages to be atomically executed at the same time with timestamps, as well as addresses with wildcards and array values.

2.1 JavaScript Object Notation Overview

JavaScript Object Notation (JSON) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Ruby, Python, and many others. These properties make JSON an ideal data-interchange language.

The central website for JSON information is http://json.org. JSON is formally specified in RFC 4627 (MIME-type *application/json*).

JavaScript Object Notation (JSON) is a text format for the serialization of structured data. It is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard, Third Edition.

JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays).

A string is a sequence of zero or more Unicode characters.

An object is an unordered collection of zero or more name/value pairs, where a name is a string and a value is a string, number, boolean, null, object, or array.

An array is an ordered sequence of zero or more values.

The terms "object" and "array" come from the conventions of JavaScript.

JSON's design goals were for it to be minimal, portable, textual, and a subset of JavaScript.

3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14/RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels".

3.1 Terminology

SSC Message	protocol unit of transmission
SSC Server	device, application or person that sends SSC messages
SSC Container	named entity containing SSC Methods or other Containers
SSC Method	named attribute or action callable on a SSC Server
SSC Address	full name of a SSC Method, including names of all enclosing Containers. may be represented by a JSON object hierarchy.
SSC Address Tree	a JSON object hierarchy consisting of one or more SSC Addresses.
SSC Address Space	hierarchical tree comprising all the SSC Addresses of a SSC Server
SSC Method Call	SSC Message requesting execution of a SSC Method
SSC Method Arguments	arguments included in a SSC Method Call
SSC Method Reply	SSC Message send by SSC Server as result of a Method Call
binary OSC	the binary OSC encoding as opposed to JSON-based SSC
restricted SSC Server	a SSC Server that doesn‘t implement some optional parts of this specification

4. SSC Data Structure Specification

4.1 Applying JSON to the OSC device model

OSC models the controlled device as a tree-shaped hierarchy of *methods*, with the method addresses constructed from the names of all the nodes in the hierarchy, written like a file path.

```
/                                   container at address "/"
        audio/                      container at address "/audio/"
                out1/               container at address "/audio/out1/"
                        label name  address "/audio/out1/label": method with string argument
                        level_db 5  address "/audio/out1/level_db": method with numeric argument
                        ...         more methods of  "/audio/out1"
                out2/               container at address "/audio/out2 /"
                ...                 methods of "/audio/out2"
        device/                     container at address "/device/"
                ...                 methods of "/device"
        ...                         more methods and containers of  "/"
```

JSON allows to model that structure as a hierarchy of *JSON objects*.

```
{                                       root object
        "audio": {                      object "audio"
                "out1": {               object "audio.out1"
                        "label": name,  numerical property "audio.out1.label"
                        "level_db": 5,  boolean property  "audio.out1.level_db"
                        ...             more properties of  "audio.out1"
                },
                "out2": {               object "audio.out2"
                        ...             properties of "audio.out2"
                },
        "device": {                     object "device"
                ...                     properties of "device"
        },
        ...                             more properties and objects of  the root object
}
```

The OSC Method Address (like "/audio/out1/level_db") is interpreted as a property path navigating through the hierarchy of JSON objects. The value of each property MUST be either a primitive JSON data type, or a JSON array. Rationale: This allows to clearly separate SSC Method Addresses from SSC Method Arguments at JSON parser level without knowledge of the underlying method address tree.

The resulting JSON tree structure of hierarchical objects, the SSC *Address Space*, is tailored to describe the functionality of a specific SSC Server, in the same way as foreseen by OSC.

In JSON it is possible to serialise the complete state of all properties in the tree to a closed form, thus describing the complete state of the SSC Server. In this way, JSON can be used as an excellent extensible data format for configuration files, or for scripting applications, which drive a system of SSC Servers through a sequence of programmed configurations.

For command and control applications it is desirable to access single properties independently. This can be achieved in JSON syntax by the simple convention, that all the properties of a SSC Server that are not mentioned in a JSON message are left unchanged.

In this way, applied to the example above, the JSON form

```
{ "audio": { "out1": { "level_db": 5 } } }
```

can be understood as a SSC Method Call of the SSC Method "/audio/out1/level_db" with the argument 5, presumably to set the level to that level, or as an SSC Method Reply message stating the current level.

4.2 JSON Message Transaction Syntax

The SSC Message exchange is described here as transaction using the following syntax:

Prefix "`TX:`" indicates a SSC Message that a SSC Client is sending to a SSC Server.

Prefix "`RX:`" indicates a SSC Message that the SSC Server will send back to the Client.

A SSC-Message is written verbatim, enclosed by curly brackets `{ }`.

A transaction to set the level of "audio" of "out1" to 1 then looks like this:

```
TX: { "audio": { "out1": { "level_db": 1 }}}
RX: { "audio": { "out1": { "level_db": 1 }}}
```

Note that the execution of the method results in a method reply message, which for simple property setters states the actual value of the property resulting from executing the message.

The resulting value may be different from the supplied argument, e.g., for a read-only property, or if the argument is out of range, and the device may adapt it to the allowed range (this is not considered as an error):

```
TX: { "audio": { "out1": { "level_db": 20000 }}}
RX: { "audio": { "out1": { "level_db": 6 }}}
```

Getter-methods, which request the value of a property from the SSC Server, are realised by supplying the special JSON value null as argument to method sent to the address of the property:

```
TX: { "audio": { "out1": { "level_db": null }}}
RX: { "audio": { "out1": { "level_db": 6 }}}
```

Compared to binary OSC, the JSON syntax is slightly more verbose for single attribute settings, but this is compensated when multiple attributes are set in the same transaction:

```
TX: { "audio": { "out1": { "level_db": 3, "label": null }}}
RX: { "audio": { "out1": { "level_db": 3, "label": "AF_LABEL" }}}
```

4.3 SSC JSON Message Syntax

4.3.1 Elementary data types

All SSC data is composed of the primitive JSON data types:

- string: a sequence of zero or more Unicode characters in UTF-8 encoding, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. Binary zero bytes can be included in a string using Unicode escape notation: "\u0000".
- number: a number in conventional "scientific" notation. `0, 42, -23, 3.141259, 1.0e+100` are all valid numbers. A Restricted SSC Server MAY reject non-integer numeric arguments, or it MAY adapt them by silently converting them to integer values.
- `true`: the boolean true value.
- `false`: the boolean false value.
- `null`: indicates a missing value; used as pseudo argument for getter-methods.

