TANGO Device Server Design Guidelines

HDB++

Design and implementation

L.Pivetta, G.Scalamera
ELETTRA - Sincrotrone Trieste

Summary

The HDB++ is a novel TANGO device server for Historical Data Base (HDB) archiving. It's written in C++ and is fully event-driven.

Keywords

TANGO Device Server, Historical Data Base, HDB, Archiving, C++

Notes

No notes so far.

Contributions

J.M. Chaize, A.Gotz, J.Meyer, F.Poncet, E.Taurel, P.Verdier - ESRF
M.Lonza, C.Scafuri, G.Scalamera, G.Strangolino - ELETTRA

Revisions

Date	Rev.	Author	
2012-12-04	1.0	L.Pivetta	First release
2013-01-29	1.1	L.Pivetta	Merged suggestions from ESRF
2013-01-31	1.2	L.Pivetta	Cleanup
2013-05-10	1.3	L.Pivetta	Revision after HDB++ meeting on 14.03.2013
2014.01.30	1.4	L.Pivetta	Configuration Manager details + Extraction library
2014.03.07	1.5	L.Pivetta	Database interface

Contents

List of Tables

1 HDB++ TANGO Device Server

The HDB++ architecture is composed by several TANGO device servers. More in detail, at least one, but actually many, Event Subscriber TANGO device server jointly with one Configuration Manager TANGO device server and one or more Data Extraction TANGO device servers are foreseen.

2 Event Subscriber

The Event Subscriber TANGO device server, also called archiver device server, will subscribe to archive events on request by the Configuration Manager. The Event Subscriber will be able to start archiving all the already configured events even if the Configuration Manager is not running. The Event Subscriber device server must have the following characteristics:

1. the archiving mechanism is event-based, thus the device server tries to subscribe to the event; an error means a fault. A transparent re-subscription to the faulty event is required.

2. one additional thread is in charge of events subscription and call-back execution; the call back, acting as producer, must put the complete data of the received events in a FIFO queue; the thread and the callback must be able to handle an *arbitrary* number of events, possibly limited just by the available memory and/or the required performances; also, a high-mark threshold must be setup on the FIFO in order to alert for an overloaded Event Subscriber

3. one additional thread, acting as consumer of the FIFO, is in charge of pushing the data into the database, preserving the event data time stamp too; the code to access the database engine shall be structured to allow the use of different back-ends (MySQL, Oracle, etc...)

4. the device server methods, commands and attributes, must allow to perform the following per-instance operations:

    - start the archiving for all attributes
    - stop the archiving for all attributes
    - start the archiving for one attribute
    - stop the archiving for one attribute
    - read the number of attributes in charge
    - read the list of attributes in charge
    - read the configuration parameters of each attribute
    - read the number of working attributes
    - read the list of working attributes
    - read the number of faulty attributes
    - read the list of faulty attributes with diagnostics
    - read the size of the FIFO queue
    - read the number of attributes pending in the FIFO
    - read the list of attributes pending in the FIFO

The Event Subscriber device server must be able to run and report on the working/faulty attributes/events by means of the standard API (commands and/or attributes) without the need of a graphical interface.

The diagnostics of faults could also be stored in the general info about each attribute; the diagnostics are used by the HDB++ Device Server itself to detect that some data is not being stored as requested.

Stopping the archiving of an attribute does not persist after a restart, i.e. restarting an Event Subscriber device server instance triggers the archiving of *all* configured attributes. A property can be setup not to start archiving at Event Subscriber startup (see 2.1.3 and 2.1.4).

One NULL value with time stamp is inserted whenever the archiving of an attribute is stopped, due to error or by a specific stop command. Moreover, if an error occurred the corresponding attribute is marked as faulty in the archiving engine. In case the archiving was suspended due to error, it is automatically resumed when good data is available again. One or more alarms could be configured in the TANGO Alarm System to asynchronously inform about the status of the archiving device server.

