developer.skatelescope.org Documentation

Release 0.1.0-alpha

Marco Bartolini

May 27, 2020

CONTENTS:

1 Getting started **3**

2 API **5**

2.1 MCCS Master 5
2.2 MCCS Subarray 5
2.3 MCCS Station 5
2.4 MCCS Station Beam 5
2.5 MCCS Tile 5
2.6 MCCS Antenna 5
2.7 control_model 5
2.8 utils 5

3 Indices and tables **7**

Index **9**

This project is developing the Local Monitoring and Control (LMC) prototype for the Square Kilometre Array.

CHAPTER

ONE

GETTING STARTED

1. To set up your environment, follow the instructions on the Tango Development Environment set up page.
2. Set up your itango docker container to mount your host working directory. This will allow you to launch locally hosted code within the itango container. To do this, edit `/usr/src/ska-docker/docker-compose/itango.yml` and add the following lines under the itango service definition:

```
volumes:
  - ${HOME}:/hosthome:rw
```

3. Clone our GitLab repo.
4. Verify your setup:

```
$ cd /usr/src/ska-docker/docker-compose
$ make start itango #not needed if it already shows in "make status"
$ docker exec -it -e PYTHONPATH=/hosthome/ska-logging:/hosthome/lmc-base-classes/src \
  itango python3 \
  /hosthome/lfaa-lmc-prototype/src/ska/mccs/MccsMaster.py -?
usage :  MccsMaster instance_name [-v[trace level]] [-nodb [-dlist <device name list>
↪]]
Instance name defined in database for server MccsMaster :
$ docker exec -it -e PYTHONPATH=/hosthome/ska-logging:/hosthome/lmc-base-classes/src \
  itango tango_admin --add-server MccsMaster/01 MccsMaster lfaa/master/01
$ docker exec -it -e PYTHONPATH=/hosthome/ska-logging:/hosthome/lmc-base-classes/src \
  itango python3 \
  /hosthome/lfaa-lmc-prototype/src/ska/mccs/MccsMaster.py 01
1|2020-03-13T05:27:15.844Z|INFO|MainThread|write_loggingLevel|SKABaseDevice.py
↪#490|tango-device:lfaa/master/01|Logging level set to LoggingLevel.INFO on Python␣
↪and Tango loggers
1|2020-03-13T05:27:15.845Z|INFO|MainThread|update_logging_handlers|SKABaseDevice.py
↪#169|tango-device:lfaa/master/01|Logging targets set to []
1|2020-03-13T05:27:15.846Z|INFO|MainThread|init_device|SKABaseDevice.py#399|tango-
↪device:lfaa/master/01|No Groups loaded for device: lfaa/master/01
1|2020-03-13T05:27:15.846Z|INFO|MainThread|init_device|SKABaseDevice.py#401|tango-
↪device:lfaa/master/01|Completed SKABaseDevice.init_device
Ready to accept request
```

CHAPTER

TWO

API

2.1 MCCS Master

2.2 MCCS Subarray

2.3 MCCS Station

2.4 MCCS Station Beam

2.5 MCCS Tile

2.6 MCCS Antenna

2.7 control_model

2.8 utils

ska.mccs.utils.**call_with_json**(*func*, ***kwargs*)

Allows the calling of a command that accepts a JSON string as input, with the actual unserialised parameters.

Parameters

- **func** – the function to call
- **kwargs** – parameters to be jsonified and passed to func

Ptype func callable

Ptype kwargs any

Returns the return value of func

Example Suppose you need to use MccsMaster.Allocate() to command a master device to allocate certain stations and tiles to a subarray. Allocate() accepts a single JSON string argument. Instead of

parameters={"id": id, "stations": stations, "tiles": tiles} json_string=json.dumps(parameters) master.Allocate(json_string)

save yourself the trouble and

call_with_json(master.Allocate, id=id, stations=stations, tiles=tiles)

class ska.mccs.utils.**json_input**(*schema_path=None*)

Method decorator that parses and validates JSON input into a python object. The wrapped method is thus called with a JSON string, but can be implemented as if it had been passed an object.

If the string cannot be parsed as JSON, an exception is raised.

Parameters **schema_path** – an optional path to a schema against which the JSON should be validated. Not working at the moment, so leave it None.

Ptype string

Example Conceptually, MccsMaster.Allocate() takes as arguments a subarray id, an array of stations, and an array of tiles. In practice, however, these arguments are encoded into a JSON string. Implement the function with its conceptual parameters, then wrap it in this decorator:

@json_input def MccsMaster.Allocate(id, stations, tiles):

The decorator will provide the JSON interface and handle the decoding for you.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

C

call_with_json() (*in module ska.mccs.utils*), 5

J

json_input (*class in ska.mccs.utils*), 6