Struct Definitions Reference

This page provides a reference for the purpose and structure of all structs standard groups and databases.

Structs groups and databases are marked with a JSON object using the key `xs_struct`. This contains three standard parameters:

- `"type"` - the name of the type of struct element as a string
- `"v"` - the current version this instance of the specified type as a string
- `"conf"` - optional JSON object, format depends on type

Versioning is tied to the version number of the XINA server. Typically server updates will automatically apply needed changes to all structs schema elements, incrementing their `"v"` property to the latest version. In the event an upgrade cannot be performed the version will not be changed.

Groups

Note that group versioning is used to manage databases required within groups.

Project

Top level struct group. All struct groups and databases must be decendants of a project to be recognized. Name and label are customizable.

Created with the STRUCT CREATE PROJECT action.

Struct Parameters

Parameter	Value
type	`"project"`

Group Parameters

Parameter	Value
name	*
label	*

Category

Mid-level struct group for organization. Must be a child of a project or category group. Name and label are customizable.

Created with the STRUCT CREATE CATEGORY action.

Struct Parameters

Parameter	Value
type	`"category"`

Group Parameters

Parameter	Value
name	*
label	*

Model

Group for which all data is locally co-relevant. Must be a child of either a project or category group. Name and label are customizable.

Created with the STRUCT CREATE MODEL action.

Struct Parameters

Parameter	Value
type	`"model"`

Group Parameters

Parameter	Value
name	*
label	*

Pipe

Group for all data from a single pipe. Must be the child of a model group. Name and label are customizable.

Created with the STRUCT CREATE PIPE action.

Struct Parameters

Parameter	Value	Default
type	`"pipe"`	

Conf Parameters

Parameter	Value	Default
discrete	boolean	`false`
buffer	boolean	`false`
variable	boolean	`false`
condense	boolean	`false`

Parameter	Value	Default
duration	archive length in minutes	`60`
partition	`{"from": <y0>, "to": <y1>}`	

Group Parameters

Parameter	Value
name	*
label	*

Notes

If `discrete` is `true`, mnemonic data is not considered persistent between archives, and open/close interval operations are not supported.

If `buffer` is `true`, the `mn_buffer` database will be generated, and the pipe will be included in automated archive tasks. Otherwise, the STRUCT BUFFER IMPORT action will not be supported.

If `variable` is `true`, mnemonic and event databases in the pipe will include the archive ID field (`a_id`). If `buffer` is `true`, `variable` must be `false` (since buffer-generated archives cannot be variable).

`duration` specifies the archive length in minutes, if `variable` is false. This cannot be changed. The default is 60 minutes (1 hour). A shorter window may be appropriate for very high data volumes. The maximum is 1440 (24 hours), and the value must be evenly divisible into 1440.

Changelog

11.0.0 (planned)

- renamed from `origin` to `pipe`

Definitions

Group containing definitions databases.

Struct Parameters

Parameter	Value
type	`"def"`

Group Parameters

Parameter	Value
name	`"def"`
label	`"Definitions"`

Changelog

11.0.0 (planned)

- add filter definitions database

Task

Group containing task tracking databases. Must be a child of a pipe group.

Struct Parameters

Parameter	Value
type	`"task"`

Group Parameters

Parameter	Value
name	`"task"`
label	`"Task"`

Mnemonic

Group containing mnemonic data databases. Must be a child of a pipe group.

Struct Parameters

Parameter	Value
type	`"mn"`

Group Parameters

Parameter	Value
name	`"mn"`
label	`"Mnemonic"`

Mnemonic Bin

Group containing binned mnemonic data databases. Must be a child of a mnemonic group.

Struct Parameters

Parameter	Value
type	`"mn_bin"`

Group Parameters

Parameter	Value
name	`"bin"`
label	`"Bin"`

Databases

Structs databases typically specify a set of required fields, and may permit the inclusion of additional custom fields. Changes to the spec involving fields will usually be treated as minor version changes, though they may require manual user correction if an added field conflicts with a custom field already present in a particular database instance.

Note that fields marked as **virtual** are calculated from the values of other field(s) and cannot be populated or edited manually.

Definitions

All definitions databases must be direct children of a definitions group, and all definitions groups must contain one of each definition database.

Diagram Definitions

Holds diagram definitions. The diagram itself is in an attached SVG file.

Struct Parameters

Parameter	Value
type	`"def_diagram"`

Database Parameters

Parameter	Value
name	`"diagram"`
label	`"Diagram"`
format	`"{name}"`
order	(`name` , desc)
singular	`"diagram"`

Fields

Name	Type	Req	Description
name	`utf8vstring(128)`	?	unique conf name
desc	`utf8text`		plain text description
file_name	`utf8filename`	?	file name
conf	`jsonobject`		diagram configuration
meta	`jsonobject`		additional metadata as needed

Changelog

11.0.0

- order changed to (`name`, asc)
- added `desc` field

Event Definitions

Holds event definitions, specifying how they are displayed, interpretted and processed. Filter Definition fields are defined in the `conf` field.

