developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Feb 07, 2023

TABLE OF CONTENTS

CHAPTER

ONE

QUICKSTART

This project is structured to use Docker containers for development and testing so that the build environment, test environment and test results are all completely reproducible and are independent of host environment. It uses `make` to provide a consistent UI (see *Makefile targets*).

Build a new Docker image and execute the test suite with:

```
make oci-build
```

Execute the test suite and lint the project with:

```
make python-test && make python-lint
```

Format the Python code:

```
make python-format
```

CHAPTER

TWO

MAKEFILE TARGETS

This project contains a Makefile Gitlab Submodule which acts as a UI for building Docker images, testing images, and for launching interactive developer environments. The following make targets are defined:

Makefile target	Description
oci-build	Build a new application image
python-test	Test the application image
python-lint	Lint the application image
python-format	Format the Python code
help	show a summary of the makefile targets above

CHAPTER

THREE

BACKGROUND

SKA Tango devices have commands that accept structured arguments and/or return structured responses. These structured data are often expressed as JSON-formatted strings.

The Configuration Data Model (CDM) is a data model used to describe subarray resource allocations and the subsequent configuration of those resources. It is effectively the superset of the configurations used by receptors, correlators, and data processing systems. The CDM is one such example of structured data delivered to TMC Tango devices.

This project defines object representations of the structured data passed to and from Tango devices, and serialisation schema used to convert the structured data to and from JSON. This project defines:

1. a Python object model of the CDM;
2. a Python object model for the structured arguments sent to TMC Tango devices and the structured responses received in return;
3. serialisation schema to convert the Python object model instances to and from JSON.
4. validation of the JSON strings sent between devices are compliant with the agreed interfaces.

The primary users of this shared library are the OET, SubArrayNode, and CentralNode. The OET uses this library to construct object representations of telescope configurations and resource allocation instructions, to convert those object representations to JSON-formatted payloads for TMC devices, and finally to convert the JSON responses returned by TMC devices back into Python objects.

It is intended that TMC devices also use this library to guarantee correct data exchange with the OET. TMC can also use this library to marshall and unmarshall its arguments to CSP and SDP Tango devices, which accept the appropriate subset of the JSON.

CHAPTER

FOUR

PROJECT LAYOUT

The CDM project contains three top-level packages, `ska_tmc_cdm.messages`, `ska_tmc_cdm.schemas` and `ska_tmc_cdm.jsonschema` as shown in the figure below. The `ska_tmc_cdm.messages` package contains Python object models for the JSON command arguments agreed in the ICDs. The `ska_tmc_cdm.schemas` package contains code to transform the classes defined in `ska_tmc_cdm.messages` to and from JSON. The `ska_tmc_cdm.jsonschema` package contains code to verify that the JSON strings sent between devices are compliant with the agreed interfaces.

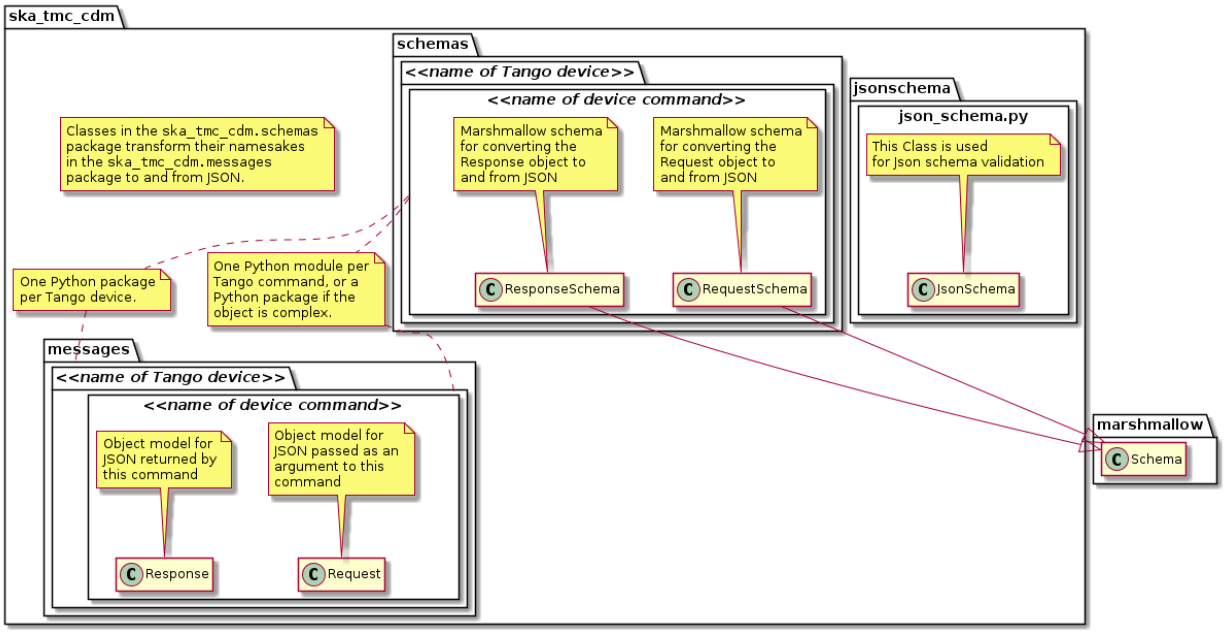


Fig. 1: Project layout and naming conventions.

The project layout and naming conventions are:

- Each Tango device has a corresponding Python sub-package in `ska_tmc_cdm.messages` and `ska_tmc_cdm.schemas`.
- Code and schema for each Tango device command are located in Python modules inside their respective package.
- Structured input for the Tango command is modelled by a `Request` object.
- Structured output from the command is modelled by a `Response` object.
- Marshmallow schema are created to transform Python `Request` and `Response` instances to an from JSON, along with any other content they contain.

4.1 Messages

The Python object model for the JSON defined in the ICD is located in the `ska_tmc_cdm.messages` package. In general, each CDM JSON entity is represented as a Python class and each CDM attribute presented as a class property.

CDM attributes can be typed as plain Python data types (strings, floats, etc.) or, where appropriate, represented by rich objects if this provides additional value to the client. For example, while astronomical coordinates are represented by floats and strings in the JSON schema, in the object model they are defined as Astropy SkyCoord instances to ensure ensure correct coordinate handling and permit easier manipulation downstream. Similarly, quantities with units could be defined as instances of Astropy Quantity to provide additional functionality.

For details on the device messages modelled by this library, see:

- *TMC CentralNode*
- *TMC SubArrayNode*
- *MCCSController*
- *MCCSSubarray*

4.2 Marshmallow Schemas

Classes to marshall the `ska_tmc_cdm.messages` objects to and from JSON are defined in the `ska_tmc_cdm.schemas` package. The ska-tmc-cdm project uses Marshmallow for JSON serialisation. Classes in the `ska_tmc_cdm.schemas` define Marshmallow schemas which are used by Marshmallow during JSON conversion.

Fig. 2: Schema mapping for objects used to communicate with TMC CentralNode device.

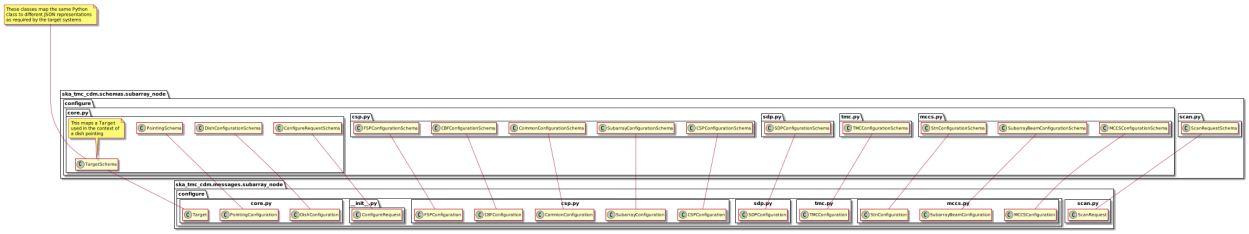


Fig. 3: Schema mapping for objects used to communicate with TMC SubArrayNode device.

4.3 JSON Schemas

The CDM library uses the SKA Telescope Model to ensure the JSON accepted and JSON generated by the library are compliant with the schema declared by the data.

