

## EMSCOPE WEBSOCKET PROTOCOL DOCUMENTATION

The Emscope software consists in a *WebSocket* server listening on *port 8010* of the machine and a *Web Server* listening on the standard HTTP *port 80* that serves the front end html and related files.

To configure, parametrize the measurements and manage the standards, a *JSON* style command set is used through a *WebSocket* connection.

### 1. First connection

Once the connection is established, the server will return the device's *serial number*, *measurement uncertainty* and *number of points*, i.e:

```
{
  "SN": "123456789",
  "measurement_uncertainty": "0.5 dB",
  "num_points": 8192
}
```

### 2. Default values

The default values for measurements are:

- Channel: LG
- Detector: PK
- Tracer: Clear/write
- Units: dbuv
- RBW: 9
- Subranges: 10
- Margin: 10
- Active: true
- Mode: circuit
- input\_attenuator: auto
- reference\_level: 100 (dbuV)
- average: 10

### 3. Commands and responses

The following parameters can be sent to configure the device:

Name	Parameter	Type	Default
Channel	measure_channel	"lg", "ng", "cm", "dm"	"lg"
Detector type	detector_type	"pk", "qp", "av"	"pk"
Tracer	trace_type	"clearwrite", "maxhold", "minhold", "freeze", "average"	"clearwrite"
RBW	rbw	"9", "200", "120", "1", "10", "200_9", "1_10"	"9"
Average (for tracer average)	average	integer (10 - 20)	10
Mode	mode	"circuit", "modal"	"circuit"
Reference level	reference_level	integer	100
Input attenuator	input_attenuator	{0..78} (integer), "auto"	"auto"
Amplitude units	amp_units	"dbm", "dbmv", "dbuv", "watts", "volts"	"dbuv"

Name	Parameter	Type	Default
Serial number	SN	string	
Sweep time	sweep_time	{1..15} (float)	1
External loss attenuator	external_loss	External loss name (string)	none

## 4. Changing RBW

After setting the RBW we shouldn't send any commands until the device it sends back the RBW value. This is so because a hot-change of firmware, that typically takes around 3.5 seconds, is required. To set a new RBW, send a **rbw** field with the string corresponding to the RBW:

```
{
  "rbw": "9"
}
```

RBW table:

Value	RBW	Frequency	Standard
200	200 Hz	9 kHz - 150 kHz	CISPR 16-1-1
9	9 kHz	150 kHz - 30 MHz	CISPR 16-1-1
120	120 kHz	30 MHz - 110 MHz	CISPR 16-1-1
1	1 kHz	10 kHz - 150 kHz	MIL-STD-461
10	10 kHz	150 kHz - 30 MHz	MIL-STD-461
200_9	200 Hz / 9 kHz	9 kHz - 30 MHz	Dual-band
1_10	1 kHz / 10 kHz	10 kHz - 30 MHz	Dual-band

## 5. Initial configuration

A typical initial configuration parameters example would be:

```
{
  "detector_type": "pk",
  "measure_channel": "lg",
  "trace_type": "clearwrite",
  "amp_units": "dbmv",
  "rbw": "9",
  "reference_level": 70,
  "input_attenuator": "auto",
  "sweep_time": "1"
}
```

## 6. Receiving the measurements

To start receiving measurements, we should at least send the **trace\_type**. The measurements are sent in a **values** object immediately after the measure is done. The result contains an array of arrays recodesenting the frequency and the read value (**[frequency, value]**). Another key, named **overload**, will be set to *true* if an overload situation is detected. Example:

```
{
  "values": [
    [ 0, 0.024687 ],
    [ 7326, 0.014846 ],
    [ 10000, 0.00344 ],
  ]
}
```

```
    ...
  ],
  "overload": false
}
```

## 7. Locking the device to the current session

To ensure that the device is only being configured from one source, you ought to set a *session\_UUID* variable to a random value. Use the same value for subsequent connections. I.e:

```
{
  "session_UUID": "12345qwerty"
}
```

This *session UUID* must be sent with every *WebSocket* connection. If the device is currently locked with another *session UUID*, the device will disconnect us with *WebSocket* close reason code 4003.