Struct Parameters

Parameter	Value
type	`"def_event"`

Database Parameters

Parameter	Value
name	`"event"`
label	`"Event"`
format	`"{name}"`
order	(`name`, asc)
singular	`"event definition"`

Fields

Name	Type	Req	Description
e_id	`int(4)`	?	unique ID
name	`utf8vstring(128)`	?	unique name
desc	`utf8text`		plain text description
meta	`jsonobject`		additional arbitrary metadata
conf	`jsonobject`		configuration for pseudo-events
aliases	`set(utf8string)`		alternative name(s)
ext_id	`asciivstring(64)`		external ID

Filter `conf` jsonobject

Name	Type	Req	Description
type	`utf8vstring(128)`	?	Must have a value of "filter"
condition	`utf8text`	?	filter condition expression
t_start_offset	`duration(us)`		start time offset (`0` if not provided)
t_end_offset	`duration(us)`		end time offset (`0` if not provided)

Name	Type	Req	Description
models	`set(asciistring)`		Models that the filter will apply to

Changelog

11.2.0

- added Filter `conf` specification (no structural change)

11.0.0

- added `aliases` field
- added `ext_id` field
- removed `type` field
- changed `e_id` type from `int(8)` to `int(4)`

Mnemonic Definitions

Holds mnemonic definitions, specifying how they are displayed, interpretted and processed.

Struct Parameters

Parameter	Value
type	`"def_mn"`

Database Parameters

Parameter	Value
name	`"mn"`
label	`"Mnemonic"`
format	`"{name} ({unit})"`
order	(`name`, asc)
singular	`"mnemonic definition"`

Fields

Name	Type	Req	Description
mn_id	`int(4)`	?	unique mnemonic ID
name	`utf8vstring(128)`	?	unique mnemonic name
subname	`utf8vstring(32)`		mnemonic sub-name
desc	`utf8text`		plain text mnemonic description
unit	`utf8vstring(32)`		measurement unit (for example, `"V"`, `"mA"`)
state	`struct_mn_state`	?	current state of mnemonic

Name	Type	Req	Description
pipes	`jsonobject`	?	map of model(s) to associated pipe(s)
full	`asciivstring(32)`		the primary database for the mnemonic, default `f8`
bin	`set(asciistring)`		the opt-in bin database(s) to include the mnemonic in
format	`asciivstring(32)`		printf-style format to render values
enums	`jsonobject`		mapping of permitted text values to numeric values
labels	`list(jsonobject)`		mapping of numeric values or ranges to labels
aliases	`set(utf8string)`		set of additional names associated with the mnemonic
meta	`jsonobject`		additional metadata as needed
query	`asciivstring(32)`		query name for pseudo-mnemonics
conf	`jsonobject`		configuration for pseudo-mnemonics
ext_id	`asciistring`		external ID

Changelog

11.0.0

- added `subname` field
- added `ext_id` field
- rename `origins` field to `pipes`

1.0.0

- `enum` changed to `enums` since "enum" is often a reserved keyword
- `meas` field removed (measure now assumed from `unit`)

Mnemonic Tracking

Used for tracking mnemonic selection activity. Although this is not strictly a definitions database, it is tightly coupled to the mnemonic definitions database, and is thus defined in the definitions context.

Struct Parameters

Parameter	Value
type	`"def_mn_track"`

Database Parameters

Parameter	Value
name	`"mn_track"`

Parameter	Value
label	`"Mnemonic Tracking"`
format	`"{t} {mn_id} {user}"`
order	(`name` , asc)
singular	`"mnemonic definition"`

Fields

Name	Type	Req	Description
t	`instant(us)`	?	time of selection
mn_id	`int(4)`	?	mnemonic ID selected
user	`user_id`	?	user taking action
mns	`set(int(4))`		other mnemonic ID(s) selected
models	`set(asciistring)`		model(s) in current context

Changelog

11.0.0 (planned)

- rename `mn_ids` to `mns`
- order changed to (`t` , desc)
- format changed to `"{t} {mn_id} {user}"`

Nominal Definitions

Holds mnemonic nominal range definitions.