The entry points for code handling JSON schema validation is located in the `ska_tmc_cdm.jsonschema` module. This module contains methods for fetching version-specific JSON schemas using interface URI and validating the structure of JSON against these schemas. Json Schema validation functionality is enabled by default with the parameter `validate=True` when converting a JSON string to CDM using `ska_tmc_cdm.schemas.CODEC.loads()` and when converting CDM to a JSON string using `ska_tmc_cdm.schemas.CODEC.dumps()`.

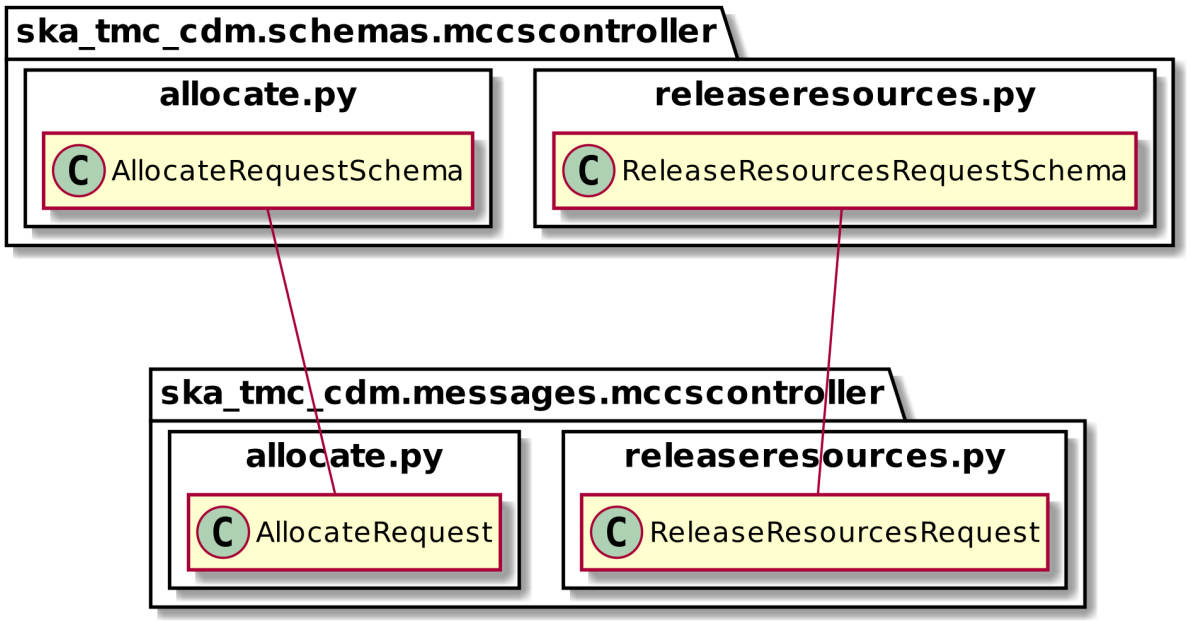


Fig. 4: Schema mapping for objects used to communicate with MCCSController device.

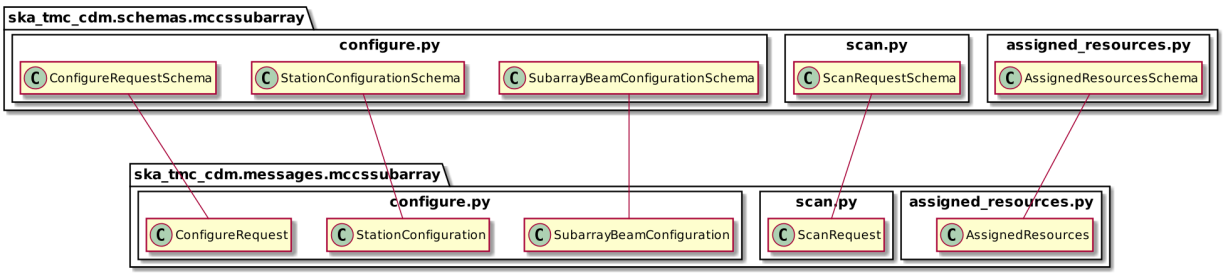
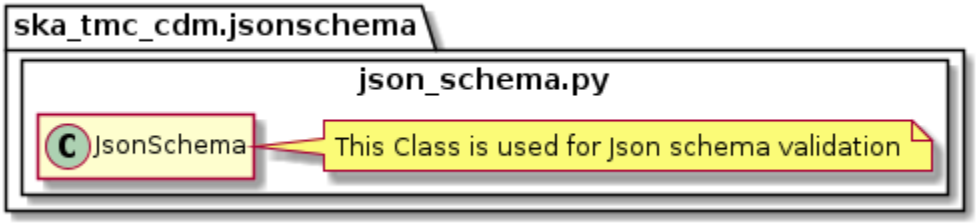


Fig. 5: Schema mapping for objects used to communicate with MCCSSubarray device.



CHAPTER

FIVE

EXTENDING THE CDM

Additional devices and applications cay use this library to communicate CDM elements wherever useful. Developers are encouraged to extend the ska-tmc-cdm project, adding object models and schemas for the structured arguments for their Tango devices.

The steps to extend the CDM are:

1. Create a new package for the Tango device in `ska_tmc_cdm.messages`.
2. For each device command, create a new module in the new package.
3. If the command accepts structured input, define a `Request` class in the module.
4. If the command returns a structured response, define a `Response` class in the module.
5. With the Python object model defined, create a corresponding package and module structure in `ska_tmc_cdm.schemas`.
6. In the schema module, define Marshmallow schemas to convert the object model classes and any structure to JSON.
7. If this is a major entity, register the schema with the `ska_tmc_cdm.schemas.CODEC` object using the `@CODEC.register_mapping` decorator.

CHAPTER

SIX

EXPAND AND CONTRACT DESIGN PATTERN IN CDM

Every PI as we gradually evolve we expect schemas to keep changing as well, some scenarios like - commands may take new keywords in addition to / or replacing existing ones; there may be change in what kind of input a keyword takes in different schemas etc.

Thus devices and helper libraries would need to support an expand/contract strategy so that devices and JSON schemas could evolve without breaking compatibility with older clients. During expand, all required version of schemas should be supported, but users are expected to migrate to using the latest one as soon as possible. There may never be a final schema since as observatory evolves the science more information may need to be communicated from start to end. However, we will certainly like to discontinue many older schemas from time to time. This will be the contract phase

Since CDM validation/serialisation library should be used to validate the JSON strings for several commands of CentralNode , SubArrayNode and hence we start there by showing how commands through CDM will support the strategy with particular example of 'release resources'.

Supporting existing and upcoming schemas with new keys in expand phase

We need to modify two message classes and two schema classes of release resource for both mid (central_node) and low (mccscontroller) telescopes respectively.

We can think of two scenarios that we need to support. Let's understand the required modifications step by step for each scenario with example of a dummy schema for mid-telescope release resources.

Scenario 1 : Small number of additional unique keys and the values that they may take is well understood.

```
{
 <existing keys> ...
 "sdp_id": "sbi-mvp01-20220919-00001", # new in this schema
 "sdp_max_length": 125.40, # new in this schema
}
```

Steps:

1. In constructor of the message class for <command>(here ReleaseResourcesRequest), add new parameters and declare them None value.

```
def __init__(
     self,
     interface: str = None,
     transaction_id: str = None,
     subarray_id: int = None,
     release_all: bool = False,
     dish_allocation: Optional[DishAllocation] = None,
     sdp_id: str = None,
     sdp_max_length: float = None,
```

(continues on next page)

(continued from previous page)

```
    ):
    # init existing keys
    ...
    self.sdp_id = sdp_id
    self.sdp_max_length = sdp_max_length

    # value errors
    ...
```

2. Inside @post_load of schema class for <command> (here 'ReleaseResourcesRequestSchema'), we modify for the same new keys as added in messages

```
@post_load
   def create_request(self, data, **_):
           ..
           sdp_id = data.get("sdp_id", None)
           sdp_max_length = data.get("sdp_max_length", None)

       return ReleaseResourcesRequest(
           ...
           sdp_id=sdp_id,
           sdp_max_length=sdp_max_length,
       )
```

3. We need to add the new keys otherwise unknown field validation error would be raised.

```
class ReleaseResourcesRequestSchema(ValidatingSchema):
# known fields
       ...
       sdp_id = fields.String()
       sdp_max_length = fields.Float()
```

Scenario 2 : While supporting multiple schemas the number of unique keys across several versions of schemas has grown very large and their validation is maintained at Telescope Model and/or the values they take is different across schemas.

