PowerSpy JSON format

The fields in the JSON format are described in the following table.

Common buffer parameters

Common	Mandatory	Default value	Notes
version	Yes	-	String containing the PowerSpy's JSON format version number. The current version is "2.0".
type	No	"analog"	Buffer type (not case sensitive): "analog" | "digital" | "frequency" | "parametric" | "table"
source	No	empty string	Device type: "fgc", "ccrt", etc...
device	No	file name	If the data is coming from a local file, the default value will be the file name without the .json or .csv suffix (if present) and with spaces, colons or commas replaced by underscores, full stops and semicolons respectively. The device will appear in the monitor title (DEVICE : BUF_NAME) and will also prefix the signal name in the tooltips (DEVCE : SIGNAL_NAME). The webserver will always specify the device in the JSON data it returns to PowerSpy.
name	No	empty string	The buffer name will appear in the monitor title (DEVICE : BUF_NAME).

Specific buffer parameters

Time Series type: "analog" | "digital"	Mandatory	Default value	Notes
signals	Yes	-	Array of objects holding the signal data.
cycleSelector	No	0	Cycle selector that applied to the acquisition of the data. For non-cyclic acquisitions, this is 0. For cyclic acquisition, it can be 1-N for cyclic or the cycle selector name as a string.
timeOrigin	No	firstSampleTime	Time origin for all the signals in the buffer in UTC Unix time. This is the time that will appear as 0.0 when PowerSpy is in relative time mode. In this mode, the time origin will be subtracted from the time of every sample before display.
firstSampleTime	No	0 (1/1/1970)	Time of first sample in UTC Unix time.
period	No	1	Sampling period (s). This should not be included if timestamps are.
timestamps	No	-	Optional sample timestamps - this should not be included if period is defined. If the samples do not have a fixed period, then a timestamps vector should be provided with the same number of elements as the samples vectors.
Frequency Series type: "frequency"	**Mandatory**	**Default value**	**Notes**
subtype	Yes	-	Must be either "bode" or "group-delay"
signals	Yes	-	Array of objects holding the signal data.
frequencies	Yes	-	Sample frequency stamps (Hz). This vector should be provided with the same number of elements as the samples vectors.
Parametric Series type: "parametric"	**Mandatory**	**Default value**	**Notes**

signals	Yes	-	Array of objects holding the signal data.
Table parameters type: "table"	**Mandatory**	**Default value**	**Notes**
subtype	No	"default"	This is to specify the type of the table. Possibilities are "default" | "event-log" | "cycle-log" | "timing-log" | "error-log" | "bode" | "group-delay" | comparison" | "properties". Properties is a case of table containing FGC properties and values which can be send directly to the converter.

Signal parameters

Time Series parameters type: "analog" | "digital" subtype:	**Mandatory**	**Default value**	**Notes**
name	Yes	-	Signal name. Any spaces will be replaced by underscores (_). Commas will be replaced by semicolons.
step	No	false	For an analog signal, when step=true, the signal will be displayed with trailing step interpolation, instead of linear interpolation. This should not be specified for digital signals, which are always shown with trailing step interpolation.
timeOffset	No	0	Time offset for the signals in seconds. The time of a sample will be calculated from firstSampleTime + period * sample_index + timeOffset. So a positive time offset will move the point later and a negative time offset will move it earlier.
samples	Yes	-	Signal samples in time order. For digital signals, the values should be 0 or 1 only. For analog signals, any valid double-precision floating point value is accepted. The number of samples must be the same for all signals in a buffer and must be at least two. If timestamps are provided, then the number of sample for all signals must match the number of timestamps provided.
Bode Series type: "frequency" subtype: "bode"	**Mandatory**	**Default value**	**Notes**
name	Yes	-	Signal name. Any spaces will be replaced by underscores (_). Commas will be replaced by semicolons.
firstSampleOffset	No	0	Sample offset relative to the frequency array defined at the buffer level. Must be positive or zero. The first sample of the gain and phase arrays gain[0] and phase[0] correspond to frequencies[firstSampleOffset]
gain	Yes	-	Gain (dB) for signal samples. Any valid double-precision floating point value is accepted. The number of samples must be equal or lesser than the number of frequency samples.
phase	Yes	-	Phase (degrees) for signal samples. Any valid double-precision floating point value is accepted. The number of samples must be equal or lesser than the number of frequency samples.
Group Delay Series type: "frequency" subtype: "group-delay"	**Mandatory**	**Default value**	**Notes**
name	Yes	-	Signal name. Any spaces will be replaced by underscores (_). Commas will be replaced by semicolons.
firstSampleOffset	No	0	Sample offset relative to the frequency array defined at the buffer level. Must be positive or zero. The first sample of the gain and phase arrays gain[0] and phase[0] correspond to frequencies[firstSampleOffset]
samples	Yes	-	Values (ms) for signal samples. Any valid double-precision floating point value is accepted. The number of samples must be the same for all signals in a buffer and must be at least two.
Parametric Series type: "parametric" subtype:	**Mandatory**	**Default value**	**Notes**
name	Yes	-	Signal name. Any spaces will be replaced by underscores (_). Commas will be replaced by semicolons.
x	Yes	-	X values. Any valid double-precision floating point value is accepted. X and y arrays must be of the same length.
y	Yes	-	Y values. Any valid double-precision floating point value is accepted. X and y arrays must be of the same length.