The Event Subscriber TANGO device server shall also expose some additional figures of merit such as:

- for each instance, total number of records per time
- for each instance, total number of failures per time
- for each attribute, number of records per time
- for each attribute, number of failures per time
- for each attribute, time stamp of last record

The system can sum these numbers in a counter which can be reset every hours/days/weeks to rank each attribute in term of data rate, error rate etc. This allows preventive maintenance and fine tuning, detecting, for instance, when an attribute is too verbose (e.g. variation threshold below the noise level). These statistics are a key element for qualifying the health of the system. All these attributes will be themselves archived to enable a follow-up versus time.

The Event Subscriber TANGO device server must maintain at least the following operating states:

- **ON**: archiving running, everything works
- **ALARM**: one or more attributes faulty or the FIFO size grows above high-mark threshold
- **FAULT**: all attributes faulty
- **OFF**: archiving stopped

2.1 Event Subscriber interface

More in detail the Event Subscriber device server interface is summarized in table 1 and 2.

2.1.1 Commands

AttributeAdd	add an attribute to archiving; the complete FQDN has to be specified otherwise it is completed by the Event Subscriber using getaddrinfo()
AttributeRemove	remove an attribute from archiving; the archived data and the attribute's archive event configuration are left untouched
AttributeStatus	read attribute status
AttributeStart	start archiving specified attribute
AttributeStop	stop archiving specified attribute
Start	start archiving
Stop	stop archiving
ResetStatistics	reset Event Subscriber statistics

Table 1: Event Subscriber Commands.

2.1.2 Attributes

AttributeOkNumber	number of archived attribute not in error
AttributeNokNumber	number of archived attribute in error
AttributePendingNumber	number of attributes waiting to be archived
AttributeNumber	number of attributes configured for archiving
AttributeList	return configured attribute list
AttributeOkList	return the list of attribute not in error
AttributeNokList	return the list of attribute in error
AttributePendingList	list of attributes waiting to be archived

Table 2: Event Subscriber Attributes.

The class and device properties availabile for configuration are shown in table 3 and 4.

2.1.3 Class properties

DbHost	hostname of host running the database engine
DbUser	database user
DbPassword	database password for DbUser
DbName	database name
DbPort	port number
StartArchivingAtStartup	start archiving at Event Subscriber startup

Table 3: Event Subscriber Class properties.

2.1.4 Device properties

SubscribeRetryPeriod	retry period for subscribe event in seconds
AttributeList	list of configured attributes
DbHost	hostname of host running the database engine
DbUser	database user
DbPassword	database password for DbUser
DbName	database name
DbPort	port number
StartArchivingAtStartup	start archiving at Event Subscriber startup

Table 4: Event Subscriber Device properties.

3 Configuration Manager

In order to address large archiving systems the need to distribute the workload over a large number of threads/processes shows up. A Configuration Manager device server will assist in the operations of adding, editing, moving, deleting an attribute to/from the archiving system. All the configuration parameters, such as polling period, variation thresholds etc., are kept in the TANGO database as properties the archived attribute. The list of attributes in charge of each Event Subscriber is stored in the TANGO database as properties of the Configuration Manager device server. The scaling capability comes transparently with the TANGO architecture itself.

The Configuration Manager device server shall be able to perform the following operations:

1. manage the request of archiving a new attribute
   - 1.1 create an entry in the HDB++ if not already done
   - 1.2 setup the attribute's archive event configuration
   - 1.3 assign the new attribute to one of the Event Subscriber device servers
     - following some rules of load balancing
     - to the specified Event Subscriber device server
2. create a new Event Subscriber device server instance
3. move an attribute from an Event Subscriber device server to another one
4. keep trace of which attribute is assigned to which Event Subscriber
5. start/stop the archiving of an attribute at runtime
6. remove an attribute from archiving

The load balancing capability of the Configuration Manager, if desired, must be enabled by means of a device property both in the Configuration Manager and the Event Subscriber device servers pool reserved for automatic load balancing. This enables hybrid reserved/balanced archiving engine configuration.