1. In constructor of the message class for <command>(here ReleaseResourcesRequest), add **kwargs. We would also want to mention in constructor explicitly only those parameters which we're sure and/or very important like we want to raise value error for incorrect value etc , rest let pass through kwargs.

2. In the body of constructor we need to add one line,

```
self.__dict__.update(kwargs)
```

Finally the code snippet should look like:-

```
def __init__(
    self,
    *_, # force non-keyword args
    interface: str = None,
    transaction_id: str = None,
    subarray_id: int = None,
    release_all: bool = False,
    dish_allocation: Optional[DishAllocation] = None,
```

(continues on next page)

(continued from previous page)

```
    sdp_id: str = None,
    sdp_max_length: float = None,
    **kwargs, # arbitary keyword-value pairs
    ):
    # init existing keys
    ...
    self.sdp_id = sdp_id
    self.sdp_max_length = sdp_max_length

    # update new keywords-value pairs.
    self.__dict__.update(kwargs)

    # value errors
    ...
```

3. Inside @post_load of schema class for <command> (here 'ReleaseResourcesRequestSchema'), we modify to allow all keys to come.

```
@post_load
def create_request(self, data, **_):
    return ReleaseResourcesRequest(**data, )
```

4. However there is an additional challenge that validation error may get raised since the new keys are not mentioned inside schema class for <command>. For this we can propose the following :

i. including unknown in class Meta found in the same file. This would pass validation and work with load. But if we dump from object to JSON string these keys on the fly won't be there. To have them working in both load and dump it seems we need to explicitly know atleast the keys and mention as additional.

```
class Meta:
    unknown = INCLUDE # passes validation and load but dump won't show these keys
    additional=('subbands','dummy_key1',) # mention all such expected keys
```

ii. Since CDM extends Telescope Model we can expect Telescope Model to maintain all keys and accepted values for validation to pass anyway.

Expectations in Contract phase

There should be additional challenges in contract phase that will be understood as we evolve. However for now we expect to:

i. Remove support of kwargs

ii. Mention all keys by hand for the final schema.

iii. Have logical default values instead of declaring with NonelNull values. Remove null filtering in schemas.

Users should not get away without correct keys and valid values in contract phase.

How to use during expand phase

from ska_tmc_cdm.schemas import CODEC

1. If we have some JSON-formatted string release_input_str

```
{
 "interface":"https://schema.skao.int/ska-tmc-releaseresources/2.0",
```

(continues on next page)

(continued from previous page)

```
 "transaction_id":"txn-....-00001",
 "subarray_id":1,
 "release_all":true,
 "receptor_ids":[],
 "sdp_max_length": 125.40, # new key but mentioned in message, schema classes
 "subbands": [0.55e9, 0.95e9, 186], # on the fly
 "dummy_key1":"val1" # on the fly
}
```

Convert the JSON to a Python object

```
req=CODEC.loads(ReleaseResourcesRequest, release_input_str) # requested object
```

2. If we received the object and want to convert it to JSON which may be used in a DeviceProxy call

```
json_str=CODEC.dumps(req) # from object to JSON string
```

3. Inside @post_load of schema class for <command> (here 'ReleaseResourcesRequestSchema') we expect the same message class constructor 'ReleaseResourcesRequest' to be able to support across different schemas using kwargs.

```
# expand
request = ReleaseResourcesRequest(
     transaction_id="tma1",
     subarray_id=1,
     dish_allocation=DishAllocation(receptor_ids=["ac", "b", "aab"]),
     sdp_id="sbi-mvp01-20220919-00001", # new in this schema
     sdp_max_length=125.40, # new in this schema
     subbands=[0.55e9, 0.95e9, 186], # arbitary new key-value captured
     release_all=False,
     )
# contract
request = ReleaseResourcesRequest(
     transaction_id="tma1",
     subarray_id=1,
     dish_allocation=DishAllocation(receptor_ids=["ac", "b", "aab"]),
     sdp_id="sbi-mvp01-20220919-00001", # new in this schema
     )
```

Resources

1. A prototype can be found at https://gitlab.com/ska-telescope/ska-tmc-cdm/-/tree/nak-74-expand-contract-design-pattern.

2. Dummy schema for mid telescope release resource.

```
{
 "interface": https://schema.skao.int/ska-tmc-releaseresources/2.2, #optional
 "subarray_id": 1,
 "release_all": False,
 "receptor_ids": ["ac", "b", "aab"],
 "sdp_id": "sbi-mvp01-20220919-00001", # new in this schema
 "sdp_max_length": 125.40, # new in this schema
 "subbands: [0.55e9, 0.95e9, 186] # arbitary new key-value captured by kwargs
}
```

3. Dummy schema for low telescope release resource.

```
{
 "interface": https://schema.skao.int/ska-tmc-releaseresources/2.2, #optional
 "subarray_id": 1,
 "release_all": False,
 "subarray_beam_ids": [3], # new in this schema
 "channels": [[3, 4]], # new in this schema
}
```

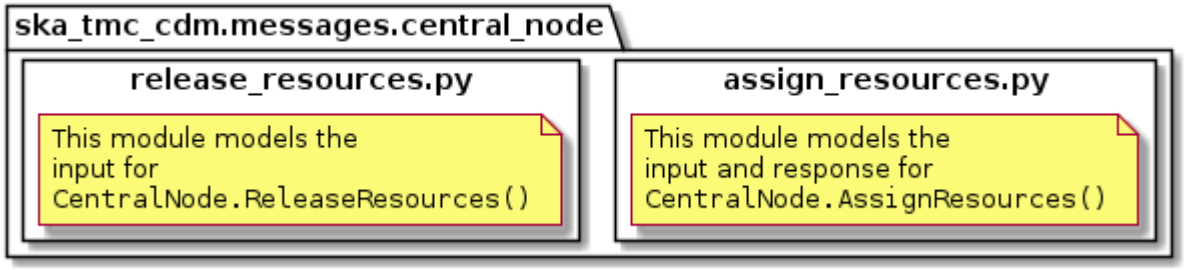
CHAPTER

SEVEN

TMC CENTRALNODE

7.1 Overview

Sub-array resource allocation is achieved via communication with a TMC CentralNode device. The `centralnode` package models the JSON input and responses for TMC CentralNode commands. The contents of this package are shown in the figure below.



Classes in the *assign_resources.py* module model the arguments for the `CentralNode.AssignResources()` command.

Classes in the *release_resources.py* module model the arguments for the `CentralNode.ReleaseResources()` command.

7.2 assign_resources.py

The `assign_resources.py` module models the the JSON input and response for a `CentralNode.AssignResources()` command.

Example PI16 JSON input modelled by `AssignResourcesRequest` for MID:

```
{
   "interface":"https://schema.skao.int/ska-tmc-assignresources/2.1",
   "transaction_id":"txn-....-00001",
   "subarray_id":1,
   "dish":{
      "receptor_ids":[
         "0001"
      ]
   },
```

(continues on next page)