4.3.2 SSC Messages

A Message is the protocol unit of transmission. Any application that sends SSC Messages is a SSC Client, any application that receives SSC Messages is a SSC Server.

A SSC Message MUST be sent as a single closed JSON form describing a JSON object. Extra whitespace between the elements of the message MUST be ignored by the receiver.

This means that every SSC Message is enclosed in a pair of curly brackets `{ }`.

4.3.3 SSC Addresses

Every SSC Server implements a set of SSC Methods. SSC Methods are the potential destinations of SSC Messages received by the SSC Server, and correspond to each of the points of control that the application makes available. "Invoking" a SSC Method is analogous to a procedure call; it means supplying the method with arguments and causing the method's effect to take place. The SSC Server MUST respond to each received SSC Message by sending a SSC Method Reply Message to the originating SSC Client.

A SSC Server's SSC Methods are arranged in a tree structure called a SSC Address Space. The leaves of this tree are the SSC Methods and the branch nodes are called SSC Containers. A SSC Server's SSC Address Space MAY be dynamic; that is, its contents and shape MAY change over time.

Each SSC Method and each SSC Container other than the root of the tree MUST have a symbolic name which MUST be composed entirely of printable ASCII characters other than the following:

" "	space,	ASCII 32
"	double quote,	ASCII 34
#	number sign,	ASCII 35
*	asterisk,	ASCII 42
,	comma,	ASCII 44
/	slash,	ASCII 47
:	colon,	ASCII 58
?	question mark,	ASCII 63
[	open bracket,	ASCII 91
]	close bracket,	ASCII 93
{	open curly brace,	ASCII 123
}	close curly brace,	ASCII 125

The SSC Address of a SSC Method is a symbolic name giving the full path to the SSC Method in the SSC Address Space, starting from the root of the tree. A SSC Method's SSC Address begins with the character "/" (forward slash), followed by the names of all the containers, in order, along the path from the root of the tree to the SSC Method, separated by forward slash characters, followed by the name of the SSC Method. The syntax of SSC Addresses was chosen to match the syntax of URLs. The SSC address syntax SHOULD be used in documentation, but it SHOULD NOT be used as an argument to other SSC Methods; the JSON syntax of hierarchical objects SHOULD be used instead.

A SSC Method may be invoked with an empty argument list by supplying the JSON null value. This kind of SSC Method call SHOULD normally have the semantics of a query resulting in the current value of the property addressed by the method, without further side effects. SSC Methods that change the state of a SSC Server SHOULD normally have arguments.

Example:

- query current level of OUT1 output of AUDIO module:

```
TX: { "audio": { "out1": { "level_db": null }}}
RX: { "audio": { "out1": { "level_db": 5 }}}
```

- change level of OUT1 output of AUDIO module (note that the server adapts the value):

```
TX: { "audio": { "out1": { "level_db": 999 }}}
RX: { "audio": { "out1": { "level_db": 6 }}}
```

5. SSC subscriptions - /osc/state/subscribe

A subscription request is sent by a client to a server for an address pattern to subscribe to. The SSC Server normally accepts the subscription request, and remembers that the requesting client wishes to be notified about value changes of the subscribed addresses.

The SSC Server MAY refuse subscription requests, subject to device-specific policy or implementation specific limitations. The SSC Server MUST reply on the subscription request immediately either by acknowledging the request, or by sending an error reply.

The SSC Server MUST send an initial subscription notification to the client, which contains the result of calling the subscribed SSC Methods immediately with null-argument when the subscription request is handled. This initial notification MAY be bundled with the reply to the subscription request itself.

Each subscription notification MUST have identical contents to the reply to an imagined SSC Method invocation with null-argument to the subscribed SSC Method Address at the time that the notification is sent.

The SSC Client MAY bundle a call to `/osc/xid` with the subscription request. If a xid is supplied, a reply to `/osc/xid` MAY be bundled with each subscription notification, with the xid of the reply identical to that supplied by the client.

The SSC Server MUST send value changes of the subscribed addresses to the SSC Client. By default, the SSC Server will send subscription notifications if and only if the subscribed addresses change in value. The SSC Client can modify this behaviour by supplying optional parameters with the subscription request, allowing to either throttle the rate of notifications, or stimulate additional periodic notifications even if the subscribed addresses do not change in value.

Every subscription is specific to the connection between SSC Client and SSC Server. Also each SSC Method can only be subscribed once per connection. This means, that if a SSC Client requests a subscription which is already subscribed by that client on that connection, then the SSC Server MUST treat this as if the existing subscription was silently terminated and immediately requested anew.

5.1 Subscription notification rate parameters

Optional subscription request parameters related to notification rate:

- "min" minimum notification period (ms), 0=none, default 0
- "max" maximum notification period (ms), 0=none, default 0

If "min" is 0, then notifications are not sent when a subscribed address changes in value, they are only sent based on the "max" period. If "min" is greater than 0, notifications are sent after the specified time duration has elapsed, even if the value of the subscribed address is unchanged. If "max" is 0, then notifications are only sent when a value changes, or based on the "min" period. If "max" is greater than 0, then notifications are sent not earlier before the specified time duration has elapsed, even if the subscribed address changes value in the meantime.

5.2 Subscription cancelling and expiration

The SSC Server MUST terminate a subscription in these cases:

- the subscribed client cancels the subscription explicitly
- a maximum number of notifications has been sent
- a maximum lifetime relating to the begin of the subscription expires
- the SSC Client closes the connection
- the transport layer of the SSC connection signals a communication error

If the SSC Server decides to terminate the connection because the lifetime or notification count expires, then it MUST inform the SSC Client by sending an error reply "310 – subscription terminated" to the SSC address that terminates subscription together with or immediately after the last subscription notification.

Optional subscription request parameters related to termination:

- "cancel" "true" cancels the subscription (default false).
- "count" maximum number of notifications to send, default 1000
- "lifetime" maximum lifetime (s) of the subscription, default 10s

The SSC Client may renew a subscription at any time, thereby resetting all of the lifetime limitations. To renew a subscription, the SSC Client re-requests it; there's no difference between an initial subscription request and a renewal request.

5.3 Subscribing to multiple addresses

The SSC Client MAY request multiple subscriptions in a single request; either by providing them explicitly as SSC Address Tree, or by specifying address patterns as subscription addresses, or even both in the same request.

The SSC Server MAY either treat all those subscription requests separately, as if the addresses had all been requested for subscription individually. In this case all the subscription notifications would each contain the SSC Method Reply to a single subscribed address.