The configuration shall be possible via the Configuration Manager device server API as well as via a dedicated GUI interface; the GUI may just use the provided API.

The Configuration Manager may also expose a certain number of attributes to give the status of what is going on:

- total number of Event Subscriber
- total number of working attributes
- total number of faulty attributes
- total number of calls per second

These attributes could be themselves archived to enable a follow up versus time.

3.1 Configuration Manager interface

More in detail the Configuration Manager device server exposes the following interface.

3.1.1 Commands

The commands availabile in the Configuration Manager are summarized in table 5.

AttributeAdd	add an attribute to archiving
AttributeRemove	remove an attribute from archiving; the archived data and the attribute's archive event configuration are left untouched
AttributeStart	start archiving an attribute
AttributeStop	stop archiving an attribute
AttributeSetArchiver	assign attribute to Event Subscriber
AttributeGetArchiver	return Event Subscriber in charge of attribute
AttributeStatus	read attribute archiving status
AttributeSearch	return list of attributes containing input pattern
ArchiverAdd	add a new Event Subscriber instance to the archivers list; the instance must have been already created and configured via jive/astor and the device shall be running; as per HDB++ Configuration Manager release CM adding an Event Subscriber device to an existing istance is not supported
ArchiverRemove	remove an Event Subscriber instance from the Configuration Manager list; neither the TANGO device instance nor the attributes configured are removed from the TANGO database
ResetStatistics	reset statistics of Configuration Manager and all Event Subscribers

Table 5: Configuration Manager Commands.

Note that the list of managed Event Subscribers is stored into the ArchiverList device property (see 3.1.3) that maintained via the AttributeAdd, AttributeRemove and AttributeSetArchiver commands. Therefore in the HDB++ archiving system the Event Subscriber device server instances can also be configured by hand, if required, an run independently.

3.1.2 Attributes

The attributes of the Configuration Manager are summarized in table 6.

AttributeOkNumber	number of archived attribute not in error
AttributeNokNumber	number of archived attribute in error
AttributePendingNumber	number of attributes waiting to be archived
AttributeNumber	number of attributes configured for archiving
ArchiverList	return list of attribute in charge to archiver
SetAttributeName	support attribute for setup
SetPollingPeriod	support attribute for setup
SetAbsoluteEvent	support attribute for setup
SetRelativeEvent	support attribute for setup
SetPeriodEvent	support attribute for setup
SetCodePushedEvent	support attribute for setup
SetArchiver	support attribute for setup

Table 6: Configuration Manager Attributes.

The SetXxxYyy attributes are used for archive event and archiver instance configuration setup and must be filled before calling the AttributeAdd command. The AttributeAdd checks the consistency of the desired event configuration and then adds the new attribute to the archiver instance specified with SetArchiver. Then the AttributeAdd command creates the required entries into the historical database.

3.1.3 Device properties

ArchiverList	list of existing archivers
MaxSearchSize	max size for AttributeSearch result

Table 7: Configuration Manager device properties.

4 Diagnostic tools

With all the statistics kept in the Event Subscriber device servers and the Configuration Manager device server, the diagnostic tool can be straightforward to develop as a simple QTango or ATK GUI. This GUI will also give read access to the configuration data stored as attribute properties in the TANGO database to display the attribute polling frequency of the involved device servers, whenever available, and the archive event configuration.

5 Database interface

A C++ API will be developed to address the writing and reading operations on the database and made availabile as a library. This library will provide the *essential* methods for accessing the database. The Event Subscriber, the Configuration Manager, the Data Extraction device servers, library and tools will eventually take advantage of the library. Actually a number of libraries are already available to encapsulate database access decouple the back-end:

libhdb++	:	HDB++ abstraction layer
libhdb++mysql	:	HDB++ table support, MySQL back-end
libhdbmysql	:	legacy HDB table support, MySQL back-end

Table 8: Available database libraries.