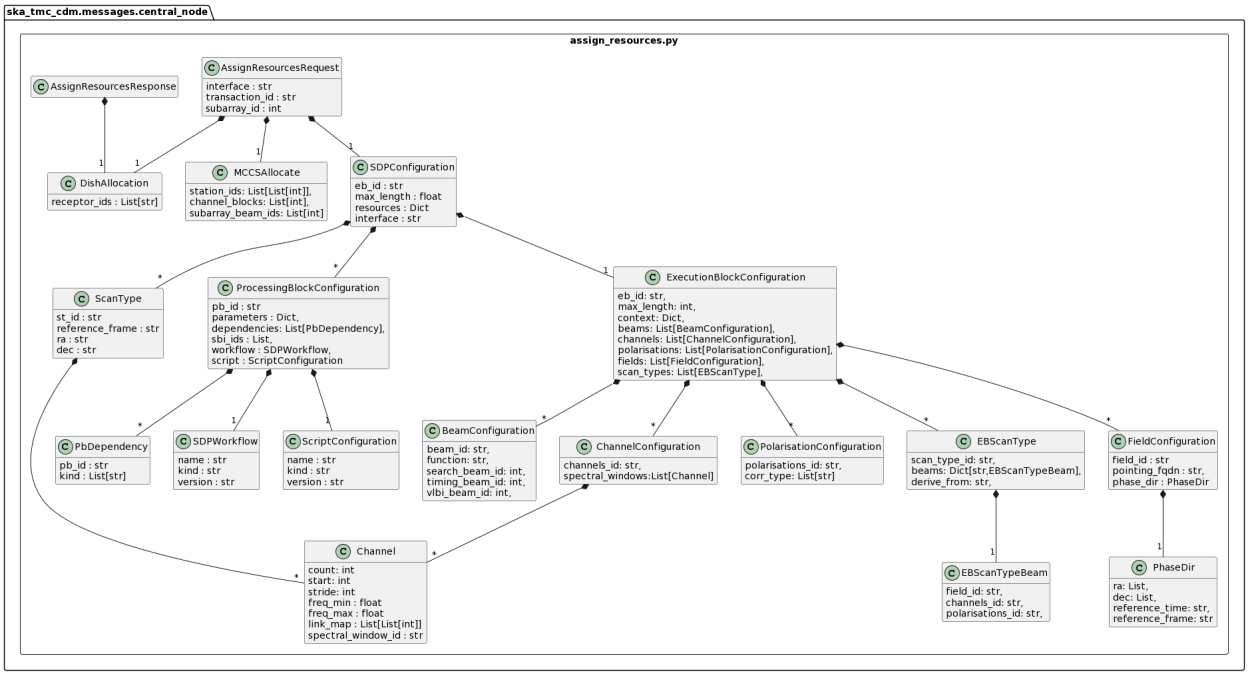


Fig. 1: assign_resources.py object model

(continued from previous page)

```
  "sdp":{
     "interface":"https://schema.skao.int/ska-sdp-assignres/0.4",
     "resources":{
        "receptors":[
           "SKA001",
           "SKA002",
           "SKA003",
           "SKA004"
        ]
     },
     "execution_block":{
        "eb_id":"eb-test-20220916-00000",
        "context":{

        },
        "max_length":3600.0,
        "beams":[
           {
              "beam_id":"vis0",
              "function":"visibilities"
           }
        ],
        "scan_types":[
           {
              "scan_type_id":".default",
              "beams":{
                 "vis0":{
```

(continues on next page)

(continued from previous page)

```
                    "channels_id":"vis_channels",
                    "polarisations_id":"all"
                }
            }
        },
        {
            "scan_type_id":"target:a",
            "derive_from":".default",
            "beams":{
                "vis0":{
                    "field_id":"field_a"
                }
            }
        },
        {
            "scan_type_id":"calibration:b",
            "derive_from":".default",
            "beams":{
                "vis0":{
                    "field_id":"field_b"
                }
            }
        }
    ],
    "channels":[
        {
            "channels_id":"vis_channels",
            "spectral_windows":[
                {
                    "spectral_window_id":"fsp_1_channels",
                    "count":4,
                    "start":0,
                    "stride":2,
                    "freq_min":350000000.0,
                    "freq_max":368000000.0,
                    "link_map":[
                        [
                            0,
                            0
                        ],
                        [
                            200,
                            1
                        ],
                        [
                            744,
                            2
                        ],
                        [
                            944,
                            3
                        ]
```

(continues on next page)

(continued from previous page)


```
                    ]
                }
            ]
        }
    ],
    "polarisations":[
        {
            "polarisations_id":"all",
            "corr_type":[
                "XX",
                "XY",
                "YX",
                "YY"
            ]
        }
    ],
    "fields":[
        {
            "field_id":"field_a",
            "phase_dir":{
                "ra":[
                    123.0
                ],
                "dec":[
                    -60.0
                ],
                "reference_time":"...",
                "reference_frame":"ICRF3"
            },
            "pointing_fqdn":"..."
        },
        {
            "field_id":"field_b",
            "phase_dir":{
                "ra":[
                    123.0
                ],
                "dec":[
                    -60.0
                ],
                "reference_time":"...",
                "reference_frame":"ICRF3"
            },
            "pointing_fqdn":"..."
        }
    ]
},
"processing_blocks":[
    {
        "pb_id":"pb-test-20220916-00000",
        "script":{
            "kind":"realtime",
```


(continues on next page)

(continued from previous page)

```
                "name":"test-receive-addresses",
                "version":"0.5.0"
            },
            "sbi_ids":[
                "sbi-test-20220916-00000"
            ],
            "parameters":{

            }
        }
    ]
  }
}
```

For PI14 JSON, Please refer confluence schema page

Example JSON response modelled by `AssignResourcesResponse` for MID:

```
{
  "dish": {
    "receptor_ids_allocated": ["0001", "0002"]
  }
}
```

Example JSON input modelled by `AssignResourcesRequest` for LOW:

```
{
  "interface": "https://schema.skao.int/ska-low-tmc-assignresources/2.0",
  "subarray_id": 1,
  "mccs": {
      "subarray_beam_ids": [1],
      "station_ids": [[1,2]],
      "channel_blocks": [3]
   }
}
```

7.3 release_resources.py

The `release_resources.py` module models the input JSON for a `CentralNode.ReleaseResources()` command.

Example ReleaseResourcesRequest JSON that requests specific dishes be released from a sub-array:

```
{
  "interface": "https://schema.skao.int/ska-tmc-releaseresources/2.1",
  "transaction_id": "txn-mvp01-20200325-00001",
  "subarray_id": 1,
  "receptor_ids": ["0001", "0002"]
}
```

Example JSON that requests all sub-array resources be released:

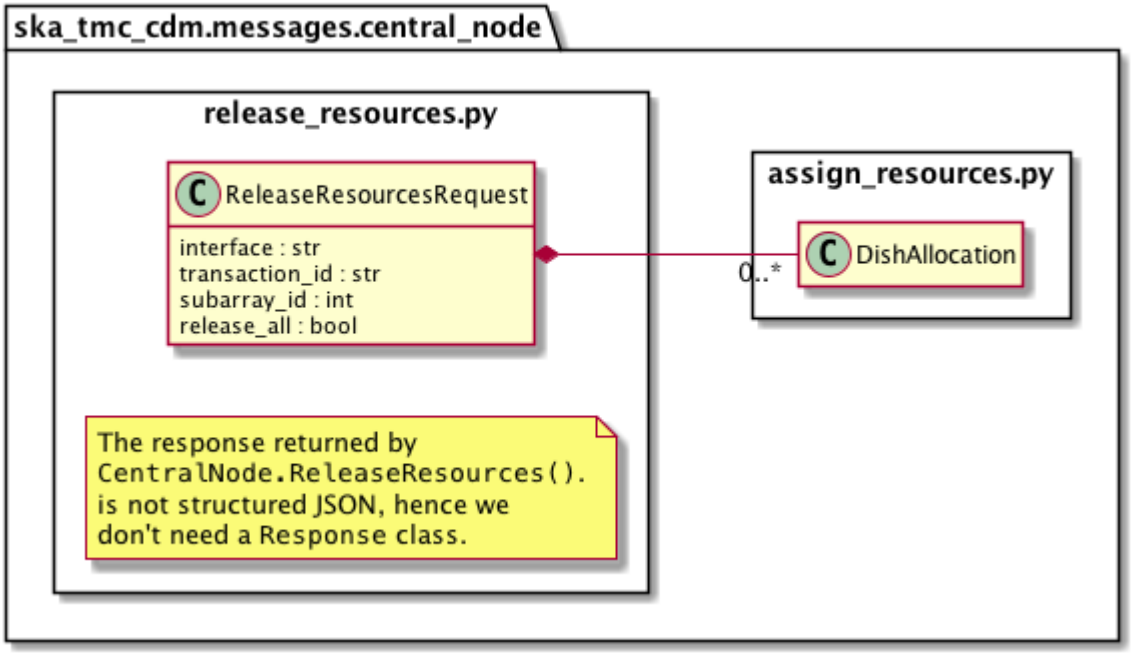


Fig. 2: release_resources.py object model

```
{
  "interface": "https://schema.skao.int/ska-tmc-releaseresources/2.1",
  "transaction_id": "txn-mvp01-20200325-00001",
  "subarray_id": 1,
  "release_all": true
}
```

Example JSON that requests all sub-array resources be released for LOW:

```
{
  "interface": "https://schema.skao.int/ska-low-tmc-releaseresources/2.0",
  "subarray_id": 1,
  "release_all": true
}
```

CHAPTER

EIGHT

VALIDATING JSON SCHEMA THROUGH CDM IN CENTRAL NODE

'ska-tmc-cdm' validation/serialisation library contains message and schema classes for several commands of CentralNode , SubArrayNode of both mid and low telescope. The classes for message provide the way to create Python object for the requested command with correct attributes that comes from a JSON string which must contain just the right keys and their valid values. This input JSON is first validated in the classes for schema and then passed to constructor of message class for finally creating object. Further the CDM extends the 'Telescope Model' which should contain all logical checks related to the validity of attribute values for a given command.

