Reporting Tool Config Guide

A quick summary of the required fields in a new file configuration for the burn-in reporting tool including common issues when relevant.

# “name”:

A descriptive name for this configuration. Not required to be unique, however a unique name could be convenient when checking logging files. As a rule, the “name” field should match the name of the file.

# “file\_identifier”:

A regex string used to check the type of file being parsed.

# “header\_identifier”:

A regex string used to scan for the end of the file header.

## REGEX tips

To start learning about regular expressions I recommend the following article found on the Python docs: <https://docs.python.org/3/howto/regex.html#regex-howto>.

For a more detailed run down on the syntax try: <https://docs.python.org/3/library/re.html>.

**ALL forward slashes “\” -> “\\” must be doubled up.** This is because “\” is used as an escape character for special characters in C++ strings, so it must be escaped itself.

Common regular expression examples:

* Match 0+ whitespace characters (“ “, \r, \n, \t, \v, \f): “[\\s]\*”.
* Match 1+ numerical digits (0-9): “[\\d]+”.

# “delim”:

A string indicating the delimiter between data values.

# “titles”:

An array of strings containing the column titles which will be displayed in the reporting tool.

# “types”:

An array of integers indicating the data type of each column.

1. Integer
2. Double
3. String

3+) Invalid

# “start\_time”:

JSON object containing two key-value pairs, **“method”** and **“params”**. Used to parse start time of data acquisition for current file.

### method: “in\_header”

params:

* “re\_pattern”: Regex pattern. Used to scan the header for a start time. E.g., ";First Pulse Arrived : ([\\d]+/[\\d]+/[\\d]+ at [\\d]+:[\\d]+:[\\d]+)[\\s]\*"
* “time\_pattern”: C++ <chrono> format string (<https://en.cppreference.com/w/cpp/chrono/parse>) . E.g., "%d/%m/%Y at %H:%M:%S" matches times like “12/03/2022 at 13:23:12”.

### method: “in\_file\_path”

params:

* “re\_pattern”: Regex pattern. Used to scan file path for a start time.
* “time\_pattern”: C++ <chrono> format string (<https://en.cppreference.com/w/cpp/chrono/parse>)

### method: “in\_data”

params:

# “interval”

JSON object containing two key-value pairs, **“method”** and **“params”**. Used to parse interval between measurements for current file.

### method: “in\_header”

params:

* “re\_pattern”: Regex pattern. Used to scan the header for a measurement interval.
* “time\_pattern”: C++ <chrono> format string (<https://en.cppreference.com/w/cpp/chrono/parse>)

### method: “value”

params:

* “increment”: Double indicating interval between measurements in seconds.

### method: “automatic”

params:

* “title”: String indicating a column to detect interval period from. Must match a value from the “titles” field. If the measurement period is not consistent, will fail to parse the file.

# “trim\_filter\_key”

If some aspects of the software or hardware fails during a life test, it may lead to a large amount of empty data being recorded. To remove this empty data two pieces of information are needed. The first is a filter key specified in the config file, e.g: “trim\_filter\_key”: “PDOF 1”. The second piece of information is the max time the laser is expected to be on for while cycling on/off, this value can be changed in the “Settings” page of the reporting tool (It’s a good idea to add a small bit of extra time to this value). If you don’t want to apply filtering, change the “Trim Blank Data” value found on the “Settings” page of the reporting tool to FALSE. If you provide an empty string or a filter key which doesn’t exist, the tool will continue as if it is set to FALSE.