Additional libraries are foreseen to support different database engines, such as Oracle, Postgres or possibly noSQL implementations.

5.1 HDB++ database structure

The structure of the legacy HDB is based on three tables, (*adt*, *amt*, *apt*) shown in appendix A. In addition, one table, named att_xxxxx is created for each attribute or command to be archived. Many of the columns in the lagacy tables are used for HDB archiving engine and archiving parameters configuration and are no more required.

The new database structure, whose tables have been designed for the HDB++ archiver, provides just the necessary columns and takes advantage of μs resolution support for daytime.

The *att_conf* table associates the attribute name with a unique id and selects the data type; it's worth notice that the *att_name* raw always contains the complete FQDN, e.g. with the hostname and the domainname. The *att_history* table stores the timestamps relevant for archiving diagnostics.

In addition a number of different data types, listed in table 9, are natively supported for archiving. As an example the table *att_scalar_int8_rw*, for archiving one byte-size read/write values, is also shown below. Three timestamp rows are currently supported: the attribute event timestamp, the reception timestamp and the database insertion timestamp.

```
mysql> desc att_conf;
+-------------+-------------------------------+------+-----+---------+----------------+
| Field       | Type                          | Null | Key | Default | Extra          |
+-------------+-------------------------------+------+-----+---------+----------------+
| att_conf_id | int(10) unsigned              |  NO  | PRI | NULL    | auto_increment |
| att_name    | varchar(255)                  |  NO  | UNI | NULL    |                |
| data_type   | enum('scalar_double_ro',      |      |     |         |                |
|             |      'scalar_double_rw',      |      |     |         |                |
|             |      'array_double_ro',       |      |     |         |                |
|             |      'array_double_rw',       |      |     |         |                |
|             |      'scalar_int64_ro',       |      |     |         |                |
|             |      'scalar_int64_rw',       |      |     |         |                |
|             |      'array_int64_ro',        |      |     |         |                |
```

```
|             |      'array_int64_rw',        |      |     |         |                |
|             |      'scalar_int8_ro',        |      |     |         |                |
|             |      'scalar_int8_rw',        |      |     |         |                |
|             |      'array_int8_ro',         |      |     |         |                |
|             |      'array_int8_rw',         |      |     |         |                |
|             |      'scalar_string_ro',      |      |     |         |                |
|             |      'scalar_string_rw',      |      |     |         |                |
|             |      'array_string_ro',       |      |     |         |                |
|             |      'array_string_rw')       | NO   |     | NULL    |                |
+-------------+-------------------------------+------+-----+---------+----------------+
```

```
mysql> desc att_history;
+-------------+-------------------------------------+------+-----+---------+-------+
| Field       | Type                                | Null | Key | Default | Extra |
+-------------+-------------------------------------+------+-----+---------+-------+
| att_conf_id | int(10) unsigned                    | NO   | MUL | NULL    |       |
| time        | datetime(6)                         | NO   |     | NULL    |       |
| event       | enum('add','remove','start','stop') | NO   |     | NULL    |       |
+-------------+-------------------------------------+------+-----+---------+-------+
```

```
mysql> desc att_scalar_int8_rw;
+-------------+------------------+------+-----+---------+-------+
| Field       | Type             | Null | Key | Default | Extra |
+-------------+------------------+------+-----+---------+-------+
| att_conf_id | int(10) unsigned | NO   | MUL | NULL    |       |
| event_time  | datetime(6)      | NO   | MUL | NULL    |       |
| recv_time   | datetime(6)      | NO   |     | NULL    |       |
| insert_time | datetime(6)      | NO   |     | NULL    |       |
| value_r     | tinyint(1)       | YES  |     | NULL    |       |
| value_w     | tinyint(1)       | YES  |     | NULL    |       |
+-------------+------------------+------+-----+---------+-------+
```