The whole purpose of maintaining all the classes for message and schema at CDM is so that other TMC interfaces can communicate with all TANGO devices without requiring to validate by its own the available JSON. Otherwise there will be duplicacy of logic as well as hard time maintaining the different components by different teams working internationally.

Here we shall see one such example, where Central Node shall use CDM to validate the received JSON string for release resources request replacing its local validation.

Steps

1. Import from 'ska-tmc-cdm' message classes for the command here ReleaseResources as well as CODEC from schemas in following way

```
# for mid telescope
from ska_tmc_cdm.messages.central_node.release_resources import
(
    ReleaseResourcesRequest,
)
# for low telescope
from ska_tmc_cdm.messages.mccscontroller.releaseresources import
(
    ReleaseResourcesRequest as ReleaseResourcesRequestLow,
)
```

CODEC provides the loads and dumps methods for converting JSON String—>Python object and vice versa for classes defined in ska_tmc_cdm.message

```
from ska_tmc_cdm.schemas import CODEC
```

2. Find the appropriate place where currently the JSON string is being validated and result code, error message is being returned.

In this example, for release resources we found one validate_input_json method in release_resources_command.py.

```
try:
    # created python dictionary parsing from input json string
```

(continues on next page)

(continued from previous page)

```
    jsonArgument = json.loads(argin)
except Exception as e:
    # ResultCode Failed and custom error message
    ...
# all validations here and if all success then
return ResultCode.OK, ""
```

3. Changes to be considered :

i. As we saw during import there are two seperate message classes - **one for mid telescope and other for low telescope** so we would require **two validate methods in place of one**.

ii. Replace **json.loads->CODEC.loads within try block** and **any error message** should **come from CDM extending Telescope Model** if request JSON is invalid for the command be it by syntax or logical.

iii. For the time being some validations which are not been checked at CDM and/or Telescope Model need to be done within else block of try. Finally code snippet should look like:

```
try:
    # creation of cdm object from input json argin.
    release_request = CODEC.loads(ReleaseResourcesRequest, argin)
except Exception as excep:
    # return ResultCode Failed and exception message
    ...
else:
    # remaining custom local validation
    ...

    # if no error occurred
    return ResultCode.OK, ""
```

Scenarios for unit tests

We can only be sure that this approach worked by writing unit-tests where we see ResultCode to be Ok and successfully requested object gets created when our JSON input is valid. In other case, three error scenarios we have tried for mid-telescope release resource to verify the message is indeed appropriate and comes from CDM :

Test scenario 1: JSON is missing (a mandatory key) sub array id.

Test scenario 2: The input JSON has misspelt 'release_all' key as 'releaseall' – invalid key error.

Test scenario 3: The input JSON string has provided number to 'release_all' key which takes either True/False - invalid value error.

Resources

1. A proof of concept for replacing custom JSON validation for commands in Central Node (above) can be found at https://gitlab.com/ska-telescope/ska-tmc/ska-tmc-centralnode/-/tree/nak-75-replacing-customjsonparsing-cdmobj.

2. Central Node is a coordinator of the complete Telescope Monitoring and Control (TMC) system. Find ska-tmc-centralnode repository at https://gitlab.com/ska-telescope/ska-tmc/ska-tmc-centralnode.

3. SKA Control Data Model provides Python/JSON serialisation for the command arguments for various TMC interfaces with other subsystems. Find ska-tmc-cdm repository at https://gitlab.com/ska-telescope/ska-tmc-cdm/

4. SKA Telescope Model is a dynamic computational model to answer all queries about the state of the Telescope. Find this library at https://gitlab.com/ska-telescope/ska-telmodel

CHAPTER

NINE

TMC SUBARRAYNODE

9.1 Overview

Sub-array configuration and scan control is achieved via communication with a TMC SubArrayNode Tango device. The diagram below shows the packages and high-level object model used for telescope configuration and control.

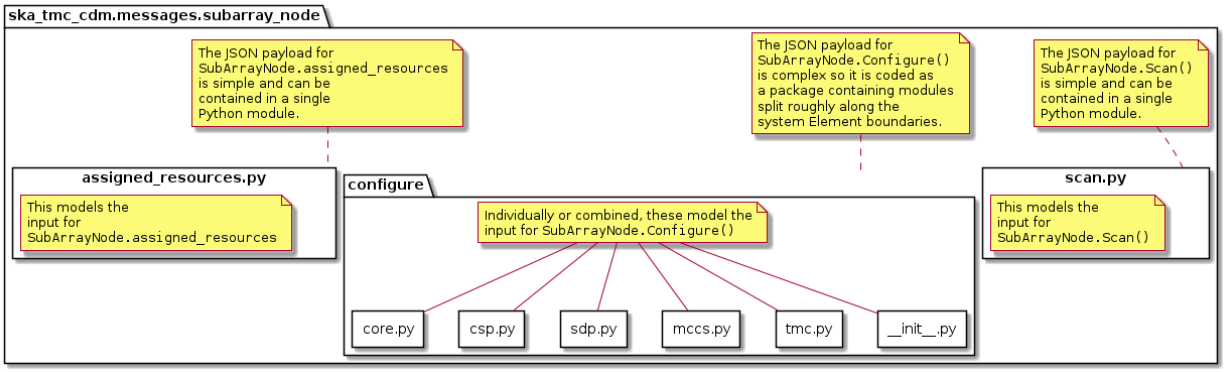


Fig. 1: High-level object model for communication with a TMC SubArrayNode device.

Classes in the *configure* package model the arguments for the `SubArrayNode.Configure()` command.

Classes in the *scan.py* module model the arguments for the `SubArrayNode.Scan()` command.

9.2 configure

The configuration JSON is complex, the module is split between several modules. The `configure` package contains five modules:

- __init__.py
- *core.py*
- *tmc.py*
- *csp.py*
- *sdp.py*
- *mccs.py*

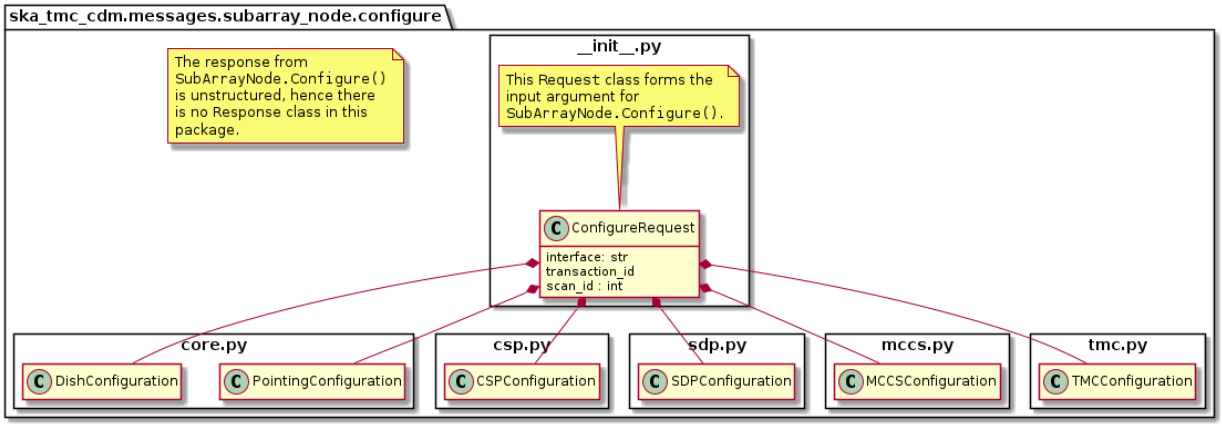


Fig. 2: High-level overview of the configure package

`__init__.py` references sub-modules in the main ConfigureRequest object, as illustrated in the diagram above.