Notes: The JSON data will be rejected if:

- any mandatory fields are missing, or
- any fields are repeated, or

- any constraint listed above is not respected, or
- incompatible fields are provided (e.g. period and timestamps or frequency_step and frequencies).

Analog Buffer

Version 2.0 of the PowerSpy JSON format can encode analog time series signals as illustrated in this example:

```
{
  "version":"2.0",
  "type":"analog",
  "source":"fgc",
  "device":"SYSTEM_NAME",
  "name":"BUFFER_NAME",
  "cycleSelector":"LHCPILOT",
  "timeOrigin":1458137212.000000,
  "firstSampleTime":1458137212.000000,
  "period":1.00000E-04,
  "signals":[
    {
      "name":"SIGNAL1",
      "samples":[10.1,11.5E+3,-12.2E-2]
    },
    {
      "name":"SIGNAL2",
      "step":true,
      "timeOffset":0.500000,
      "samples":[-5,-3,1]
    },
    {
      "name":"SIGNAL3",
      "timeOffset":-2.500000,
      "samples":[1,2,3]
    }
  ]
}
```

Digital Buffer

Version 2.0 of the PowerSpy JSON format can encode digital time series signals as illustrated in this example:

```
{
  "version":"2.0",
  "type":"digital",
  "source":"ccrt",
  "device":"SYSTEM_NAME",
  "name":"BUFFER_NAME",
  "cycleSelector":"21",
  "timeOrigin":1458137212.000000,
  "firstSampleTime":1458137212.000000,
  "timestamps":[1.00000E-04,3.00000E-04,4.00000E-04],
  "signals":[
    {
      "name":"SIGNAL1",
      "timeOffset":0.000100,
      "samples":[0,0,1]
    },
    {
      "name":"SIGNAL2",
      "samples":[1,0,1]
    },
    {
      "name":"SIGNAL3",
      "samples":[0,1,1]
    }
  ]
}
```

Frequency: Bode Buffer

Version 2.0 of the PowerSpy JSON format can encode complex frequency series signals as illustrated in this example:

```
{
  "version":"2.0",
  "type":"frequency",
  "subtype":"bode",
  "source":"tfa",
  "device":"SYSTEM_NAME",
  "name":"BODE",
  "frequencies":[1.00000E+00,2.00000E+01,4.00000E+01],
  "signals":[
    {
      "name":"SIGNAL1",
      "gain":[-10.5,-24.6,-54.32],
      "phase":[-5.1,-35.2,-80.4]
    }
  ]
}
```

In this case, the samples have non-equally spaced frequencies

Frequency: Group Delay Buffer

Version 2.0 of the PowerSpy JSON format can encode complex frequency series signals as illustrated in this example:

```
{
  "version":"2.0",
  "type":"frequency",
  "subtype":"group-delay",
  "source":"tfa",
  "device":"SYSTEM_NAME",
  "name":"GROUP_DELAY",
  "frequencies":[1.00000E+00,2.00000E+01,4.00000E+01],
  "signals":[
    {
      "name": "SIGNAL1",
      "firstSampleOffset": 1,
      "samples": [-24.6,-54.32]
    },
    {
        "name": "SIGNAL1",
        "firstSampleOffset": 1,
        "samples": [-24.6,-54.32]
    }
  ]
}
```

In this case, the samples have non-equally spaced frequencies

Parametric Buffer

Version 2.0 of the PowerSpy JSON format can encode complex frequency series signals as illustrated in this example:

```
{
  "version":"2.0",
  "type":"parametric",
  "source":"tfa",
  "device":"SYSTEM_NAME",
  "name":"PARAMETRIC",
  "signals":[
    {
      "name": "SIGNAL1",
      "x": [0.1, 2, 3],
      "y": [0.001, 4, 9]
    }
  ]
}
```

Event Log

Version 2.0 can encode event logs as illustrated in this example:

```
{
  "version":"2.0",
  "type":"table",
  "subtype":"event-log",
  "source":"FGC",
  "device":"SYSTEM_NAME",
  "name":"BUFFER_NAME",
  "table": {
    "rows":[
      {
        "timestamp":1464722837.172000,
        "cells":["STATUS.ST_UNLATCHED","START_EVENT","SET_BIT","+"]
      },
      {
        "timestamp":1464722837.272000,
        "cells":["STATUS.ST_UNLATCHED","START_EVENT","CLR_BIT","+"]
      },
      {
        "timestamp":1464722837.242000,
        "cells":["STATE.PC","RUNNING","SET"," "]
      }
    ]
  }
}
```

The rows are sorted by timestamps automatically in PowerSpy. If there is a double quote in any of the cells, it is escaped with a backslash in JSON:

```
{"data": "\"Alrighty then!\""}
```