att_scalar_int8_ro, att_scalar_int8_rw	byte size, e.g. state
att_scalar_int64_ro, att_scalar_int64_rw	short to long int
att_scalar_double_ro, att_scalar_double_rw	float and double
att_scalar_string_ro, att_scalar_string_rw	string
att_array_int8_ro, att_array_int8_rw	byte size, e.g. state
att_array_int64_ro, att_array_int64_rw	short to long int
att_array_double_ro, att_array_double_rw	float and double
att_array_string_ro, att_array_string_rw	string

Table 9: Supported data types for archiving.

The complete SQL source for all the tables is reported in appendix B. The main differences can be summarized as:

- μs timestamp resolution
- no per-attribute additional tables; the number of tables used is fixed and does not depend on the number of archived attributes
- specific data type support

5.2 Performance figures

Some tests have been carried out to compare the performance of the existing database structure to the proposed one. The machine hosting the MySQL database engine is equipped with:

- chipset Intel x58 Express ICH10R
- one Intel(R) Core(TM) i7 CPU 980, 6C/12T, 3.3 GHz 4.8 GT/s, 12MB cache
- 24GB DDR3 1333 MHz
- two 120 GB SSD drive OCZ Vertex III MAX IOPS

5.2.1 Legacy HDB

5.2.2 HDB++

6 Data Extraction

A native tool, available to be run locally, as well as a reworked web interface (E-Giga) are foreseen. A specific library with a dedicated API could be developed to address the extraction and the be used into whatever tool may be provided: a TANGO device server, a web interface, a native graphical panel, etc. The Data Extraction library shall be able to deal with event based archived data. The eventual lack of data inside the requested time window shall be properly managed:

- returning some *no-data-available* error: in this case the reply contains no data and a *no-data-available* error is triggered. Care must be taken whenever the requirement of getting multiple data is foreseen.
- enlarging the time window itself to comprehend some archived data: the requested time interval is enlarged in order to comprehend some archived data. A mechanism shall be provided to notify the client of the modified data set. No fake samples have to be introduced to fill the values in correspondence of the requested timestamps.
- returning the value of the last archived data anyhow: the requested time interval is kept and the last available data sample is returned. The validity of the data is guaranteed when the archiving mechanism is based on archive event on change; care must be taken when using the data in case of periodic event.

Moreover, whenever extracting multiple rows, the Data Extraction library shall allow to select one of the following behaviours:

- return variable length data arrays for each row
- return equal length data arrays for all rows, filling the gaps with the previous data value

The extraction library shall be able to manage a query and data cache locally on the host. This allows to enforce some advantages:

- avoid repeating queries on the historical database
- allow issuing small queries just to supplement the cached data
- speed-up the execution of the client

The extraction library shall guarantee the consistency of the local cache with respect to the query and to the archive data. A configuration parameter can be setup to invalidate the local cache after a predefined period of time (e.g. 1 hour, 1 day...). The behaviour of the extraction library, as well as the maximum size of the local cache and every other parameter, shall be configurable via the following mechanisms, ordered by increasing priority:

- system-wide configuration file
- per-client/per-user configuration file
- environment variables

7 General remarks

Care must be taken to avoid introducing dependencies from libraries not already needed by the TANGO core.