Struct Parameters

Parameter	Value
type	`"def_nominal"`

Database Parameters

Parameter	Value
name	`"nominal"`
label	`"Nominal"`
format	`"{mn_id} {color} ({min}, {max}) {label}"`
order	(`mn_id` , asc), (`label` , asc)
singular	`"nominal definition"`

Fields

Name	Type	Req	Description

unid	`uuid`	?	unique nominal range ID
mn_id	`int(4)`	?	unique mnemonic ID
label	`utf8vstring(128)`	?	nominal range label
desc	`utf8text`		plain text nominal range description
color	`struct_nominal_color`		range color indicator
min	`float(8)`		min value for the range
max	`float(8)`		max value for the range
models	`set(asciistring)`		models for which this range should apply (all if `null`)
meta	`jsonobject`		additional metadata as needed

Notes

The `struct_nominal_color` type is an enum of **green** (0), **yellow** (1), and **red** (2).

Changelog

11.2.0

- added "meta" field

11.0.0

- renamed "nominal_id" to "unid"

Plot Configuration Definitions

Holds mnemonic plot configuration definitions.

Struct Parameters

Parameter	Value
type	`"def_plot"`

Database Parameters

Parameter	Value
name	`"plot"`
label	`"Plot Conf"`
format	`"{name}"`
order	(`name`, asc)
singular	`"plot configuration"`

Fields

Name	Type	Req	Description
name	utf8vstring(128)	?	unique conf name
desc	utf8text		plain text conf description
plot_conf	struct_plot_conf	?	configuration
models	set(asciistring)		models for which this conf should apply (any if null)

Changelog

11.0.0

- renamed field conf to plot_conf (to match profile definition)
- changed type of plot_conf from jsonobject to struct_plot_conf

Profile Definitions

Holds mnemonic profile definitions.

Struct Parameters

Parameter	Value
type	"def_profile"

Database Parameters

Parameter	Value
name	"profile"
label	"Profile"
format	"{name}"
order	(name, asc)
singular	"profile"

Fields

Name	Type	Req	Description
name	utf8vstring(128)	?	unique profile name
desc	utf8text		plain text profile description
models	set(asciistring)		models for which this conf should apply (all if null)
data_conf	struct_data_conf	?	profile data configuration
plot_conf	struct_plot_conf		profile plot configuration. If not provided, defaults to 1 mnemonic per plot, 1 plot per page.
auto_confs	list(struct_auto_conf)		automation configuration

`struct_data_conf` jsonobject

Name	Type	Req	Description
ids	`utf8text`		CSV range list of `mn_id`s e.g. `"1,2-10,100"` or `ext_id` formatted string e.g. `"@[1,4-8,12,100-200]sci;@[2,3]raw"`
filter	`jsonarray` of filter `jsonobject`s		List of filters to apply. Each filter object may reference an existing filter by name using the `filter` key, or directly provide a filter definition (the `type` and `models` fields are ignored). An `ids` key can be used to define which mnemonics the filter should apply to. If `ids` is not provided, then the filter will be applied to every mnemonic. Only one filter per mnemonic is currently supported.
limit	`boolean`		If `True`, generate the Limit Report. Defaults to `False`.
pkt	`boolean`		If `True`, export using Packet Time instead of Ground Receipt Time. Defaults to `False`. Ignored if the archives only contain 1 time source.
join	`boolean`		If `True`, the data file will be formatted with 1 unique time per row, and 1 mnemonic per column. Defaults to `False`.
fill	`boolean`		If `True` and `join` is `True`, empty cells will be populated with the most recent value. Defaults to `False`.
dis	`boolean`		If `True`, each mnemonic will also be exported using the Discrete conversion, if available. Defaults to `False`.
columns	`jsonobject`		Defines which columns are included in the data file.

`columns` jsonobject

Name	Type	Req	Description
date_utc	`boolean`		
ts_utc_iso	`boolean`		
ts_utc_excel	`boolean`		
ts_utc_excel_ms	`boolean`		
ts_utc_doy	`boolean`		
t_utc_unix_s	`boolean`		
t_utc_unix_ms	`boolean`		

Name	Type	Req	Description
t_utc_unix_us	`boolean`		
date_tai	`boolean`		
ts_tai_iso	`boolean`		
ts_tai_doy	`boolean`		
t_tai_unix_s	`boolean`		
t_tai_unix_ms	`boolean`		
t_tai_unix_us	`boolean`		
t_tai_tai_s	`boolean`		
t_tai_tai_ms	`boolean`		
t_tai_tai_us	`boolean`		
t_rel_s	`boolean`		
t_rel_ms	`boolean`		
t_rel_us	`boolean`		
name	`boolean`		
unit	`boolean`		

`plot_conf` jsonobject

See Export Plot Format

`struct_auto_conf` jsonobject

Note: The `auto_confs` field is a list of these described JSON objects.

Name	Type	Req	Description
daily	`boolean`		If `True`, the Profile will be exported once per day. Defaults to `False`.
mine	`boolean`		If `True`, the Profile will be exported during the Mining Task when any of the defined `intervals` are processed.
users	`set(utf8vstring(128))`		The list of NASA AUIDs to notify via email when the Daily Profile Export is generated

Notes

Requires review before use.