Alternatively, the SSC Server MAY bundle subscription notifications which happen to be sent at the same time into a single notification. The SSC Client MUST be able to handle a bundled notification if it requests multiple subscriptions in a single request, but it MUST NOT rely on the SSC Server bundling the notifications.

In any case the SSC Server SHOULD NOT bundle notification causes, meaning that the SSC Server SHOULD NOT send any subscription notifications for addresses in a bundle with notifications to other addresses, if they would not be sent if all subscriptions had been requested individually.

If some of the SSC addresses in a subscription request must be rejected with errors, whereas other subscriptions succeed, then the SSC Server MAY reject the request completely with an error reply detailing all the failed addresses. If possible, the SSC Server SHOULD instead execute the successful subscriptions and only reject the erroneous ones. This MUST result in a successful reply message to the subscription request, with the reply value including only the successful addresses. In this case the SSC Error state MUST be set to "210 – Partial Success", and MAY be accompanied by a parameter named "failed_addresses" with an Array of Address trees composed of all the failed Method Addresses (erroneous Addresses replaced by `{}`), in bundled or unbundled representation. The value of the Address in the Address Tree SHOULD be set to the SSC Error Code relating to the failure of the specific Address. See also the transaction example.

The SSC Server MAY also send a SSC Error "210 – Partial Success" when in fact all of the subscriptions have failed, because the SSC Client receives sufficient information in this Error Reply to work out this fact.

5.4 Supscription request and reply syntax

The SSC Address for subscriptions is `/osc/state/subscribe`.

This SSC Method may be called with a null parameter, which results in a SSC Address tree of all addresses currently subscribed by the SSC Client on the current connection.

The SSC Method also takes a structured parameter, specified as a JSON array.

Each element of the array is a SSC Address Tree specifying the SSC addresses that the SSC Client requests to subscribe. The SSC Address Tree MAY contain Address patterns.

A SSC Server that supports subscription MUST be able to interpret a single Address Tree element in the Method Argument array. Multiple Address Trees MAY be supported, or the SSC Server MAY reject them with a SSC Error 414 (request too complex).

The Response to the subscription Request will normally echo the Request, if all subscriptions can be handled successfully. If subscription parameters were requested, then the SSC Server MAY adapt the requested parameters, and MUST send back the adapted parameter values in the Reply. If multiple subscriptions are requested in a single Request, then the SSC Server might find it necessary to adapt subscription parameters differently for different Addresses. In that case, the array in the Reply MAY contain additonal Address trees containing additional adapted parameter objects. The SSC Server MAY also reject the subscription request completely (with SSC Error code 406), or partially (with SSC Error code 210) in such a case.

6. SSC Transport Layer Adaptations

The SSC data format as defined in the previous sections can be transported by different transport protocols, or stored in persistent files. This section specifies what transports are supported, and how the specific features of transport layers shall be applied to transporting SSC Messages.

6.1 UDP/IP

UDP/IP is the standard transport for all devices with an Ethernet interface or another interface typically used for internet connectivity. All those device MUST implement the UDP/IP transport for SSC.

The device implements UDP over IPv4.

One UDP datagram is used to transport one SSC Message. If the SSC Message is really large (e.g., a complete device configuration), IP fragmentation might fail, if a restricted device does not implement IP re-assembly properly. In that case, the SSC Server should break up the message into multiple SSC Method Calls instead. If atomic execution is relevant, SSC time tags may be used.

The UDP port number to used by the SSC Server should normally be discovered by the SSC Client by means of the server discovery protocol. The default port number is 45.

Rationale: No other standard UDP service is expected to use 45. The IANA reservation for a "Message Passing Service" is historic, and SSC is actually passing messages itself. Sennheiser was founded in 1945.

6.2 SSC Server Discovery

Networked devices implement DNS-SD (Apple Bonjour) as discovery protocol.

The DNS Service-Type is specified as "_ssc".

Because all networked SSC Servers must implement SSC-over-UDP, they MUST all publish a DNS-SD service under "_ssc. _udp". Those servers that additionally support TCP MUST publish another DNS-SD service under "_ssc._tcp".

The DNS-SD service instance name must be identical to the device name accessible as /device/name. DNS-SD automatic name collision resolution SHOULD be performed, and the resulting name changes MUST be reflected back into /device/name and the persistent device configuration. The renaming rules MAY be tailored to suit product specific requirements.

The DNS-SD service registration includes the port numbers used. SSC Clients SHOULD NOT rely on default ports.

The DNS-SD hostname SHOULD NOT be presented to the user. It may contain a unique identification part (e.g., derived from the device MAC or serial), to avoid name collisions and automatic renaming.

Additional information about the SSC Server may be provided with a DNS-SD TXT-record.

The following properties are currently defined for the TXT record:

- txtvers Version of the TXT record format. Currently "1".
- version SSC-Version provided by the SSC Server.

7. Developer’s Guide for EW-DX EM 2

This chapter describes in detail how a developer should use the SSC interface as implemented for the EW-DX EM 2.

7.1 Limitations

7.1.1 SSC Transport Layer

The SSC Server implemented for EW-DX devices supports only UDP/IP as transport protocol. All the devices support IPv4.

7.1.2 Subscriptions

The EW-DX EM 2 receiver supports SSC Method subscriptions from up to eight different SSC clients simultaneously.

8. SSC Method List (EW-DX EM 2)

8.1 /device/identification/visual

Command to start/stop device identification visual (true: start; false: stop; null: read status).

- type: Read/Write
- value: Number
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.2 /device/identity/version

Product version.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

8.3 /device/identity/vendor

Command replies "Sennheiser electronic GmbH & Co. KG".

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

8.4 /device/identity/serial

Not yet implemented. Serial no. from production.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

8.5 /device/identity/product

Product label.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

8.6 /device/network/ether/macs

List of MAC addresses.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

8.7 /device/network/ether/interfaces

List of all Ethernet interfaces.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

8.8 /device/network/ipv4/manual_netmask

Static (manual) Netmask like "255.255.255.0".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

8.9 /device/network/ipv4/manual_ipaddr

Static (manual) IPv4 address like "192.168.1.203".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

8.10 /device/network/ipv4/manual_gateway

Static (manual) Gateway address like "192.168.1.1".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

8.11 /device/network/ipv4/netmask

Online Netmask like "255.255.255.0".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

8.12 /device/network/ipv4/ipaddr

Online IPv4 address like "192.168.1.203".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

8.13 /device/network/ipv4/interfaces

List of IPv4 interface indices.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

8.14 /device/network/ipv4/gateway

Online Gateway address like "192.168.1.1".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

8.15 /device/network/ipv4/auto

Command to configure IP settings automatically
(true: use DHCP and ZeroConf (Auto-IP); false: set IP address/netmask/gateway manually).