## 8. Standards (or Limits)

### 8.1 Getting the standards list

Send the field **get\_standards** with a value of "true":

```
{
  "get_standards": true
}
```

It will return a **standards** object with an array of standards with the following format:

```
"Standard name": [ { "rbw": value, "data":
  [ From frequency (Mhz), To frequency (Mhz), From Quasi-peak (dBuV), To Quasi-peak (dBuV),
  Average from Quasi-peak (dBuV), Average to Quasi-peak (dBuV) ],
  ...
}
]
```

For example:

```
{
  "standards": [
    {
      "Standard One": { "rbw": "9", "data": [
        [1, 2, 3, 4, 5, 6],
        [7, 8, 9, 10, 11, 12],
        ...
      ]
    }
  ],
  {
    "Standard Two": { "rbw": "200", "data": [
      [21, 22, 23, 24, 25, 26],
      [27, 28, 29, 210, 211, 212],
    ]
  }
}
```

```
    ...  
  ]  
}  
,  
...  
]  
}
```

## 8.2 Creating a standard

To create a standard, send a **name** field with the name of the new standard, a **modify** field set to "false" and a **values** array in the same format we described before when receiving the standards list.

```
{  
  "name": "New standard",  
  "modify": false,  
  "standard_rbw": "10",  
  "values": [  
    ["1", "2", "3", "4", "5", "6"],  
    ["11", "12", "13", "14", "15", "16"],  
    ...  
  ]  
}
```

## 8.3 Editing a standard

Send the same values as creating it, but with the field *modify* set to *true* and an additional field, *original\_name*, just in case we wanted to rename the original *name*, i.e:

```
{  
  "original_name": "New standard",  
  "name": "Modified standard",  
  "modify": true,  
  "standard_rbw": "1",  
  "values": [  
    ["21", "22", "23", "24", "25", "26"],  
    ["11", "12", "13", "14", "15", "16"],  
    ...  
  ]  
}
```

## 8.4 Deleting a standard

Simply send a **delete\_standard** with the name of the standard, i.e:

```
{  
  "delete_standard": "Standard One"  
}
```

## 8.5 Resetting standards to factory defaults

Simply send a **reset\_standards** with a value of "true":

```
{
  "reset_standards": true
}
```

## 9. Sessions (or Presets)

### 9.1 Getting the sessions list

Send the field **get\_sessions** with a value of "true":

```
{
  "get_sessions": true
}
```

It will return a **sessions** object with an array of sessions with the following format:

```
[
  "Session name": [BLOB DATA],
  ...
]
```

For example:

```
{
  "sessions": [
    {
      "Session One":
      "{\"PROGRAM_VERSION\":1.25,\"id_counter\":2,\"current_center_freq_units\":\"mhz\", \"current_start_freq_units\":\"mhz\", \"current_stop_freq_units\":\"mhz\", \"current_span_units\":\"mhz\", \"current_center_frequency\":15000000, \"current_start_frequency\":0, \"current_stop_frequency\":30000000, \"current_span\":30000000, \"measure_mode\":\"circuit\", \"current_display_scale\":0, \"aDetectors\": [{\"type\":\"normal\", \"measure_channel\":\"lg\", \"trace_type\":\"clearwrite\", \"detector_type\":\"pk\", \"amp_units\":\"dbuv\", \"color\":\"red\", \"hidden\":false, \"standard\":null, \"id\":0, \"sweep_time\":1}, {\"type\":\"normal\", \"measure_channel\":\"lg\", \"trace_type\":\"clearwrite\", \"detector_type\":\"pk\", \"amp_units\":\"dbuv\", \"color\":\"red\", \"hidden\":false, \"standard\":null, \"id\":1}], \"reference_level\":100, \"reference_level_volts\":0.1, \"amplitude_step\":10, \"aCurrentFreq\": [150000, 30000000], \"aCurrentSpan\": [0, 100], \"aMarkers\": [[], []], \"aMMarkers\": [], \"aStandards\": {\"CISPR 22 CLASS A\": [[\"0.15\", \"0.5\", \"79\", \"79\", \"66\", \"66\"], [\"0.5\", \"30\", \"73\", \"73\", \"60\", \"60\" ]], \"CISPR 22 CLASS B\": [[\"0.15\", \"0.5\", \"66\", \"56\", \"56\", \"46\"], [\"0.5\", \"5\", \"56\", \"56\", \"46\", \"46\" ], [\"5\", \"30\", \"60\", \"60\", \"50\", \"50\"]]}, \"aStandardsQuasiPeak\": {}, \"aStandardsAverage\": {}, \"active_tab\":1, \"ZERO_SPAN_MODE\":false, \"amp_units\":\"dbuv\", \"rbw\":\"9\", \"input_atenuator\":\"auto\", \"sweep_time\":1, \"current_color_theme\":\"dark\"}",
      ...
    ]
  },
  {
    "Session Two": ".....",
    ...
  }
]
```

```
    ]  
  },  
  ...  
]  
}
```

## 9.2 Creating a session

To create a session, send a **session\_name** field with the name of the new session, a **modify** field set to “false” and a **session\_values** array in the same format we described before when receiving the sessions list.

```
{  
  "session_name": "New session",  
  "modify": false,  
  "session_values": ".....",  
  ...  
}
```

