AAVS Calibration DB Documentation

# Collection Overview: db.py

The database consists of four collections: Antenna, Fit, Channel and Coefficient. To increase efficiency all time stamps are stored as Unix timestamps.

## Antenna

Apart from the status fields for the pols, the antenna collection only holds static data.

### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Comment** | |
| id | BSON | unique id of the antenna | |
| antenna\_nr | Int | antenna number within the station | |
| station\_id | Int |  | |
| x\_pos | Float | meters from the station centre in x direction | |
| y\_pos | Float | meters from the station centre in y direction | |
| base\_id | Int |  | |
| tpm\_id | Int |  | |
| tpm\_rx | Int | rx number within the tpm | |
| antenna\_type | String | new antenna types can be defined in the antenna class | |
| status\_x | String | current status of the x pol | new states can be defined in the antenna class |
| status\_y | String | current status of the y pol |

## Fit

### Fields

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Comment** |
| id | BSON | unique id of the fit |
| acquisition\_time | Int | Unix timestamp of the acquisition time. See 1.2.2 for more info. |
| antenna\_id | BSON | id of the antenna the fit relates to |
| pol | Int | x = 0, y =1 |
| fit\_time | Int | Unix timestamp of the fit time. See 1.2.2 for more info. |
| fit\_comment | String |  |
| channels | List | List with channels (collection) |
| flags | String |  |
| phase\_0 | Float |  |
| delay | Float |  |
| status\_y | String |  |

### Functions

To save a time stamp in the form of a *datetime* instance use the function *set\_acquisition\_time* or *set\_fit\_time* respectively. The time stamp should be in UTC.

To retrieve a time stamp as a *datetime* instance use the function *get\_acquisition\_time* or *get\_fit\_time* respectively. The returned time stamp is in UTC.

## Channel

### Fields

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Comment** |
| id | BSON | unique id of the channel |
| frequency | Int | Unix timestamp of the acquisition time. See 1.2.2 for more info. |
| amp | Float |  |
| phase | Float | x = 0, y =1 |
| fit\_id | BSON | id of the related fit |
| antenna\_id | BSON | id of the related antenna |

## Coefficient

### Fields

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Comment** |
| id | BSON | unique id of the coefficient |
| antenna\_id | BSON | id of the related antenna |
| pol | Int | x = 0, y =1 |
| calibration | List | List of complex numbers as strings. See 1.4.2 for more info. |
| download\_time | Int | Unix timestamp of the download time. See 1.2.2 for more info. |

### Functions

To save a time stamp in the form of a *datetime* instance use the function *set\_download\_time*. The time stamp should be in UTC.

To retrieve a time stamp as a *datetime* instance use the *get\_download\_time* function. The returned time stamp is in UTC.

To save a list of complex numbers use the function *set\_calibrations* and to retrieve it use the *get\_calibrations* function.

## Functions

*connect\_to\_db* establishes a connection with mongoDB. Default values for the database name, the IP address and the port are defined as global variables.

To save a *datetime* while creating an instance, the function *convert\_datetime\_to\_timestamp* can be used.

# Other Files

## example\_queries.py

Offers some examples on how to access data stored in the database. For more information, visit <http://docs.mongoengine.org/guide/querying.html>.

## populate\_db.py

This script populates the database with random data for test purposes. It also provides examples of how to populate the db.

## purge\_db.py

All dynamic collections (Fit, Channel and Coefficient) are dropped from the database and only Antenna remains.

## refresh\_antenna\_data.py

Refreshes the antenna data from the excel sheet. Matching of the sheet data and the database is done by comparing *antenna\_nr* and *station\_id.*

## setup.py

Installs the required packages for the project when calling *python setup.py install*.

## test.py

Unit tests for the project. Uses a separate test database which is dropped after the tests.

## install.sh

Install script for the project on Ubuntu. It installs mongoDB, the project and the required packages. Needs to be executed with administrator privileges and a Python installation is required.