Changelog

11.0.0

- added `auto_conf`
- renamed field `data` to `data_conf`
- renamed field `plot` to `plot_conf`
- changed type of `plot_conf` from `jsonobject` to `struct_plot_conf`
- changed type of `data_conf` from `jsonobject` to `struct_data_conf`

Trend Definitions

Holds mnemonic trend definitions.

Struct Parameters

Parameter	Value
type	`"def_trend"`

Database Parameters

Parameter	Value
name	`"trend"`
format	`"Trend"`
order	(`name` , asc)
singular	`"trend definition"`

Fields

Name	Type	Req	Description
name	`utf8vstring(128)`	?	unique trend name
desc	`utf8text`		plain text trend description
profiles	`set(utf8string)`	?	profile name(s) to include in trend
models	`set(asciistring)`		models for which this trend should apply (any if `null`)
trend_conf	`struct_trend_conf`		trend configuration
plot_conf	`jsonobject`		Mapping of profile name to `struct_plot_conf` objects, alllowing you to override a Profile's plot conf.
auto_conf	`struct_auto_conf`		automated generation configuration

`struct_trend_conf` jsonobject

Name	Type	Req	Description
bin_size	`int(4)`		The bin size in minutes to use when trending time ranges

Name	Type	Req	Description
bin_count	`int(4)`		Multiplier of the `bin_size` to use when trending time ranges. The actual trended bin size in minutes is `bin_size` * `bin_count`.
t	array of time range JSON objects		List of JSON objects describing time ranges to trend e.g. `{"start": "2021-06-30T00:00:00Z", "end": "2021-07-21T00:00:00Z" }`
intervals	array of interval JSON objects		List of JSON objects describing Event Intervals to trend
disable_filter	`boolean`		If `True`, do not use any filtered data. Defaults to `False`.

`plot_conf` jsonobject

Allows the Trend Definition to override a Profile's plot configuration. In the below example, `FLT_CRIT_TEMPS` is a profile name. The plot configuration is the same format as the Profile's plot configuration.

```
{
 "FLT_CRIT_TEMPS": {
  "pages": [
   {
    "plots": [
     {
      "title": "Chamber Pressure",
      "mnemonics": [
       "oci.gse.ocl_pc.Pressure.hk.Chamber"
      ]
     }
    ]
   },
  ...
```

`struct_auto_conf` jsonobject

Name	Type	Req	Description
daily	`boolean`		If `True`, the Trend will be generated once per day. Defaults to `False`.
mine	`boolean`		If `True`, the Trend will be generated during the Mining Task when either the Time Range or Interval condition is satisfied.

Name	Type	Req	Description
users	`set(utf8vstring(128))`		The list of NASA AUIDs to notify via email when the Daily Trend is generated

Changelog

11.2.0

- added `models` field
- added `trend_conf` field
- added `auto_conf` field
- removed `intervals` field

11.0.0

- initial release

Files

Archive

Contains all archive files for a pipe. Must be a child of a pipe group.

Struct Parameters

Parameter	Value
type	`"file_archive"`

Database Parameters

Parameter	Value
name	`"archive"`
label	`"Archive"`
format	`"{t_start} {t_end}"`
singular	`"archive file"`

Fields

Name	Type	Req	Description
a_id	`int(4)`	?	archive ID
ufid	`uuid`	?	file UUID
t_start	`instant(us)`	?	start time of the time range that the file covers
t_end	`instant(us)`	?	end time of the time range that the file covers, not inclusive
dur	`duration(us)`	?	**virtual** `t_end` - `t_start`

Name	Type	Req	Description
t_min	`instant(us)`	?	time of first data in file
t_max	`instant(us)`	?	time of last data in file
file_name	`utf8filename`	?	archive file name
meta	`jsonobject`		additional metadata as needed
format	`asciivstring(32)`		file format: `xbin` or `xpf`, where `xpf` is a zipped dir (default `xbin`)
conf	`jsonobject`		configuration for format as needed

Changelog

11.0.0

- changed type from `"mn_file_archive"` to `"file_archive"`
- renamed `uuid` to `ufid`

Buffer

Contains all buffer files for a pipe. Must be a child of a pipe group.

