Structs DSV Format

The XINA Structs DSV (delimiter separated values) format provides a standard delimited text data file format. This is recommended for data files attached to events, and forms the basis for the structs buffer file format.

Files have certain standard requirements:

- Must be UTF-8 encoded
- New lines will be interpretted from either `\n` or `\r\n`
- Blank lines will be ignored
- Lines starting with the `#` character are treated as comments and ignored

The `conf` object may define other customization of the format:

Key	Value	Default	Description
delimiter	string	`','` (comma character)	value delimiter
quote_char	character	`"` (double quote character)	value quote character
ignore_lines	number	`0`	lines to ignore at the start of the file
zone	string	UTC	time zone to use if not provided
values	JSON object		preferred interpretation of string literals (see below)

It is strongly recommended to include a unique appropriately generated 128-bit UUID in the standard 36 character format as a comment in the first processed line of each file. (If `ignore_lines > 0`, this would be the first line after that number of lines.)

There are two format modes for DSV files: the **row** format, in which each line contains a time, label, and value, and the **column** format, in which each row contains a time and one or more values. The first processed uncommented line will be intepretted as the column header, which is used to determine the file format. The file will be treated as row mode if it contains exactly three columns, with each having one of the reserved column names in the table below.

Name	Description	Alternate Names
t	Unix time or ISO8601 zoned timestamp	ts, time, timestamp, datetime, unix_time, unix, utc
k	key	key, m, m_id, mn, mn_id, mnemonic, mnemonic_id, n, name
v	value (numeric, empty, or `null`)	val, value

The header is used to determine the order of the columns.

For example (whitespace added for clarity, not required):

```
# 123e4567-e89b-12d3-a456-426614174000
t , k    , v
0 , v_mon , 1
0 , i_mon , 5
1 , t_mon , 100
2 , v_mon , 1.1
2 , i_mon , 4
3 , t_mon , null
4 , v_mon , 1.2
4 , i_mon , 3
5 , t_mon , 101
```

Otherwise, the file will be interpretted as the column format, where the first must be the time column, followed by a column for each mnemonic. The column headers must specify the mnemonic name or ID for each column.

For example, the following is equivalent to the above example (whitespace added for clarity, not required):

```
# 123e4567-e89b-12d3-a456-426614174000
t , v_mon , i_mon , t_mon
0 , 1     , 5     ,
1 ,       ,       , 100
2 , 1.1   , 4     ,
3 ,       ,       , null
4 , 1.2   , 3     ,
5 ,       ,       , 101
```

Time Parsing

The mode of time processing is determined by the value for `t` in `conf`. The `auto` mode attempts to interpret the most likely formatting for the timestamp. If the value is an integer or floating point format, it will be interpretted as a Unix timestamp, with precision based on these rules:

- t > `1e16`: error, value above typical range
- t > `1e14`: microseconds
- t > `1e11`: milliseconds
- t > `1e8`: seconds
- t <= `1e8`: error, value below typical range

Otherwise it will be interpretted as a zoned ISO8601 timestamp. If `t` is set explicitly in the configuration the time will always be interpretted in that context. The ISO timestamp may use the standard format:

`2023-05-31T17:55:07.000`

Or condensed format:

`20230531T175507.000`

If the `zone` property provided in the configuration, the timestamps do not require a zone. Otherwise they must include an explicit zone.

Non-Numeric Values

Values which are non-numeric may be ignored, treated as `null`, or mapped to explicit values using the `values` property of the `conf` object. Ignored values are treated by XINA as though they do not exist in the file. `null` values are stored as an actual data point in the XINA database, but with the value `null` instead of a numeric value. (This is primarily useful to create a visual gap in plots.)

The following values are **ignored** by default (note, case-insensitive and whitespace agnostic):

- `""` (empty string)
- `"nv"` (no value, ITOS)
- `"na"`
- `"n/a"`

The following values are interpretted as `null` by default (note, case-insensitive and whitespace agnostic):

- `"null"`
- `"nil"`
- `"none"`
- `"nan"`
- `"inf"`
- `"+inf"`
- `"-inf"`
- `"infinity"`
- `"+infinity"`
- `"-infinity"`

Custom value interpretations may be specified in the values object as either `"ignore"`, `null`, or a numeric value. For example:

```
{
 "?": "ignore",
 "notta": null,
 "onetwothree": 123
}
```

Any text value which does not include a custom or default mapping will cause an error. The defaults may be extended in the future.

Revision #43
Created 28 July 2022 20:31:12 by Nick Dobson
Updated 23 December 2024 19:36:04 by Nick Dobson