A Legacy HDB tables structure

```
mysql> describe adt;
+-------------+--------------------------------+------+-----+---------+----------------+
| Field       | Type                           | Null | Key | Default | Extra          |
+-------------+--------------------------------+------+-----+---------+----------------+
| ID          | smallint(5) unsigned zerofill | NO   | PRI | NULL    | auto_increment |
| time        | datetime                       | YES  |     | NULL    |                |
| full_name   | varchar(200)                   | NO   | PRI |         |                |
| device      | varchar(150)                   | NO   |     |         |                |
| domain      | varchar(35)                    | NO   |     |         |                |
| family      | varchar(35)                    | NO   |     |         |                |
| member      | varchar(35)                    | NO   |     |         |                |
| att_name    | varchar(50)                    | NO   |     |         |                |
| data_type   | tinyint(1)                     | NO   |     | 0       |                |
| data_format | tinyint(1)                     | NO   |     | 0       |                |
| writable    | tinyint(1)                     | NO   |     | 0       |                |
| max_dim_x   | smallint(6) unsigned           | NO   |     | 0       |                |
| max_dim_y   | smallint(6) unsigned           | NO   |     | 0       |                |
| levelg      | tinyint(1)                     | NO   |     | 0       |                |
| facility    | varchar(45)                    | NO   |     |         |                |
| archivable  | tinyint(1)                     | NO   |     | 0       |                |
| substitute  | smallint(9)                    | NO   |     | 0       |                |
+-------------+--------------------------------+------+-----+---------+----------------+

mysql> describe amt;
+-------------------+--------------------------------+------+-----+---------+-------+
| Field             | Type                           | Null | Key | Default | Extra |
+-------------------+--------------------------------+------+-----+---------+-------+
| ID                | smallint(5) unsigned zerofill | NO   |     | 00000   |       |
| archiver          | varchar(255)                   | NO   |     |         |       |
| start_date        | datetime                       | YES  |     | NULL    |       |
| stop_date         | datetime                       | YES  |     | NULL    |       |
| per_mod           | int(1)                         | NO   |     | 0       |       |
| per_per_mod       | int(5)                         | YES  |     | NULL    |       |
| abs_mod           | int(1)                         | NO   |     | 0       |       |
| per_abs_mod       | int(5)                         | YES  |     | NULL    |       |
| dec_del_abs_mod   | double                         | YES  |     | NULL    |       |
| gro_del_abs_mod   | double                         | YES  |     | NULL    |       |
| rel_mod           | int(1)                         | NO   |     | 0       |       |
| per_rel_mod       | int(5)                         | YES  |     | NULL    |       |
| n_percent_rel_mod | double                         | YES  |     | NULL    |       |
| p_percent_rel_mod | double                         | YES  |     | NULL    |       |
| thr_mod           | int(1)                         | NO   |     | 0       |       |
| per_thr_mod       | int(5)                         | YES  |     | NULL    |       |
| min_val_thr_mod   | double                         | YES  |     | NULL    |       |
| max_val_thr_mod   | double                         | YES  |     | NULL    |       |
| cal_mod           | int(1)                         | NO   |     | 0       |       |
| per_cal_mod       | int(5)                         | YES  |     | NULL    |       |
| val_cal_mod       | int(3)                         | YES  |     | NULL    |       |
| type_cal_mod      | int(2)                         | YES  |     | NULL    |       |
| algo_cal_mod      | varchar(20)                    | YES  |     | NULL    |       |
| dif_mod           | int(1)                         | NO   |     | 0       |       |
| per_dif_mod       | int(5)                         | YES  |     | NULL    |       |
```

```
| ext_mod           | int(1)                        | NO   |     | 0       |       |
| refresh_mode      | tinyint(4)                    | YES  |     | 0       |       |
+-------------------+-------------------------------+------+-----+---------+-------+

mysql> describe apt;
+---------------+--------------------------+------+-----+---------+-------+
| Field         | Type                     | Null | Key | Default | Extra |
+---------------+--------------------------+------+-----+---------+-------+
| ID            | int(5) unsigned zerofill | NO   | PRI | 00000   |       |
| time          | datetime                 | YES  |     | NULL    |       |
| description   | varchar(255)             | NO   |     |         |       |
| label         | varchar(64)              | NO   |     |         |       |
| unit          | varchar(64)              | NO   |     | 1       |       |
| standard_unit | varchar(64)              | NO   |     | 1       |       |
| display_unit  | varchar(64)              | NO   |     |         |       |
| format        | varchar(64)              | NO   |     |         |       |
| min_value     | varchar(64)              | NO   |     | 0       |       |
| max_value     | varchar(64)              | NO   |     | 0       |       |
| min_alarm     | varchar(64)              | NO   |     | 0       |       |
| max_alarm     | varchar(64)              | NO   |     | 0       |       |
+---------------+--------------------------+------+-----+---------+-------+
```

