UMT calibration JSON file format Version 1.0

Table of contents

Format general description

Some of our customers prefer to parse ASCII-like formats for calibration for ease or human reading.

For this reason EMpower will enable exporting to JSON format, which is an ASCII-based format, with the advantage that most modern languages will have already a library to parse JSON files, making the implementation of a reader for these files easy, since no special parsers have to be programmed in most cases. Please consult the list of languages and libraries at https://www.json.org/.

File names

The calibration file names exported to JSON format will have the following structure:

`AAAAA_FFFFFFFF.[X].json`

Where "`AAAAA_FFFFFFFF.[X]`" is the name of the calibration file being exported, represented in this structure:

- ***AAAAA*:** Sensor serial number for sensor calibrations, or receiver serial number for receiver calibrations
- ***FFFFFFFF*:** Calibration start date and time as an epoch, in seconds since 00:00:00, January 1, 1970, GPS time base, converted to hexadecimal
- ***[X]*:** File extension.
  - **`.scal:`** sensor calibration
  - **`.rxcal:`** receiver calibration

For example:

An exported sensor calibration for sensor 53880, starting at January 2, 2019, 3:00.PM [GPS time] would have the name:

`53880_5C2CD1F0`**`.scal`**`.json`

An exported receiver calibration for receiver 10125, starting at January 2, 2019, 3:00.PM [GPS time] would have the name:

`10125_5C2CD1F0`**`.rxcal`**`.json`

Field descriptions

Main header

- **manufacturer**: "Phoenix Geophysics"
- **file_type**: “sensor calibration” or “receiver calibration”
- **file_version**: The version for this type of file
- **timestamp_gps**: The timestamp (i.e. GPS-based epoch from 00:00:00, January 1, 1970) of this calibration start time
- **empower_version**: The string describing the version of EMpower used to generate this file
- **instrument_type**: The commercial name of the receiver used to acquire this time series (for instance MTU-8A or MTU-5C)
- **instrument_model**: The technical model name for the receiver used to generate the calibration (for instance RMT01 represents one assembly model for an MTU-5C)
- **sensor_serial**: The sensor serial number. Optional - only for sensor calibrations
- **inst_serial**: The serial number of the receiver used to generate the calibration
- **altitude**: Altitude coordinate in meters
- **latitude**: Latitude coordinate in Decimal Degrees format
- **longitude**: Longitude coordinate in Decimal Degrees format
- **num_channels**: The total number of active electric and magnetic channels recorded in this calibration, may vary depending on the model of the receiver.
    - One-axis sensor calibrations will always have 1 channel.
    - Receiver calibrations will have data on all available channels unless the receiver is operating incorrectly.

Section “cal_data”

The **cal_data** tag contains an array of JSON objects which represent the calibration response. Each channel contains a JSON array of at least one response curve. Each response curve consists of a JSON array of values for frequency, magnitude, and phase.

“cal_data” fields

A JSON array containing information about each channel. Each item in the array has the following fields:

- **tag:** The channel the calibration was performed on
    - Any of *E1-E5, H1-H6*, depending on the receiver model
- **num_of_responses:** The number of response curves associated with this channel.
    - A sensor calibration will only have one response curve
    - A receiver calibration will have one for each low pass filter. Curves are ordered by their frequency, and differ by receiver model
        - *MTU-5C, MTU-8A, RXU-8A, MTU-2C*: 10 KHz, 1 KHz, 100 Hz, 10 Hz
        - *MTU-5D*: 17.8 KHz, 10 KHz, 1 KHz, 10 Hz
- **chan_data:** The array of data about each response curve associated with this channel

“chan_data” fields

A JSON array containing information about the response curves for a channel. Each item in the array has the following fields:

- **num_records**: The number of records in the response curve
- **freq_Hz**: The array of frequencies of this response
- **magnitude**: The array which will hold the following: For sensor calibrations, *the flat part of this curve (or top part of the curve for MTC-50 type of sensors) will represent the nominal gain of the sensor.*. For receiver calibrations, this is *normalized to 1.*
- **phs_deg**: The array of phase values in degrees (°) of this response

Data values

Values are packed in a JSON array for each response curve. The values can be written in scientific notation to allow for best precision using a minimum amount of characters. Fixed notation might be used when it makes sense.

Example


```
{
  "manufacturer": "Phoenix Geophysics",
  "file_type": "receiver calibration",
  "file_version": "1.0",
  “timestamp_utc”: 1496950038
  "empower_version": "1.27.0.1:1.27.0.3",
  "instrument_type": "MTU-5C",
  “instrument_model”: “RMT01”,
  “inst_serial”: “10022”,
  “altitude”: 1045.2454833984375,
  “latitude”: 37.2016716003418,
  “longitude”: -114.69085693359375,
  “cal_data”: [
    {
      “tag”: “E1”,
      “num_of_responses”: 2,
      “chan_data”: [
        {
          “num_records”: 10,
          “freq”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0],
          “magnitude”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0],
          “phs_deg”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0]
        },
        {
          “num_records”: 10,
          “freq”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0],
          “magnitude”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0],
          “phs_deg”: [0, 1, 0, 1, 1, 0, 1, 0, 1, 0]
        }
      ]
    }
  ]
}
```

Annex A

Parsing the file

A large JSON file might not appear easy to parse using common libraries, but there might be libraries designed to parse JSON as a stream (as per this link), or alternatively, you can parse the headers manually, and then pass the internal “data” vectors through a library, allowing the programmer to implement a streamed reader by partially using existing libraries.

Note that although this is not conventional **JSON**, we have made the data arrays, for instance, in the example below:

```
"freq_Hz": [0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1],
```

be a single line in the file.

This means that from the tag describing the channel frequency data **(“freq_Hz”)** to the closing bracket **‘]’** and the following **comma,** there will not be a carriage return character. This will make it easy to pass these objects to a JSON library from languages that can retrieve files line by line, making it easier for a programmer to create a streamed reader instead of a monolithic all-at-once JSON parser.

Also, the curly braces of the object containing the data vector are in their own line *(the closing bracket may only be followed by a comma when necessary).* In this way, the streamed reader can scan for opening or closing braces to separate objects to be parsed.

NOTE: the format described in this document is only guaranteed immediately after exporting from EMpower within a local filesystem that does not reformat **ASCII** files (e.g. that does not try to append new lines or carriage returns for long lines), or otherwise modified by an end-user or a 3rd party program.
Transferring the calibration JSON files over the internet result in the reinterpretation and/or formatting the JSON file to another valid representation of the same stream. If the file is to be transferred via e-mail or other network protocols, it is recommended to first compressing the file to ensure that the JSON file is not re-formatted in the transit.