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.16 /device/network/mdns

Command to enable, disable or read mDNS state.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

8.17 /device/timeprecision

SSC timed method execution is not supported. Returns 0.

- limits
  - type: Boolean

8.18 /device/time

SSC timed method execution is not supported. Returns 0.

- limits
  - type: Boolean

8.19 /device/preset_spacing

Get device preset spacing in kHz depending on LD mode being active or not.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: true
  - writeable: false
  - units: kHz

Example:

```
Tx: {"device":{"preset_spacing":null}}
Rx: {"device":{"preset_spacing":300}}
```

8.20 /device/restore

Command to restore device default settings.

- type: Write-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
"FACTORY_DEFAULTS"	restores all factory defaults
"AUDIO_DEFAULTS"	restores audio defaults only

8.21 /device/restart

Restarts the device.

- limits
  - type: Boolean
  - const: false
  - writeable: true

8.22 /device/name

Command to set or read the device name.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 18
  - subscr: true

8.23 /device/location

Command to set or read the device location.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 400
  - subscr: true

8.24 /device/link_density_mode

Switch RF transmission link density mode on if true is specified or off (standard mode) if false is specified.

Note: This command leads to a device reboot!

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

Example:

```
{"device":{"link_density_mode":true}}
```

8.25 /device/language

List of supported languages. English only = ["en_GB"].

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

8.26 /device/frequency_ranges

Command to read receiver device frequency ranges.

Format of replied string array:

```
["<range_1 start frequency>:<range_1 step>:<range_1 end frequency>",
 "<range_2 start frequency>:<range_2 step>:<range_2 end frequency>",...,
 "<range_n start frequency>:<range_n step>:<range_n end frequency>"]
```

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: -1

Example:

```
{"device":{"frequency_ranges":
["470000000:25000:514875000","522100000:25000:550000000"]}}
```

This means: [470MHz..514.875MHz], [522.1MHz..550MHz] step=25kHz

8.27 /device/frequency_code

Command to read receiver device frequency range code.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

Example:

```
{"device":{"frequency_code":"Q1-9"}}
```

8.28 /device/encryption

Activate/Deactivate encryption on the device.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.29 /interface/version

Command to read EW-DX SSC interface version.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

8.30 /osc/state/auth/access

Read authentication access of current SSC client. Response value is an array of strings, comma separated.

- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - subscr: true

8.31 /osc/state/prettyprint

SSC reply output style is not supported. Returns false.

- limits (hidden)

8.32 /osc/state/close

SSC connection close.

- limits (hidden)

8.33 /osc/state/subscribe

SSC subscriptions.

- limits (hidden)

8.34 /osc/feature/timetag

SSC timed method execution is not supported. Returns false.

- limits (hidden)

8.35 /osc/feature/baseaddr

SSC interactive method address base is not supported. Returns false.

- limits (hidden)

8.36 /osc/feature/subscription

SSC subscriptions are supported. Returns true.

- limits (hidden)

8.37 /osc/feature/pattern

SSC message dispatching and pattern matching are supported. Returns "*?".

- limits (hidden)

8.38 /osc/limits

SSC method parameter range reflection.

- limits (hidden)

8.39 /osc/schema

SSC schema reflection.

- limits (hidden)

8.40 /osc/version

SSC protocol version.

- limits (hidden)

8.41 /osc/xid

SSC transaction ID.

- limits (hidden)

8.42 /osc/ping

SSC ping.

- limits (hidden)

8.43 /osc/error

SSC error state.

- limits (hidden)

8.44 /rx1/identification/visual

Command to start/stop receiver channel 1 identification visual (true: start; false: stop; null: read status).

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.45 /rx1/presets/user/0

Set/get list of user frequencies [kHz] for receiver channel 1.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - count: -1
  - subscr: true

Example:

```
{"rx1":{"presets":{"user":{"0":[549200, 549800]}}}}
```

8.46 /rx1/presets/active

Set/get frequency from specific (presets) frequency list for receiver channel 1.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - subscr: true

Example:

select from factory list (index=0, channel=2):

```
{"rx1":{"presets":{"active":"factory:0:2"}}}
```

or select from user list (index=0, channel=1):

```
{"rx1":{"presets":{"active":"user:0:1"}}}
```

8.47 /rx1/sync_settings/trim_ignore

Skip/apply trim setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.48 /rx1/sync_settings/trim

TX trim for receiver channel 1.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 6
  - min: -12
  - inc: 1
  - subscr: true

8.49 /rx1/sync_settings/name_ignore

Skip/apply channel name setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.50 /rx1/sync_settings/mute_config_ignore

Skip/apply mute configuration (mute mode) setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.51 /rx1/sync_settings/mute_config

TX mute configuration (mute mode) for receiver channel 1.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	Disabled
rf_mute	RF Mute
af_mute	AF Mute

  - subscr: true

8.52 /rx1/sync_settings/lowcut_ignore

Skip/apply low cut frequency setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.53 /rx1/sync_settings/lowcut

TX low cut frequency for receiver channel 1.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	off
30 Hz	30 Hz
60 Hz	60 Hz
80 Hz	80 Hz
100 Hz	100 Hz
120 Hz	120 Hz

  - subscr: true

8.54 /rx1/sync_settings/lock_ignore

Skip/apply Auto Lock setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.55 /rx1/sync_settings/lock

TX Autolock off/on for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.56 /rx1/sync_settings/led_ignore

Skip/apply LED setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.57 /rx1/sync_settings/led

TX LED off/on for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.58 /rx1/sync_settings/frequency_ignore

Skip/apply frequency setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.59 /rx1/sync_settings/cable_emulation_ignore

Skip/apply cable emulation setting on TX sync action for receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.60 /rx1/sync_settings/cable_emulation

TX cable emulation for receiver channel 1.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	off
type1	Type 1
type2	Type 2
type3	Type 3

  - subscr: true

8.61 /rx1/warnings

Active warnings on receiver channel 1 stored in an array of strings.

Warnings: [Aes256Error, LowSignal, NoLink, RfPeak].