Struct Parameters

Parameter	Value
type	`"file_buffer"`

Database Parameters

Parameter	Value
name	`"buffer"`
label	`"Buffer"`
format	`"{file_name}"`
singular	`"buffer file"`

Fields

Name	Type	Req	Description
ufid	`uuid`	?	file UUID
file_name	`utf8filename`	?	buffer file name
t_min	`instant(us)`	?	time of first data in file
t_max	`instant(us)`	?	time of last data in file
dur	`duration(us)`	?	**virtual** `t_max` - `t_min`
state	`struct_buffer_state`	?	buffer file state

Name	Type	Req	Description
flag	`struct_buffer_flag`		buffer file flag
meta	`jsonobject`		additional metadata as needed
format	`asciivstring(32)`		buffer file format (default `"csv"`)
conf	`jsonobject`		configuration for format as needed

Notes

The state field may be one of four values:

- `PENDING` - the file data is present in the mnemonic buffer database but has not been processed further
- `ARCHIVED` - the file contents have been distributed to the appropriate archive file(s)
- `DEPRECATED` - the file is preserved but no longer included in archive files

The flag field may be one of two values:

- `DEPRECATE` - the file is queued for deprecation
- `RESTORE` - the file is queued for restoration
- `DELETE` - the file is queued for deletion

Changelog

11.0.0

- changed type from `"mn_file_buffer"` to `"file_buffer"`
- renamed `uuid` to `ufid`
- removed `PROCESSED` state

CFT

Current filter table for a pipe. Stores a file for each archive containing the state of each filter at the end of the archive. Only created in pipes where `discrete` is `false`.

Struct Parameters

Parameter	Value
type	`"file_cft"`

Database Parameters

Parameter	Value
name	`"cft"`
label	`"CFT"`
format	`"{file_name}"`
singular	`"CFT file"`

Fields

Name	Type	Req	Description
a_id	`int(4)`	?	archive ID
ufid	`uuid`	?	file UUID
file_name	`utf8filename`	?	CVT file name
format	`asciivstring(32)`		CVT file format (default `"csv"`)
conf	`jsonobject`		configuration for format as needed

Changelog

11.0.0

- initial release

CVT

Current value table for a pipe. Stores a file for each archive containing the values for each mnemonic at the end of the archive. Only created in pipes where `discrete` is `false`.

Struct Parameters

Parameter	Value
type	`"file_cvt"`

Database Parameters

Parameter	Value
name	`"cvt"`
label	`"CVT"`
format	`"{file_name}"`
singular	`"CVT file"`

Fields

Name	Type	Req	Description
a_id	`int(4)`	?	archive ID
ufid	`uuid`	?	file UUID
file_name	`utf8filename`	?	CVT file name
meta	`jsonobject`		additional metadata as needed
format	`asciivstring(32)`		CVT file format (default `"csv"`)
conf	`jsonobject`		configuration for format as needed

Changelog

11.0.0

- initial release

Package

Stores generated export packages. Child of a model group.

Struct Parameters

Parameter	Value
type	`"file_package"`

Database Parameters

Parameter	Value
name	`"package"`
label	`"Package"`
format	`"{file_name}"`
singular	`"package file"`

Fields

Name	Type	Req	Description
t_start	`instant(us)`	?	start time
t_end	`instant(us)`	?	end time
a_id	`int(4)`		archive ID (if generated automatically)
ueid	`UUID`		UEID, if time range from event
label	`utf8vstring(128)`	?	package label
file_name	`utf8filename`	?	package file name
data_conf	`struct_data_conf`	?	data configuration
plot_conf	`struct_plot_conf`		plot configuration
auto_conf	`struct_auto_conf`		automation configuration
profile	`utf8vstring(128)`		profile name, if profile used
profile_version	`int(4)`		profile version, if profile used
meta	`jsonobject`		additional metadata as needed

Changelog

11.0.0

- initial release

Trend Package

Stores generated Trend Export packages. Child of a model group.

Struct Parameters

Parameter	Value
type	`"file_trend"`

Database Parameters

Parameter	Value
name	`"trend"`
label	`"Trend"`
format	`"{file_name}"`
singular	`"trend file"`

Fields

Name	Type	Req	Description
name	`utf8vstring(128)`	?	Name of Trend Definition used to generate the Trend Package
desc	`utf8text`	?	Description of Trend Definition used to generate the Trend Package
profiles	`set(utf8string)`	?	Profiles included in the Trend Package
trend_conf	`struct_trend_conf`	?	Trend Definition's trend conf used to generate the Trend Package
plot_conf	`struct_plot_conf`	?	Trend Definition's plot conf used to generate the Trend Package
file_name	`utf8filename`	?	The filename of the Trend Package zip file
meta	`jsonobject`	?	Metadata
t_min	`instant(us)`	?	Time of the earliest bin point used in the trend
t_max	`instant(us)`	?	Time of the latest bin point used in the trend
generated_at	`instant(us)`	?	Datetime when the Trend Package was generated
generated_by	`user_id`	?	ID of user that generated the Trend Package

Changelog

Events

Event databases come in three forms, simple events, single file per event, and multiple files per event.