B HDB++ tables SQL

```
CREATE TABLE IF NOT EXISTS att_conf
(
att_conf_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
att_name VARCHAR(255) UNIQUE NOT NULL,
data_type
ENUM('scalar_double_ro','scalar_double_rw','array_double_ro','array_double_rw',
'scalar_int64_ro','scalar_int64_rw','array_int64_ro','array_int64_rw',
'scalar_int8_ro','scalar_int8_rw','array_int8_ro','array_int8_rw',
'scalar_string_ro','scalar_string_rw','array_string_ro','array_string_rw')
NOT NULL,
INDEX(att_name)
) ENGINE=MyISAM COMMENT='Attribute Configuration Table';


CREATE TABLE IF NOT EXISTS att_history
(
att_conf_id INT UNSIGNED NOT NULL,
time DATETIME(6) NOT NULL,
event ENUM('add','remove','start','stop') NOT NULL,
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Attribute Configuration Events History Table';


CREATE TABLE IF NOT EXISTS att_scalar_double_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r DOUBLE DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Double ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_double_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r DOUBLE DEFAULT NULL,
value_w DOUBLE DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Double ReadWrite Values Table';


CREATE TABLE IF NOT EXISTS att_array_double_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
```

```
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r DOUBLE DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Double ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_array_double_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r DOUBLE DEFAULT NULL,
value_w DOUBLE DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Double ReadWrite Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_int64_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r BIGINT DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Int up to 64 bit ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_int64_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r BIGINT DEFAULT NULL,
value_w BIGINT DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Int up to 64 bit ReadWrite Values Table';


CREATE TABLE IF NOT EXISTS att_array_int64_ro
(
```

```
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r BIGINT DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Int up to 64 bit ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_array_int64_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r BIGINT DEFAULT NULL,
value_w BIGINT DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Int up to 64 bit ReadWrite Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_int8_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r TINYINT(1) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Int up to 8 bit ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_int8_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r TINYINT(1) DEFAULT NULL,
value_w TINYINT(1) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar Int up to 8 bit ReadWrite Values Table';
```

```
CREATE TABLE IF NOT EXISTS att_array_int8_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r TINYINT(1) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Int up to 8 bit ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_array_int8_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r TINYINT(1) DEFAULT NULL,
value_w TINYINT(1) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array Int up to 8 bit ReadWrite Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_string_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r VARCHAR(16384) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar String ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_scalar_string_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
value_r VARCHAR(16384) DEFAULT NULL,
value_w VARCHAR(16384) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Scalar String ReadWrite Values Table';
```

```
CREATE TABLE IF NOT EXISTS att_array_string_ro
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r VARCHAR(16384) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array String ReadOnly Values Table';


CREATE TABLE IF NOT EXISTS att_array_string_rw
(
att_conf_id INT UNSIGNED NOT NULL,
event_time DATETIME(6) NOT NULL,
recv_time DATETIME(6) NOT NULL,
insert_time DATETIME(6) NOT NULL,
idx INT UNSIGNED NOT NULL,
dim_x INT UNSIGNED NOT NULL,
dim_y INT UNSIGNED NOT NULL DEFAULT 0,
value_r VARCHAR(16384) DEFAULT NULL,
value_w VARCHAR(16384) DEFAULT NULL,
INDEX(event_time),
INDEX(att_conf_id)
) ENGINE=MyISAM COMMENT='Array String ReadWrite Values Table';
```