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - options:

Option	Description
Aes256Error	AES 256 error
LowSignal	LowSignal
NoLink	NoLink
RfPeak	RfPeak

  - subscr: true

Example:

```
Tx: {"rx1":{"warnings":null}}
Rx: {"rx1":{"warnings":["Aes256Error","RfPeak"]}}
```

8.62 /rx1/name

User-settable channel name.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 8
  - subscr: true

8.63 /rx1/mute

Set audio output mute on/off of receiver channel 1.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.64 /rx1/mates

Information about active mates of the receiver channel 1 stored as array of string values.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

Example:

```
Tx: {"rx1":{"mates":null}}
Rx: {"rx1":{"mates":["mates/tx1"]}}
```

8.65 /rx1/gain

Gain level for receiver channel 1. Valid values in range [-3dB..+42dB], increment step = 3dB.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 42
  - min: -3
  - inc: 3
  - subscr: true

8.66 /rx1/frequency

Carrier Frequency in kHz for receiver channel 1.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: kHz
  - max: 1999000
  - min: 470200
  - inc: 25
  - subscr: true

Example:

```
{"rx1":{"frequency":null}}
```

8.67 /rx1/audio

Information about active audio inputs/outputs of the receiver channel 1 stored as array of string values.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

Example:

```
Tx: {"rx1":{"audio":null}}
Rx: {"rx1":{"audio":["audio1/out1"]}}
```

8.68 /rx1/restore

Command to restore device audio default settings of receiver channel 1.

Valid options: "AUDIO_DEFAULTS" (restore audio defaults only).

- type: Write-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
AUDIO_DEFAULTS	Restore audio defaults only

8.69 /rx2/identification/visual

Command to start/stop receiver channel 2 identification visual
(true: start; false: stop; null: read status).

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.70 /rx2/presets/user/0

Set/get list of user frequencies [kHz] for receiver channel 2.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - count: -1
  - subscr: true

Example:

```
{"rx2":{"presets":{"user":{"0":[548000, 548300, 548600]}}}}
```

8.71 /rx2/presets/active

Set/get frequency from specific (presets) frequency list for receiver channel 2.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - subscr: true

Example:

select from factory list (index=0, channel=2):

```
{"rx2":{"presets":{"active":"factory:0:2"}}}
```

or select from user list (index=0, channel=1):

```
{"rx2":{"presets":{"active":"user:0:1"}}}
```

8.72 /rx2/sync_settings/trim_ignore

Skip/apply trim setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.73 /rx2/sync_settings/trim

TX trim for receiver channel 2.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 6
  - min: -12
  - inc: 1
  - subscr: true

8.74 /rx2/sync_settings/name_ignore

Skip/apply channel name setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.75 /rx2/sync_settings/mute_config_ignore

Skip/apply mute configuration (mute mode) setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.76 /rx2/sync_settings/mute_config

TX mute configuration (mute mode) for receiver channel 2.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	Disabled
rf_mute	RF Mute
af_mute	AF Mute

  - subscr: true

8.77 /rx2/sync_settings/lowcut_ignore

Skip/apply low cut frequency setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.78 /rx2/sync_settings/lowcut

TX low cut frequency for receiver channel 2.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	off
30 Hz	30 Hz
60 Hz	60 Hz
80 Hz	80 Hz
100 Hz	100 Hz
120 Hz	120 Hz

  - subscr: true

8.79 /rx2/sync_settings/lock_ignore

Skip/apply Auto Lock setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.80 /rx2/sync_settings/lock

TX Autolock off/on for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.81 /rx2/sync_settings/led_ignore

Skip/apply LED setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.82 /rx2/sync_settings/led

TX LED off/on for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.83 /rx2/sync_settings/frequency_ignore

Skip/apply frequency setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.84 /rx2/sync_settings/cable_emulation_ignore

Skip/apply cable emulation setting on TX sync action for receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.85 /rx2/sync_settings/cable_emulation

TX cable emulation for receiver channel 2.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
off	off
type1	Type 1
type2	Type 2
type3	Type 3

  - subscr: true

8.86 /rx2/warnings

Active warnings on receiver channel 2 stored in an array of strings.

Warnings: [Aes256Error, LowSignal, NoLink, RfPeak].

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - options:

Option	Description
Aes256Error	AES 256 error
LowSignal	LowSignal
NoLink	NoLink
RfPeak	RfPeak

  - subscr: true

Example:

```
Tx: {"rx2":{"warnings":null}}
Rx: {"rx2":{"warnings":["Aes256Error"]}}
```

8.87 /rx2/name

User-settable channel name.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 8
  - subscr: true

8.88 /rx2/mute

Set audio output mute on/off of receiver channel 2.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

8.89 /rx2/mates

Information about active mates of the receiver channel 2 stored as array of string values.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

Example:

```
Tx: {"rx2":{"mates":null}}
Rx: {"rx2":{"mates":["mates/tx2"]}}
```

8.90 /rx2/gain

Gain level for receiver channel 2. Valid values in range [-3dB..+42dB], increment step = 3dB.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 42
  - min: -3
  - inc: 3
  - subscr: true

8.91 /rx2/frequency

Carrier Frequency in kHz for receiver channel 2.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: kHz
  - max: 1999000
  - min: 470200
  - inc: 25
  - subscr: true

Example:

```
{"rx2":{"frequency":null}}
```

8.92 /rx2/audio

Information about active audio inputs/outputs of the receiver channel 2 stored as array of string values.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

Example:

```
Tx: {"rx2":{"audio":null}}
Rx: {"rx2":{"audio":["audio1/out2"]}}
```

8.93 /rx2/restore

Command to restore device audio default settings of receiver channel 2.

Valid options: "AUDIO_DEFAULTS" (restore audio defaults only).

- type: Write-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
AUDIO_DEFAULTS	Restore audio defaults only

8.94 /audio1/out1/level

AF output level for receiver channel 1.

Valid values in range [-24dB..+18dB], increment step = 6dB.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 18
  - min: -24
  - inc: 6
  - subscr: true

8.95 /audio1/out2/level

AF output level for receiver channel 2.

Valid values in range [-24dB..+18dB], increment step = 6dB.

- type: Read/Write
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: true
  - units: dB
  - max: 18
  - min: -24
  - inc: 6
  - subscr: true

8.96 /m/rx1/rssi

Returns RF radio signal strength indicator (RSSI) for receiver channel 1.

- type: double (ro)
- value range: -107.0..0.0 [dBm]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dBm
  - max: 0
  - min: -107
  - subscr: true

Example:

```
{"m":{"rx1":{"rssi":null}}}
```

8.97 /m/rx1/rsqi

Returns RF signal quality indicator for receiver channel 1.