Event

Each record is a single event. May be a child of either a model or pipe group.

Struct Parameters

Parameter	Value
type	`"event"`

Database Parameters

Parameter	Value
name	* (default `"event"`)
label	* (default `"Event"`)
format	* (default `"{t_start} {event_id} {label}"`)
order	* (default (`t_start`, desc), (`event_id`, asc))
singular	* (default `"event"`)

Fields

Name	Type	Req	Virtual	Description
ueid	`uuid`	?		event UUID
e_id	`int(4)`	?		event ID (default to `0` if not provided)
a_id	`int(4)`	?		archive ID (only present if child of a pipe group)
t_start	`instant(us)`	?		start time
t_end	`instant(us)`			end time, not inclusive (if `null`, event is an open interval)
dur	`duration(us)`	?	?	duration in microseconds (`null` if open)
interval	`boolean`	?	?	`t_start` != `t_end`
open	`boolean`	?	?	`t_end` is `null`
type	struct_event_type	?		event type (default to `message` if not provided)
level	`struct_event_level`	?		event level (default to `none` if not provided)
label	`utf8vstring(128)`	?		plain text label

Name	Type	Req	Virtual	Description
content	`utf8text`			extended event content
meta	`jsonobject`			additional metadata as needed
conf	`jsonobject`			configuration for specific event types

Notes

Virtual fields are calculated from other fields and cannot be populated manually.

Changelog

11.0.0

- changed `uuid` to `ueid`
- changed `e_id` type from `int(8)` to `int(4)`
- removed `name` field
- added `a_id` field (when child of pipe group)

1.0.2

- corrected `t_end` and `dur` as not required

1.0.1

- corrected `name` as not required

1.0.0

- `pid` (primary ID) changed to `e_id` (event ID) to avoid confusion
- `sid` removed (additional IDs may be added as needed)
- `int` changed to `interval` (`int` is commonly reserved keyword)
- `dur`, `interval`, and `open` are now derived fields from `t_start` and `t_end`
- added `struct_event_type` and `struct_event_level` data types
- added `name` as event definition association

Event File

Uses same structure as event database, with one additional field.

Database Parameters

Parameter	Value
name	* (default `"eventf"`)
label	* (default `"Event File"`)
singular	* (default `"event file"`)

Name	Type	Req	Description
file_name	`utf8filename`	?	file name

Event Files

Uses same structure as event database, but with a child file database, allowing each event to contain zero or more files.

Database Parameters

Parameter	Value
name	* (default `"eventfs"`)
label	* (default `"Event Files"`)
singular	* (default `"event files"`)

Event Update

Captures updates to events as records.

Struct Parameters

Parameter	Value
type	`"event_update"`

Database Parameters

Parameter	Value
name	`"{name}_update"`)
label	`"{label} Update"`
format	`"{t_start} {ueid} {label}"`)
order	(`t`, desc)
singular	`"event change"`

Fields

Name	Type	Req	Description
t	`instant(us)`	?	event change time
ueid	`uuid`	?	event UUID
update	`jsonobject`	?	field(s) to update

Changelog

11.0.0

- initial release

Mnemonics

Databases containing mnemonic data. Unless otherwise indicated, mnemonic databases can be configured with a partitioning system which subdivides the tables internally into UTC calendar months. This is beneficial for selective mnemonic data repopulation, since each month can be instantly erased before being regenerated, and doesn't require downtime of the entire dataset. At creation time a start and end year must be specified, and partitions will be created for each month in that range. Data with timestamps outside the range will be stored in either a pre-range or post-range partition as applicable.

Mn Buffer

Each record is a single mnemonic data point. Holds data imported through the buffer pipeline. Unlike other mnemonic databases, may contain duplicate data points.

This database does not hold data indefinitely. The automated archive pipeline will remove data as it is archived and mined into the primary mnemonic databases.

Struct Parameters

Parameter	Value
type	`"mn_buffer"`

Database Parameters

Parameter	Value
name	`"buffer"`
label	`"Buffer"`
format	`"{t} {mn_id} {v}"`
singular	`"mnemonic buffer datapoint"`

Fields

Name	Type	Req	Description
t	`instant(us)`	?	time
mn_id	`int(4)`	?	unique mnemonic ID
v	`{conf.type}`		value

Mn Full

Each record is a single mnemonic data point. The data type for mnemonic values is configurable, and determines the database name. By default the mnemonic data group will contain a full `float(8)` database.