In the context of a full JSON example object, __init__.py defines the a basic container object, while the sub-modules define the details.

```
# JSON modelled specifically by __init__.py
{
  "scanID": 12345,
  ...
}
```

9.2.1 core.py

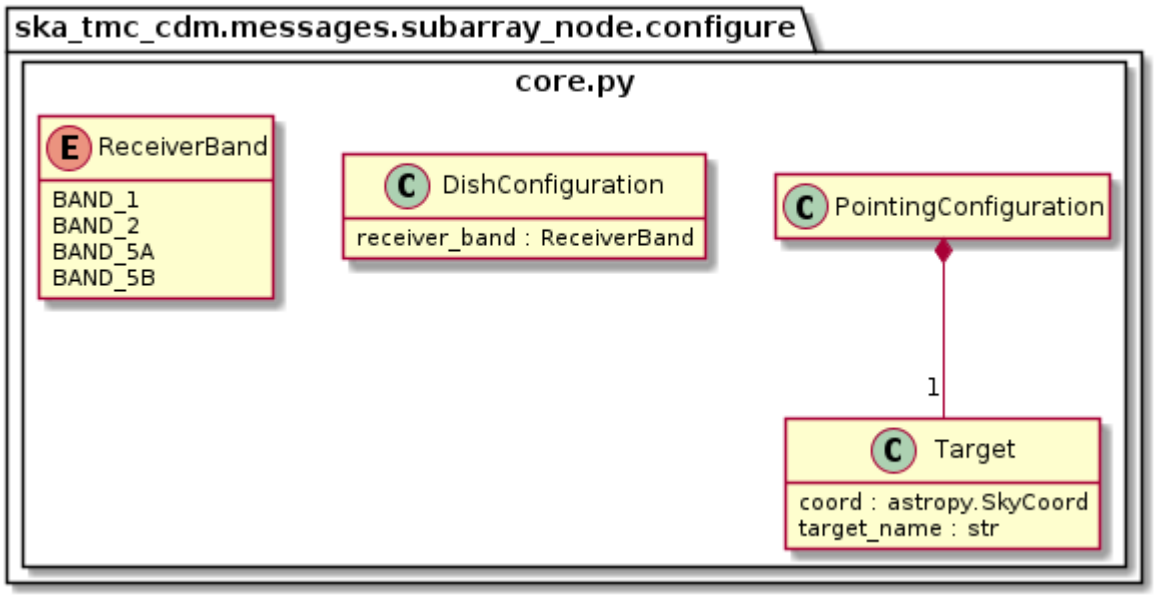


Fig. 3: core.py object model

The `core.py` module models receptor pointing and receiver band JSON elements. In the context of a full CDM JSON object, the elements this maps to are:

```
# JSON modelled specifically by core.py
{
  ...
  "pointing": {
    "target": {
      "reference_frame":"ICRS",
      "name": "NGC6251",
      "ra": 1.0,
      "dec": 1.0
    },
  },
  ...
  "dish": {
    "receiver_band": "1"
  }
  ....
}
```

9.2.2 tmc.py

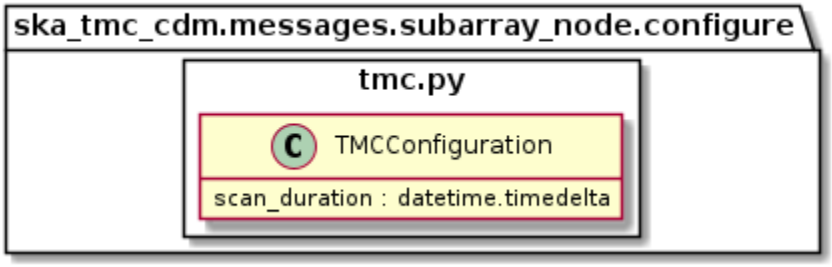


Fig. 4: tmc.py object model

The `tmc.py` module models TMC configuration JSON elements. Below is an example JSON command argument that this code can model.

```
# JSON modelled specifically by tmc.py
{
  "tmc": {
    "scan_duration": 10.0,
  }
}
```

9.2.3 csp.py

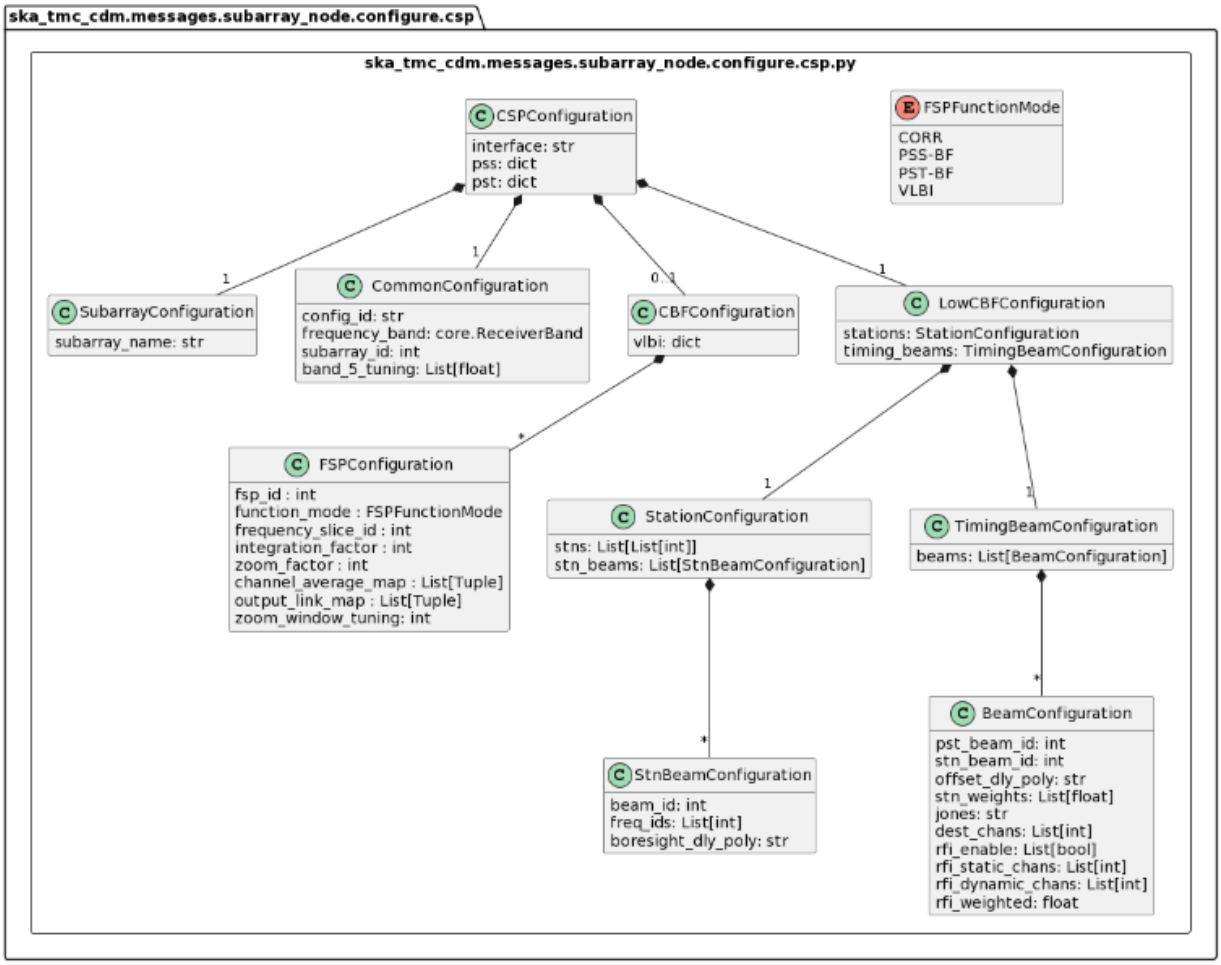


Fig. 5: csp.py object model

The `csp.py` module models CSP configuration JSON elements. In the context of a full CDM JSON object, the elements this maps to are:

```
#Mid JSON specifically by csp.py
{
  ...
  csp": {
  "interface": "https://schema.skao.int/ska-csp-configure/2.0",
  "subarray": {
    "subarray_name": "science period 23"
  },
  "common": {
    "config_id": "sbi-mvp01-20200325-00001-science_A",
    "frequency_band": "1",
    "subarray_id": 1
  },
  "cbf": {
```