- type: number (ro)
- value range: 0..100 [%]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
{"m":{"rx1":{"rsqi":null}}}
```

8.98 /m/rx1/divi

Returns RF diversity indicator for receiver channel 1.

0 is for None; 1 is for antenna input A; 2 is for antenna input B.

- type: number (ro)
- limits
  - type: Number
  - const: false
  - writeable: false
  - max: 2
  - min: 0
  - inc: 1
  - options:

Option	Description
0	none
1	antenna input A
2	antenna input B

  - subscr: true

Example:

```
{"m":{"rx1":{"divi":null}}}
```

8.99 /m/rx1/af

Returns audio level for receiver channel 1.

- type: double (ro)
- value range: -138.5..0.0 [dBfs]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dBfs
  - max: 0
  - min: -138.5
  - subscr: true

8.100 /m/rx2/rssi

Returns RF radio signal strength indicator (RSSI) for receiver channel 2.

- type: double (ro)
- value range: -107.0..0.0 [dBm]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dBm
  - max: 0
  - min: -107
  - subscr: true

Example:

```
{"m":{"rx2":{"rssi":null}}}
```

8.101 /m/rx2/rsqi

Returns RF signal quality indicator for receiver channel 2.

- type: number (ro)
- value range: 0..100 [%]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
{"m":{"rx2":{"rsqi":null}}}
```

8.102 /m/rx2/divi

Returns RF diversity indicator for receiver channel 2.

0 is for None; 1 is for antenna input A; 2 is for antenna input B.

- type: number (ro)
- limits
  - type: Number
  - const: false
  - writeable: false
  - max: 2
  - min: 0
  - inc: 1
  - options:

Option	Description
0	none
1	antenna input A
2	antenna input B

  - subscr: true

Example:

```
{"m":{"rx2":{"divi":null}}}
```

8.103 /m/rx1/af

Returns audio level for receiver channel 2.

- type: double (ro)
- value range: -138.5..0.0 [dBfs]
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dBfs
  - max: 0
  - min: -138.5
  - subscr: true

8.104 /mates/tx1/battery/type

TX battery type info got from receiver channel 1 stored as string.

Possible values: "Battery", "Primary Cell".

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"battery":{"type":null}}}}
Rx: {"mates":{"tx1":{"battery":{"type":"Battery"}}}}
```

8.105 /mates/tx1/battery/lifetime

TX battery lifetime in minutes got from receiver channel 1 stored as number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery information.

- type: Read-only
- value: int
- limits
  - type: Number
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"battery":{"lifetime":null}}}}
Rx: {"mates":{"tx1":{"battery":{"lifetime":312}}}}
```

8.106 /mates/tx1/battery/gauge

TX battery gauge (charge in percent) got from receiver channel 1 stored as number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery gauge.

- type: Read-only
- value: int
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"battery":{"gauge":null}}}}
Rx: {"mates":{"tx1":{"battery":{"gauge":65}}}}
```

8.107 /mates/tx1/warnings

TX warnings occuring on receiver active link channel 1 stored in an array of strings.

Warnings: [AfPeak, LowBattery].

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - options:

Option	Description
AfPeak	AF Peak
LowBattery	Low Battery

  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"warnings":null}}}
Rx: {"mates":{"tx1":{"warnings":["LowBattery","AfPeak"]}}}
```

8.108 /mates/tx1/version

TX SW version information got from receiver channel 1 stored as a string .

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"version":null}}}
Rx: {"mates":{"tx1":{"version":"1.2.3"}}}
```

8.109 /mates/tx1/type

TX type information got from receiver channel 1 stored as a string .

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"type":null}}}
Rx: {"mates":{"tx1":{"type":"SKMPLUS"}}}
```

8.110 /mates/tx1/trim

TX trim information got from receiver channel 1 stored as a number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: number
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dB
  - max: 6
  - min: -12
  - inc: 1
  - subscr: true

Example:

```
Tx: {"mates":{"tx1":{"trim":null}}}
Rx: {"mates":{"tx1":{"trim":-12}}}
```

8.111 /mates/tx1/name

TX name information got from receiver channel 1 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"name":null}}}]}}}
Rx: {"mates":{"tx1":{"name":"DEVICE2"}}}
```

8.112 /mates/tx1/mute_config

TX mute configuration (mute mode) information got from receiver channel 1 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - options:

Option	Description
off	Disabled
rf_mute	RF mute
af_mute	AF mute

  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"mute_config":null}}}]}}}
Rx: {"mates":{"tx1":{"mute_config":"af_mute"}}}
```

8.113 /mates/tx1/mute

TX Mute Switch On/Off information got from receiver channel 1 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"mute":null}}}]}}}
Rx: {"mates":{"tx1":{"mute":true}}}
```

8.114 /mates/tx1/lowcut

TX lowcut information got from receiver channel 1 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

Valid strings: "off", "30 Hz", "60 Hz", ...

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"lowcut":null}}}]}}}
Rx: {"mates":{"tx1":{"lowcut":"60 Hz"}}}
```

8.115 /mates/tx1/lock

TX auto lock information got from receiver channel 1 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"lock":null}}}]}}}
Rx: {"mates":{"tx1":{"lock":true}}}
```

8.116 /mates/tx1/led

TX LED information got from receiver channel 1 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"led":null}}}]}}}
Rx: {"mates":{"tx1":{"led":true}}}
```

8.117 /mates/tx1/identification

TX identification information got from receiver channel 1 stored as a boolean.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"identification":null}}}]}}}
Rx: {"mates":{"tx1":{"identification":false}}}
```

8.118 /mates/tx1/capsule

TX capsule information got from receiver channel 1 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

Valid strings: "unknown", "KK 205", "KK 204", "ME 9002", "ME 9004", "ME 9005", "MME 865", "MD 9235", "MMD 945", "MMD 935", "MMD 845", "MMD 835", "MMD 815-1", "MMK 965", "MMD 42-I"

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"capsule":null}}}]}}}
Rx: {"mates":{"tx1":{"capsule":"MMD 935"}}}
```

8.119 /mates/tx1/cable_emulation

TX cable emulation information got from receiver channel 1 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - options:

Option	Description
off	Off
type1	Type 1
type2	Type 2
type3	Type 3

  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx1":{"cable_emulation":null}}}]}}}
Rx: {"mates":{"tx1":{"cable_emulation":"type2"}}}
```

8.120 /mates/tx2/battery/type

TX battery type info got from receiver channel 2 stored as string.

Possible values: "Battery", "Primary Cell".

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"battery":{"type":null}}}}
Rx: {"mates":{"tx2":{"battery":{"type":"Battery"}}}}
```

8.121 /mates/tx2/battery/lifetime

TX battery lifetime in minutes got from receiver channel 2 stored as number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery information.