## 9.3 Editing a session

Send the same values as creating it, but with the field *modify* set to *true* and an additional field, *original\_session*, just in case we wanted to rename the original *name*, i.e:

```
{  
  "original_session": "New session",  
  "session_name": "Modified session",  
  "modify": true,  
  "values": ".....",  
  ...  
}
```

## 9.4 Deleting a session

Simply send a **delete\_session** with the name of the session, i.e:

```
{  
  "delete_session": "Session One"  
}
```

## 9.5 Deleting all sessions

Simply send a **reset\_sessions** with a value of “true”:

```
{  
  "reset_sessions": true  
}
```

## 10. External loss attenuators

### 10.1 Getting the external loss attenuators list

Send the field **get\_external\_losses** with a value of "true":

```
{
  "get_external_losses": true
}
```

It will return a **external\_losses** object with an array of external loss attenuators with the following format:

```
"External loss attenuator name": [ [ Frequency (Mhz), Line (dB), Neutral (dB) ], [ Frequency (Mhz), Line (dB), Neutral (dB) ], ... ]
```

For example:

```
{
  "external_losses": [
    {
      "External loss attenuator 1": [ [ 1, 2, 3 ], [ 2, 3, 4 ], ... ]
    },
    {
      "External loss attenuator 2": [ [ 10, 20, 30 ], [ 20, 30, 40 ], ... ]
    },
    ...
  ]
}
```

### 10.2 Creating an external loss attenuator

To create an external loss attenuator, send an **external\_loss\_name** field with the name of the new external loss attenuator, a **modify** field set to "false" and a **values** array in the same format we described before when receiving the external loss attenuators list.

```
{
  "external_loss_name": "New External loss attenuator",
  "modify": false,
  "values": [
    [ "1", "2", "3" ],
    [ "10", "20", "30" ],
    ...
  ]
}
```

### 10.3 Editing an external loss attenuator

Send the same values as creating it, but with the field *modify* set to *true* and an additional field, *original\_external\_loss\_name*, just in case we wanted to rename the original *external\_loss\_name*, i.e:

```
{
```

```

"original_external_loss_name": "New External loss attenuator",
"external_loss_name": "Modified External loss attenuator",
"modify": true,
"values": [
  ["11", "22", "33"],
  ["21", "22", "23"],
  ...
]
}

```

## 10.4 Deleting an external loss attenuator

Simply send a **delete\_external\_loss** with the name of the external loss attenuator, i.e:

```

{
  "delete_external_loss": "External loss attenuator 1"
}

```

## 11. Report lists

To generate a report list, we must send 3 values: *standard* (the standard name string), *subranges* (number of subranges, an integer) and *margin* (the margin, an integer), i.e:

```

{
  "standard": "One standard",
  "subranges": 10,
  "margin": 10
}

```

The answer to the request will be a boolean *frase* and a *report* object with an array of arrays in the following order:

```

[
  Marker (integer),
  Frequency in Mhz (float),
  Peak level in dBuV (float),
  Quasi-Peak level in dBuV (float),
  Quasi-Peak limit in dBuV (float),
  Distance to Quasi-Peak limit [dB] (float),
  Average level in dBuV (float),
  Average limit in dBuV (float),
  Distance to Average limit [dB] (float),
  Channel (string),
  Compliance (PASS/FAIL)
]

```

The *frase* will be **true** if fewer than 6 emissions are within 10dB of the observed limit, and **false** otherwise.

For example:

```

{
  "report": [
    [ 1, 5, 116.964216, 116.418941, 119.000000, 10.000000, 116.480995, 109.000000, 10.000000,
      "L", "PASS" ],
    [ 2, 6, 313.963336, 133.418211, 1123440000, 111220000, 334.440995, 239.020000, 30.000000,

```



```
"N", "FAIL" ],  
    ...  
],  
"frase": true  
}
```

## 12. Miscellaneous

### 12.1 Getting the activated licenses

Send the field **get\_licenses** with a value of "true":

```
{  
  "get_licenses": true  
}
```

As a response you will get a **licenses** object with an array with the name of each activated licenses:

```
{"licenses": [ "emi", "osc" ]}
```

### 12.2 Getting the current PCB and FPGA temperature

Send the field **get\_temps** with a value of "true":

```
{  
  "get_temps": true  
}
```

As a response you will get a **temperatures** object with an array with the celsius degrees. The first value is for the PCB, the second one for the FPGA:

```
{"temperatures": [ 45.12345, 50.12345 ]}
```

From:  
<https://emzer.com/wiki/> -

Permanent link:  
[https://emzer.com/wiki/doku.php?id=public:emscope:websocket\\_api](https://emzer.com/wiki/doku.php?id=public:emscope:websocket_api)

Last update: **2020/12/02 11:45**