(continues on next page)

(continued from previous page)

```
    "fsp": [
      {
        "fsp_id": 1,
        "function_mode": "CORR",
        "frequency_slice_id": 1,
        "integration_factor": 1,
        "zoom_factor": 0,
        "channel_averaging_map": [
          [
            0,
            2
          ],
          [
            744,
            0
          ]
        ],
        "channel_offset": 0,
        "output_link_map": [
          [
            0,
            0
          ],
          [
            200,
            1
          ]
        ]
      },
      {
        "fsp_id": 2,
        "function_mode": "CORR",
        "frequency_slice_id": 2,
        "integration_factor": 1,
        "zoom_factor": 1,
        "channel_averaging_map": [
          [
            0,
            2
          ],
          [
            744,
            0
          ]
        ],
        "channel_offset": 744,
        "output_link_map": [
          [
            0,
            4
          ],
          [
```

(continues on next page)

(continued from previous page)

```
            200,
            5
          ]
        ],
        "zoom_window_tuning": 650000
      }
    ],
    "vlbi": {

    }
  },
  "pss": {

  },
  "pst": {

  },
 },
  ...
}

#Low JSON  specifically by csp.py
  {
  "interface": "https://schema.skao.int/ska-csp-configure/2.0",
  "subarray": {
    "subarray_name": "science period 23"
  },
  "common": {
    "config_id": "sbi-mvp01-20200325-00001-science_A",

  },
  "lowcbf": {
    "stations": {
      "stns": [
        [
          1,
          0
        ],
        [
          2,
          0
        ],
        [
          3,
          0
        ],
        [
          4,
          0
        ]
      ],
      "stn_beams": [
```

(continues on next page)

(continued from previous page)


```
        {
          "beam_id": 1,
          "freq_ids": [
            64,
            65,
            66,
            67,
            68,
            68,
            70,
            71
          ],
          "boresight_dly_poly": "url"
        }
      ]
    },
    "timing_beams": {
      "beams": [
        {
          "pst_beam_id": 13,
          "stn_beam_id": 1,
          "offset_dly_poly": "url",
          "stn_weights": [
            0.9,
            1.0,
            1.0,
            0.9
          ],
          "jones": "url",
          "dest_chans": [
            128,
            256
          ],
          "rfi_enable": [
            true,
            true,
            true
          ],
          "rfi_static_chans": [
            1,
            206,
            997
          ],
          "rfi_dynamic_chans": [
            242,
            1342
          ],
          "rfi_weighted": 0.87
        }
      ]
    },
```


(continues on next page)

(continued from previous page)

```
  }
}
```

9.2.4 sdp.py

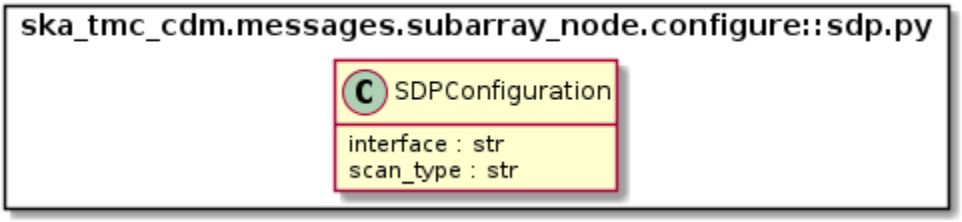


Fig. 6: sdp.py object model

The `sdp.py` module models SDHP configuration JSON elements. In the context of a full CDM JSON object, the elements this maps to are:

```
# JSON modelled specifically by sdp.py
{
  ...
  "sdp": {
    "scan_type": "science_A"
  },
  ...
}
```

9.2.5 mccs.py

The `mccs.py` module models MCCS configuration JSON elements. In the context of a full CDM JSON object, the elements this maps to are:

```
# JSON modelled specifically by mccs.py
{
  "mccs": {
      "stations": [
        {
          "station_id": 1
        },
        {
          "station_id": 2
        }
      ],
      "subarray_beams": [
        {
          "subarray_beam_id": 1,
          "station_ids": [1, 2],
          "update_rate": 0,
```

(continues on next page)

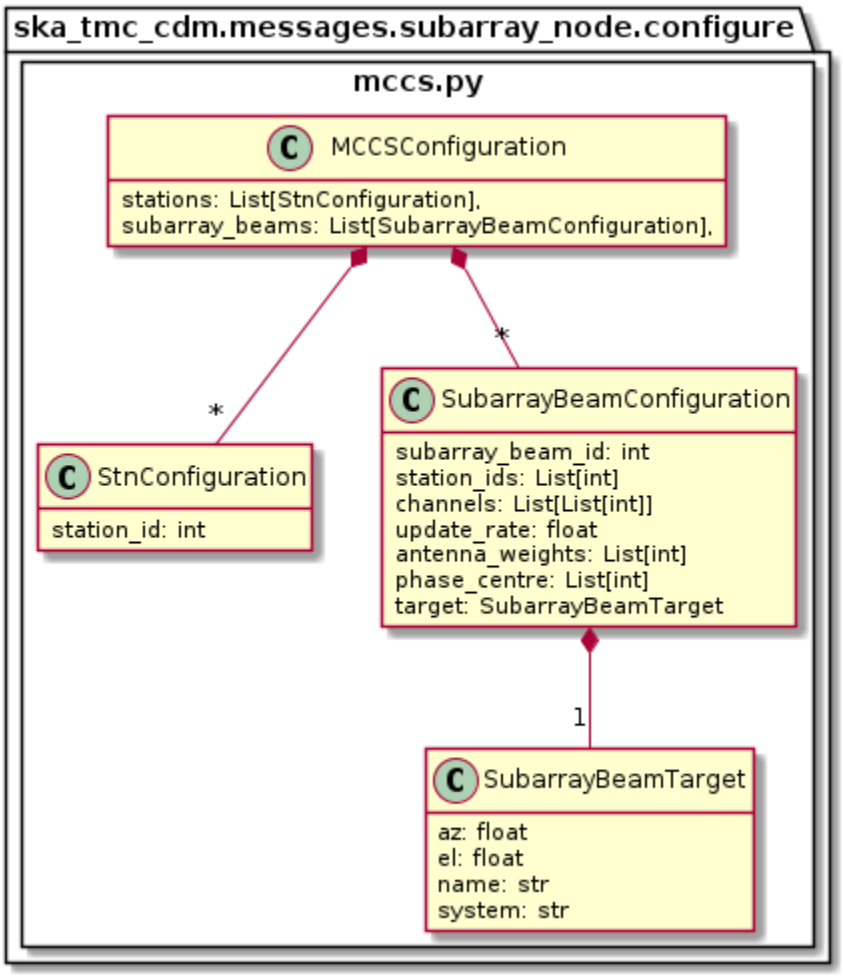


Fig. 7: mccs.py object model

(continued from previous page)

```
          "channels": [
            [0, 8, 1, 1],
            [8, 8, 2, 1],
            [24, 16, 2, 1]
          ],
          "antenna_weights": [1, 1, 1],
          "phase_centre": [0, 0],
          "target": {
            "system": "HORIZON",
            "name": "DriftScan",
            "az": 180,
            "el": 45
          }
        }
      ]
   }
}
```

9.3 assigned_resources.py

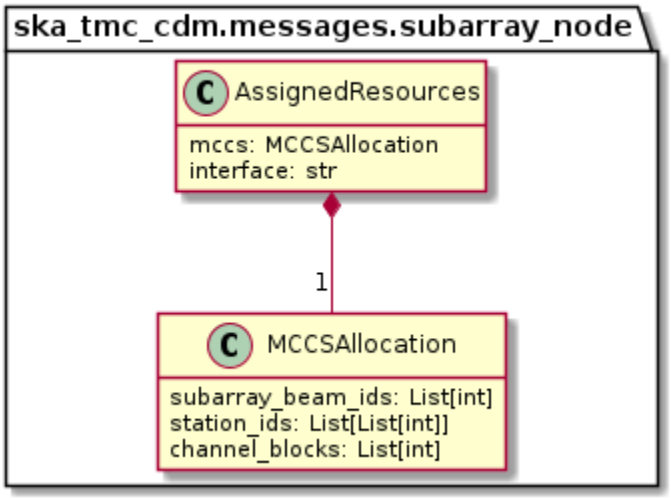


Fig. 8: assigned_resources.py object model

The `assigned_resources.py` module describes which resources have been assigned to the sub-array.