- type: Read-only
- value: int
- limits
  - type: Number
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"battery":{"lifetime":null}}}}
Rx: {"mates":{"tx2":{"battery":{"lifetime":312}}}}
```

8.122 /mates/tx2/battery/gauge

TX battery gauge (charge in percent) got from receiver channel 2 stored as number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid battery gauge.

- type: Read-only
- value: int
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"battery":{"gauge":null}}}}
Rx: {"mates":{"tx2":{"battery":{"gauge":80}}}}
```

8.123 /mates/tx2/warnings

TX warnings occuring on receiver active link channel 2 stored in an array of strings.

Warnings: [AfPeak, LowBattery].

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - options:

Option	Description
AfPeak	AF Peak
LowBattery	Low Battery

  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"warnings":null}}}
Rx: {"mates":{"tx2":{"warnings":[]}}}
```

8.124 /mates/tx2/version

TX SW version information got from receiver channel 2 stored as a string .

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"version":null}}}
Rx: {"mates":{"tx2":{"version":"1.2.3"}}}
```

8.125 /mates/tx2/type

TX type information got from receiver channel 2 stored as a string .

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"type":null}}}
Rx: {"mates":{"tx2":{"type":"SKM"}}}
```

8.126 /mates/tx2/trim

TX trim information got from receiver channel 2 stored as a number.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: number
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: dB
  - max: 6
  - min: -12
  - inc: 1
  - subscr: true

Example:

```
Tx: {"mates":{"tx2":{"trim":null}}}
Rx: {"mates":{"tx2":{"trim":-12}}}
```

8.127 /mates/tx2/name

TX name information got from receiver channel 2 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"name":null}}}]}}}
Rx: {"mates":{"tx2":{"name":"DEVICE2"}}}
```

8.128 /mates/tx2/mute_config

TX mute configuration (mute mode) information got from receiver channel 2 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - options:

Option	Description
off	Disabled
rf_mute	RF mute
af_mute	AF mute

  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"mute_config":null}}}]}}}
Rx: {"mates":{"tx2":{"mute_config":"af_mute"}}}
```

8.129 /mates/tx2/mute

TX Mute Switch On/Off information got from receiver channel 2 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"mute":null}}}]}}}
Rx: {"mates":{"tx2":{"mute":true}}}
```

8.130 /mates/tx2/lowcut

TX lowcut information got from receiver channel 2 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

Valid strings: "off", "30 Hz", "60 Hz", ...

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"lowcut":null}}}]}}}
Rx: {"mates":{"tx2":{"lowcut":"60 Hz"}}}
```

8.131 /mates/tx2/lock

TX auto lock information got from receiver channel 2 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"lock":null}}}]}}}
Rx: {"mates":{"tx2":{"lock":true}}}
```

8.132 /mates/tx2/led

TX LED information got from receiver channel 2 stored as a bool.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"led":null}}}]}}}
Rx: {"mates":{"tx2":{"led":true}}}
```

8.133 /mates/tx2/identification

TX identification information got from receiver channel 2 stored as a boolean.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: bool
- limits
  - type: Boolean
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"identification":null}}}]}}}
Rx: {"mates":{"tx2":{"identification":false}}}
```

8.134 /mates/tx2/capsule

TX capsule information got from receiver channel 2 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

Valid strings: "unknown", "KK 205", "KK 204", "ME 9002", "ME 9004", "ME 9005", "MME 865", "MD 9235", "MMD 945", "MMD 935", "MMD 845", "MMD 835", "MMD 815-1", "MMK 965", "MMD 42-I"

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"capsule":null}}}]}}}
Rx: {"mates":{"tx2":{"capsule":"MMD 935"}}}
```

8.135 /mates/tx2/cable_emulation

TX cable emulation information got from receiver channel 2 stored as a string.

Replied error code 424 indicates that TX is not present or doesn‘t send valid information.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - options:

Option	Description
off	Off
type1	Type 1
type2	Type 2
type3	Type 3

  - subscr: true

Example:

```
Tx: {"osc":{"state":{"subscribe":[{"#":{"min":0, "max":0,"count":1000,
     "lifetime":60},"mates":{"tx2":{"cable_emulation":null}}}]}}}
Rx: {"mates":{"tx2":{"cable_emulation":"type2"}}}
```

8.136 /mates/active

Information about active mates of the device stored as array of string values.

- type: Read-only
- value: string
- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - subscr: true

Example:

