Events

To organize time based data in XINA, we employ **events**, which come in two forms: **instants**, referring to a **single moment in time**, and **intervals**, referring to a **range of time**. The goal of events is to make it easy to find, compare, and trend data. Each has their own databases and include fields for:

- **type** (indicates how the event should be viewed and interpreted)
- **UUID** (universally unique identifier, generated at the creation of the event)
- numeric **event ID** (meaning can depend on type)
- plain text **label** (up to 128 bytes)
- plain text, HTML, or JSON **content**
- optional JSON object **metadata**

The UUID uniquely identifies an event, and is the only way to permanently, globally specify it. It should be applied at the time of creation to ensure consistency even if data is reprocessed. The event ID is optional, and can be used as needed (when not provided it will be zero by default). Its much faster and more reliable to query numbers than text, so this is the best way to indicate events having commmon meaning.

Event Database

Default Location

`<model>.event`

`<model>.eventf` (single file per event)

`<model>.eventfs` (multi file per event)

Required Fields

field	type	description
`uuid`	`uuid`	UUID
`e_id`	`int(8)`	event ID
`t_start`	`instant(us)`	start time (inclusive)
`t_end`	`instant(us)`	end time (exclusive)
`dur`	`duration(us)`	`t_end` - `t_start`
`interval`	`boolean`	`true` if event is an interval, `false` if event is an instant
`open`	`boolean`	`true` if event is an open interval, `false` otherwise
`type`	`struct_event_type`	event type (see below)
`level`	`struct_event_level`	event level (see below)
`label`	`utf8vstring(128)`	plain text label
`content`	`utf8text`	extended text / CSV / HTML / JSON

field	type	description
`meta`	`jsonobject`	additional metadata as needed
`conf`	`jsonobject`	additional information specific to `type`

Note that `duration`, `interval`, and `open` are **computed automatically** from `t_start` and `t_end` and **cannot be provided manually**.

Event Types

XINA defines a fixed set of standard event types, each with an associated numeric code. The type is stored as the code in the database for performance reasons; for practical purposes most actions can use the type name directly, unless interacting directly with the API.

Standard Types

code	name	ins	int	description
`0`	`message`	?	?	Basic event, IDs optional, no implicit ID interpretation
`1`	`marker`	?	?	Organized event, IDs imply related events
`2`	`alert`	?	?	Organized event, level (severity) required, IDs imply related events
`2000`	`test`		?	Discrete test period, may not overlap other tests, IDs optional, unique if used
`2001`	`activity`		?	Discrete activity period, may not overlap other activities, IDs optional, unique if used
`2002`	`phase`		?	Discrete phase period, may not overlap other phases, IDs optional, unique if used
`3000`	`data`	?	?	General purpose data set
`3001`	`spectrum`	?	?	General purpose spectrum data

Additional types will be added in the future as needed, with codes based on this chart:

Standard Type Code Ranges

code	ins	int	description
`0-999`	?	?	General types for instants and intervals
`1000-1999`	?		General types for instants only
`2000-2999`		?	General types for intervals only

code	ins	int	description
`3000-3999`	?	?	Data set types for instants and intervals
`4000-4999`	?		Data set types for instants only
`5000-5999`		?	Data set types for intervals only

Data Format

The `data` event type indicates a basic data set. This is typically used with the single file per event database structure, in which case the file will contain the data set. For event databases without files, the data is expected to be stored in the `content` field. This is only recommended for small datasets (less than 1MB).

Files must be either ASCII or UTF-8 encoded. New lines will be interpretted from either `\n` or `\r\n`. The `conf` object may define other customization of the format:

Conf Definition

Key	Value	Default	Description
`delimiter`	`string`	auto detect (`','`, `'\t'`, `';'`)	value delimiter
`quoteChar`	`character`	`"` (double quote character)	value quote character
`ignoreLines`	`number`	`0`	number of lines to skip before the header
`invalid`	`null`, `'NaN'`, `number`	`null`	preferred interpretation of invalid literal
`nan`	`null`, `'NaN'`, `number`	`null`	preferred interpretation of `'Nan'` literal
`pInfinity`	`null`, `'Inf'`, `number`	`null`	preferred interpretation of positive `'Infinity'` literal
`nInfinity`	`null`, `'Inf'`, `number`	`null`	preferred interpretation of negative `'Infinity'` literal
`utc`	`boolean`	`false`	if `true`, interpret all unzoned timestamps as UTC

Starting after the number provided for `ignoreLines`, the content must include a header for each column, with a name and optional unit in parentheses. Special standard unit names may be used to indicate time types, which will apply different processing to the column:

Unit	Description
`ts`	text timestamp, interpretted in local browser timezone (absent explicit zone)
`ts_utc`	text timestamp, interpretted as UTC timezone (absent explicit zone)
`unix_s`	Unix time in seconds
`unix_ms`	Unix time in milliseconds
`unix_us`	Unix time in microseconds

Revision #30
Created 27 July 2022 18:13:41 by Nick Dobson
Updated 1 May 2024 13:15:43 by Bradley Tse