Struct Parameters

Parameter	Value
type	`"mn_full"`

Conf Parameters

Parameter	Value	Default

type	`"int(1)"`, `"int(2)"`, `"int(4)"`, `"int(8)"`, `"float(4)"`, or `"float(8)"`	`"float(8)"`

Database Parameters

Parameter	Value
name	`"i1"`, `"i2"`, `"i4"`, `"i8"`, `"f4"`, or `"f8"`
label	`"Full <conf.type>"`
format	`"{t} {mn_id} {v}"`
singular	`"mnemonic datapoint"`

Fields

Name	Type	Req	Description
a_id	`int(4)`	?	archive ID
t	`instant(us)`	?	time
mn_id	`int(4)`	?	unique mnemonic ID
v	`{conf.type}`		value

Changelog

11.0.0

- added `a_id`

Mn Delta

An optimized mnemonic storage solution with each record representing one or more mnemonic data points, by only including points where the mnemonic value actually changes. The value data type is customizable, as with the Mn Full database. By default the mnemonic data group will contain a delta `float(8)` database.

Struct Parameters

Parameter	Value
type	`"mn_delta"`

Conf Parameters

Parameter	Value	Default
type	`"int(1)"`, `"int(2)"`, `"int(4)"`, `"int(8)"`, `"float(4)"`, or `"float(8)"`	`"float(8)"`

Database Parameters

Parameter	Value
name	`"di1"`, `"di2"`, `"di4"`, `"di8"`, `"df4"`, or `"df8"`

Parameter	Value
label	`"Delta {conf.type}"`
format	`"{t} {mn_id} {v} ({n})"`
singular	`"mnemonic delta datapoint"`

Fields

Name	Type	Req	Description
a_id	`int(4)`	?	archive ID
t	`instant(us)`	?	time
mn_id	`int(4)`	?	unique mnemonic ID
v	`{conf.type}`		value
n	`int(4)`	?	number of datapoints included in this point

Changelog

11.0.0

- added `a_id`

Mn Bin Time

Contains mnemonic data binned on fixed time intervals. By default these will be created for 1 minute (`"t60"`) and 10 minute (`"t600"`) bin sizes.

Struct Parameters

Parameter	Value
type	`"mn_bin_time"`

Conf Parameters

Parameter	Value
t	bin size in seconds

Database Parameters

Parameter	Value
name	`"t<conf.t>"`
label	`"Time (<conf.t>s)"`
format	`"{t} {mn_id} {avg} ({min}, {max})"`
singular	`"mnemonic bin"`

Fields

Name	Type	Req	Description
a_id	int(4)	?	archive ID
t	instant(us)	?	start time of the bin
mn_id	int(4)	?	unique mnemonic ID
t_min	instant(us)	?	time of first datapoint
t_max	instant(us)	?	time of last datapoint
n	int(4)	?	number of datapoints in bin
avg	float(8)	?	average
min	float(8)	?	min
max	float(8)	?	max
std	float(8)	?	sample standard deviation

Changelog

11.0.0

- added `a_id` field
- removed `med` and `var` fields

Mn Bin Interval

Contains mnemonic data binned by interval events.

Struct Parameters

Parameter	Value
type	`"mn_bin_interval"`

Database Parameters

Parameter	Value
name	`"interval"`
label	`"Interval"`
format	`"{t} {e_id} {mn_id} {avg} ({min}, {max})"`
singular	`"mnemonic bin"`

Fields

Name	Type	Req	Description
a_id	int(4)	?	archive ID
ueid	uuid	?	
e_id	int(8)	?	
t_start	instant(us)	?	start time

Name	Type	Req	Description
t_end	instant(us)	?	end time
mn_id	int(4)	?	unique mnemonic ID
t_min	instant(us)	?	time of first datapoint
t_max	instant(us)	?	time of last datapoint
n	int(4)	?	number of datapoints in bin
avg	float(8)	?	average
min	float(8)	?	min
max	float(8)	?	max
std	float(8)		sample standard deviation

Changelog

11.0.0

- added `a_id` field
- renamed field `uuid` to `ueid`
- removed `med` and `var` fields

Mn Bin Edge

Contains mnemonic data binned on archive boundaries and cross-archive interval event boundaries. Used to generate interval bins efficiently for cross-archive interval events. Only created in pipes where `discrete` is `false`.

Struct Parameters

Parameter	Value
type	`"mn_bin_edge"`

Database Parameters

Parameter	Value
name	`"edge"`
label	`"Edge"`
format	`"{t_start} - {t_end} {mn_id} {avg} ({min}, {max})"`
singular	`"mnemonic bin"`

Fields

Name	Type	Req	Description
a_id	int(4)	?	archive ID
t_start	instant(us)	?	start time
t_end	instant(us)	?	end time

Name	Type	Req	Description
mn_id	`int(4)`	?	unique mnemonic ID
t_min	`instant(us)`	?	time of first datapoint
t_max	`instant(us)`	?	time of last datapoint
n	`int(4)`	?	number of datapoints in bin
avg	`float(8)`	?	average
min	`float(8)`	?	min
max	`float(8)`	?	max
std	`float(8)`	?	sample standard deviation

Changelog

11.0.0

- initial release

Tasks

Store logs and associated files for data processing tasks.