```
Tx: {"mates":{"active":null}}
Rx: {"mates":{"active":["mates/tx1","mates/tx2"]}}
```

9. SSC Error List (EW-DX EM 2)

- 100 : continue
- 102 : processing
- 200 : OK
- 201 : Created
- 202 : Adapted
- 210 : Partial Success
- 310 : subscription terminates
- 400 : message not understood
- 401 : authorisation needed
- 403 : forbidden
- 404 : address not found
- 406 : not acceptable
- 408 : request time out
- 409 : conflict
- 410 : gone
- 413 : request too long
- 414 : request too complex
- 422 : unprocessable entity
- 423 : locked
- 424 : failed dependency
- 450 : answer too long
- 454 : parameter address not found
- 500 : internal server error
- 501 : not implemented
- 503 : service unavailable

10. Developer’s Guide for CHG 70N

This chapter describes in detail how a developer should use the SSC interface as implemented for the CHG 70N.

10.1 Limitations

10.1.1 SSC Transport Layer

The SSC Server implemented for EW-DX devices supports only UDP/IP as transport protocol. All the devices support IPv4.

10.1.2 Subscriptions

The CHG 70N charger supports SSC Method subscriptions from up to eight different SSC clients simultaneously.

11. SSC Method List (CHG 70N)

11.1 /device/identification/visual

Command to start/stop device identification visual (true: start; false: stop; null: read status).

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

11.2 /device/identity/version

Product version.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

11.3 /device/identity/vendor

Command replies "Sennheiser electronic GmbH & Co. KG".

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

11.4 /device/identity/serial

Not yet implemented. Serial no. from production.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

11.5 /device/identity/product

Product label.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

11.6 /device/network/ether/macs

List of MAC addresses.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

11.7 /device/network/ether/interfaces

List of all Ethernet interfaces.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

11.8 /device/network/ipv4/manual_netmask

Static (manual) Netmask like "255.255.255.0".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

11.9 /device/network/ipv4/manual_ipaddr

Static (manual) IPv4 address like "192.168.1.203".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

11.10 /device/network/ipv4/manual_gateway

Static (manual) Gateway address like "192.168.1.1".

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

11.11 /device/network/ipv4/netmask

Online Netmask like "255.255.255.0".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

11.12 /device/network/ipv4/ipaddr

Online IPv4 address like "192.168.1.203".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

11.13 /device/network/ipv4/interfaces

List of IPv4 interface indices.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

11.14 /device/network/ipv4/gateway

Online Gateway address like "192.168.1.1".

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 1
  - subscr: true

11.15 /device/network/ipv4/auto

Command to configure IP settings automatically
(true: use DHCP and ZeroConf (Auto-IP); false: set IP address/netmask/gateway manually).

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

11.16 /device/network/mdns

Command to enable, disable or read mDNS state.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

11.17 /device/update/progress

- type: Number (read only)
- value range: 0..100 [%], Returns always a valid value, also if device is not in update mode.
- limits
  - type: Number
  - const: false
  - writeable: false
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

11.18 /device/update/error

- type: String (read only)
- value range: "NONE" or a human readable error message
- limits
  - type: string
  - const: false
  - writeable: false
  - subscr: true

11.19 /device/update/enable

- type: Boolean (read-write)
- value range: "NONE" or a human readable error message
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - subscr: true

11.20 /device/timeprecision

SSC timed method execution is not supported. Returns 0.

- limits
  - type: Boolean

11.21 /device/time

SSC timed method execution is not supported. Returns 0.

- limits
  - type: Boolean

11.22 /device/restore

Command to restore device default settings.

- type: Write-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - options:

Option	Description
"FACTORY_DEFAULTS"	restores all factory defaults

11.23 /device/restart

Restarts the device.

- limits
  - type: Boolean
  - const: false
  - writeable: true

11.24 /device/name

Command to set or read the device name.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 18
  - subscr: true

11.25 /device/location

Command to set or read the device location.

- type: Read/Write
- value: String
- limits
  - type: String
  - const: false
  - writeable: true
  - length: 400
  - subscr: true

11.26 /device/language

List of supported languages. English only = ["en_GB"].

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false
  - count: 1

11.27 /interface/version

Command to read EW-DX SSC interface version.

- type: Read-only
- value: String
- limits
  - type: String
  - const: true
  - writeable: false

11.28 /osc/state/auth/access

Read authentication access of current SSC client. Response value is an array of strings, comma separated.

- limits
  - type: String
  - const: false
  - writeable: false
  - count: -1
  - subscr: true

11.29 /osc/state/prettyprint

SSC reply output style is not supported. Returns false.

- limits (hidden)

11.30 /osc/state/close

SSC connection close.

- limits (hidden)

11.31 /osc/state/subscribe

SSC subscriptions.

- limits (hidden)

11.32 /osc/feature/timetag

SSC timed method execution is not supported. Returns false.

- limits (hidden)

11.33 /osc/feature/baseaddr

SSC interactive method address base is not supported. Returns false.

- limits (hidden)

11.34 /osc/feature/subscription

SSC subscriptions are supported. Returns true.

- limits (hidden)

11.35 /osc/feature/pattern

SSC message dispatching and pattern matching are supported. Returns "*?".

- limits (hidden)

11.36 /osc/limits

SSC method parameter range reflection.

- limits (hidden)

11.37 /osc/schema

SSC schema reflection.

- limits (hidden)

11.38 /osc/version

SSC protocol version.

- limits (hidden)

11.39 /osc/xid

SSC transaction ID.

- limits (hidden)

11.40 /osc/ping

SSC ping.

- limits (hidden)

11.41 /osc/error

SSC error state.

- limits (hidden)

11.42 /bays/storage_mode

Starts storage mode.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - count: 1
  - subscr: true

11.43 /bays/identify

Starts flashing leds of CHG 70N.

- type: Read/Write
- value: Boolean
- limits
  - type: Boolean
  - const: false
  - writeable: true
  - count: 2
  - subscr: true

Example:

```
{"bays":{"identify":[true,true]}}
```

11.44 /bays/device_type

Reads type of device being plugged in CHG 70N.

- type: Read-only
- value: String
- limits
  - type: String
  - const: false
  - writeable: false
  - count: 2
  - options:

1.	EW-DX SK
2.	EW-DX SK 3-pin
3.	EW-DX SKM
4.	EW-DX SKM-S
5.	BA70
6.	NONE
7.	UNKNOWN

  - subscr: true

Example:

```
{"bays":{"device_type":[BA70,EW-DX SK]}}
```

11.45 /bays/bat_timetofull

Reads how many minutes are needed till battery is fully charged.

- type: Read-only
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: false
  - count: 2
  - units: minutes
  - max: 1000
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
{"bays":{"bat_timetofull":null}}
```

11.46 /bays/bat_health

Reads current state of health in % from battery.

- type: Read-only
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: false
  - count: 2
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
{"bays":{"bat_health":null}}
```

11.47 /bays/bat_gauge

Reads current state of charge value in % from battery.

- type: Read-only
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: false
  - count: 2
  - units: %
  - max: 100
  - min: 0
  - inc: 1
  - subscr: true

Example:

```
{"bays":{"bat_gauge":null}}
```

11.48 /bays/bat_cycles

Reads current state of cycles from battery.

- type: Read-only
- value: Number
- limits
  - type: Number
  - const: false
  - writeable: false
  - count: 2
  - max: 10000000
  - min: 0
  - subscr: true

Example:

```
{"bays":{"bat_cycles":null}}
```

12. SSC Error List (CHG 70N)

- 100 : continue
- 102 : processing
- 200 : OK
- 201 : Created
- 202 : Adapted
- 210 : Partial Success
- 310 : subscription terminates
- 400 : message not understood
- 401 : authorisation needed
- 403 : forbidden
- 404 : address not found
- 406 : not acceptable
- 408 : request time out
- 409 : conflict
- 410 : gone
- 413 : request too long
- 414 : request too complex
- 422 : unprocessable entity
- 423 : locked
- 424 : failed dependency
- 450 : answer too long
- 454 : parameter address not found
- 500 : internal server error
- 501 : not implemented
- 503 : service unavailable