Condense

Each record logs a single execution of a condense task.

Struct Parameters

Parameter	Value
type	task_condense

Database Parameters

Parameter	Value
name	`"condense"`
label	`"Condense"`
format	`"{task_id} {t}"`
singular	`"condense task"`

Fields

Name	Type	Req	Description
task_id	`task_id`	?	unique task ID
t	`instant(us)`	?	time when task submitted
meta	`jsonobject`		additional metadata as needed

Name	Type	Req	Description
conf	`jsonobject`		task configuration
condensed	`list(jsonobject)`		buffer file(s) condensed

Mine

Each record logs a single execution of a mine task.

Struct Parameters

Parameter	Value
type	task_mine

Database Parameters

Parameter	Value
name	`"mine"`
label	`"Mine"`
format	`"{task_id} {t_start}"`
singular	`"mine task"`

Fields

Name	Type	Req	Description
task_id	`task_id`	?	unique task ID
t	`instant(us)`	?	time when task submitted
ufid	`uuid`	?	source archive file UUID
t_start	`instant(us)`	?	source archive file start time
t_end	`instant(us)`	?	source archive file end time
meta	`jsonobject`		additional metadata as needed
conf	`jsonobject`		task configuration

Changelog

11.0.0

- renamed `uuid` to `ufid`

Archive

Each record logs a single execution of an archive task.

Struct Parameters

Parameter	Value

type	task_archive

Database Parameters

Parameter	Value
name	`"archive"`
label	`"Archive"`
format	`"{task_id} {t}"`
singular	`"archive task"`

Fields

Name	Type	Req	Description
task_id	`task_id`	?	unique task ID
t	`instant(us)`	?	time when task submitted
meta	`jsonobject`		additional metadata as needed
conf	`jsonobject`		task configuration
archives	`list(jsonobject)`		archives updated
archived	`list(jsonobject)`		buffer file(s) archived
restored	`list(jsonobject)`		buffer file(s) restored
deprecated	`list(jsonobject)`		buffer file(s) deprecated
deleted	`list(jsonobject)`		buffer file(s) deleted

Spectra

The spectra definition is a property for event databases.

Property	Value	Req	Description
tabs	array of tab conf(s)		custom tabs for UI
presearch	array of presearch confs		custom pre-search components for UI
filters	array of filter confs		
grouping	array of field name(s)		
charts	charts conf	?	
tables	array of table conf		
query	query conf		
labels	labels conf		

Spectra Tab Conf

Configuration for a spectra search tab. This may be a `string`, referencing the name of a custom tab implementation, or an object with a `"type"` property specifying a tab type and additional properties applicable for that type. Currently there are no custom tab types, but they may be added in the future.

Spectra Database Tab

Under Construction

The database tab employs a record search for a separate target database of any type, and a solution for converting a selection from the target database to the spectra database.

Property	Value	Req	Description
type	`"database"`	?	tab type name
database	database specifier	?	target database specifier
map	see below	?	solution to map target selection to spectra selection

The `"map"` property may be a `string`, `array` of `strings`, or `object`.

If a `string`, the value must be the name of a custom selection function (none currently exist, they may be added in the future).

Spectra Presearch Conf

Specifies a set of components to display before the main spectra search component.

Spectra Field Presearch

Specifies a standalone component to search a particular field.

Property	Value	Req	Description
type	`"field"`	?	presearch type name
field	field specifier	?	
options	see below		options for search dropdown

Spectra Filters Conf

Specifies filters / badges for spectra search.

Property	Value	Req	Description
name	`string`	?	system name for filter
label	`string`		display label (uses name if absent)
badge	`string`		badge label (uses name if absent)
desc	`string`		description for badge / filter tooltip
color	`string`		color code or CSS class

Property	Value	Req	Description
e	`expression`	?	expression to apply for filter

Spectra Charts Conf

Specifies options for each spectra chart.

Property	Value	Req	Description
summary	spectra chart conf	?	summary chart conf
spectra	spectra chart conf	?	spectra chart conf

Spectra Chart Conf

Specifies options for a single spectra chart.

Property	Value	Req	Description
x	`string[]`	?	x axis options
y	`string[]`	?	y axis options
tooltip	`string`		record format string

Spectra Tables Conf

Under Construction

Spectra Query Conf

Under Construction

Spectra Labels Conf

Labels are specified as an `object` mapping standard label values to custom values. These will be defined as needed.

Revision #136
Created 13 July 2023 17:27:53 by Nick Dobson
Updated 25 March 2025 19:14:45 by Bradley Tse