Examples below depict a populated sub-array and an empty one:

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-assignedresources/2.0",
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [[1,2]],
        "channel_blocks": [3]
```

(continues on next page)

(continued from previous page)

```
    }
}
```

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-assignedresources/2.0",
    "mccs": {
        "subarray_beam_ids": [],
        "station_ids": [],
        "channel_blocks": []
    }
}
```

9.4 scan.py

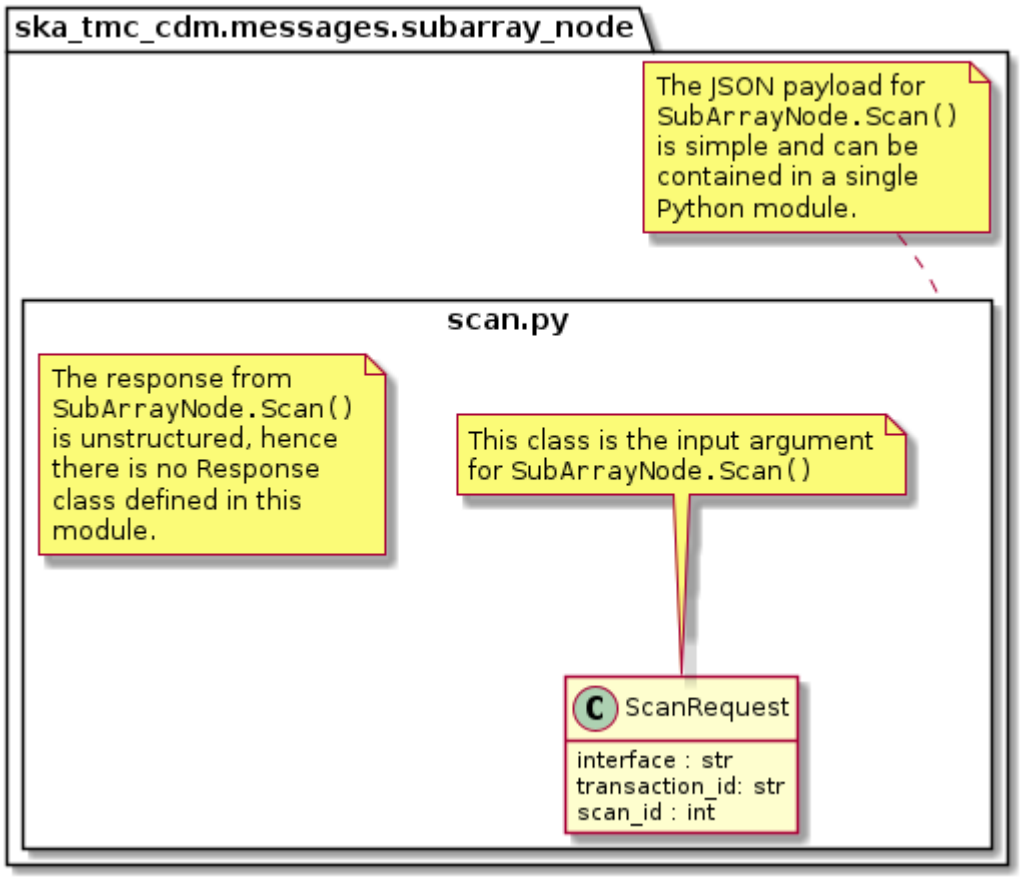


Fig. 9: scan.py object model

The `scan.py` module models the argument for the `SubArrayNode.scan()` command. Below is an example JSON command argument that this code can model.

```
{
  "interface": "https://schema.skao.int/ska-tmc-scan/2.1",
  "transaction_id": "txn-12345",
  "scan_id": 2
}
```

9.5 Example configuration JSON for MID

```
{
  "interface": "https://schema.skao.int/ska-tmc-configure/2.1",
  "transaction_id": "txn-....-00001",
  "pointing": {
    "target": {
      "reference_frame": "ICRS",
      "target_name": "Polaris Australis",
      "ra": "21:08:47.92",
      "dec": "-88:57:22.9"
    }
  },
  "dish": {
    "receiver_band": "1"
  },
  "csp": {
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
      "subarray_name": "science period 23"
    },
    "common": {
      "config_id": "sbi-mvp01-20200325-00001-science_A",
      "frequency_band": "1",
      "subarray_id": 1
    },
    "cbf": {
      "fsp": [
        {
          "fsp_id": 1,
          "function_mode": "CORR",
          "frequency_slice_id": 1,
          "integration_factor": 1,
          "zoom_factor": 0,
          "channel_averaging_map": [
            [
              0,
              2
            ],
            [
              744,
              0
            ]
          ],
```

(continues on next page)

(continued from previous page)

```
          "channel_offset": 0,
          "output_link_map": [
            [
              0,
              0
            ],
            [
              200,
              1
            ]
          ]
        },
        {
          "fsp_id": 2,
          "function_mode": "CORR",
          "frequency_slice_id": 2,
          "integration_factor": 1,
          "zoom_factor": 1,
          "channel_averaging_map": [
            [
              0,
              2
            ],
            [
              744,
              0
            ]
          ],
          "channel_offset": 744,
          "output_link_map": [
            [
              0,
              4
            ],
            [
              200,
              5
            ]
          ],
          "zoom_window_tuning": 650000
        }
      ],
      "vlbi": {

      }
    },
    "pss": {

    },
    "pst": {

    }
```

(continues on next page)

(continued from previous page)

```
  },
  "sdp": {
    "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
    "scan_type": "science_A"
  },
  "tmc": {
    "scan_duration": 10.0
  }
}
```

9.6 Example configuration JSON for LOW

```
   {
 "interface": "https://schema.skao.int/ska-low-tmc-configure/3.0",
 "transaction_id": "txn-....-00001",
 "mccs": {
   "stations": [
     {
       "station_id": 1
     },
     {
       "station_id": 2
     }
   ],
   "subarray_beams": [
     {
       "subarray_beam_id": 1,
       "station_ids": [
         1,
         2
       ],
       "update_rate": 0.0,
       "channels": [
         [
           0,
           8,
           1,
           1
         ],
         [
           8,
           8,
           2,
           1
         ],
         [
           24,
           16,
           2,
           1
```

(continues on next page)

(continued from previous page)

```
        ]
      ],
      "antenna_weights": [
        1.0,
        1.0,
        1.0
      ],
      "phase_centre": [
        0.0,
        0.0
      ],
      "target": {
        "reference_frame": "HORIZON",
        "target_name": "DriftScan",
        "az": 180.0,
        "el": 45.0
      }
    }
  ]
},
"sdp": {
  "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
  "scan_type": "science_A"
},
"csp": {
  "interface": "https://schema.skao.int/ska-csp-configure/2.0",
  "subarray": {
    "subarray_name": "science period 23"
  },
  "common": {
    "config_id": "sbi-mvp01-20200325-00001-science_A",
  },
  "lowcbf": {
    "stations": {
      "stns": [
        [
          1,
          0
        ],
        [
          2,
          0
        ],
        [
          3,
          0
        ],
        [
          4,
          0
        ]
      ],
```

(continues on next page)

(continued from previous page)


```
      "stn_beams": [
        {
          "beam_id": 1,
          "freq_ids": [
            64,
            65,
            66,
            67,
            68,
            68,
            70,
            71
          ],
          "boresight_dly_poly": "url"
        }
      ]
    },
    "timing_beams": {
      "beams": [
        {
          "pst_beam_id": 13,
          "stn_beam_id": 1,
          "offset_dly_poly": "url",
          "stn_weights": [
            0.9,
            1.0,
            1.0,
            0.9
          ],
          "jones": "url",
          "dest_chans": [
            128,
            256
          ],
          "rfi_enable": [
            true,
            true,
            true
          ],
          "rfi_static_chans": [
            1,
            206,
            997
          ],
          "rfi_dynamic_chans": [
            242,
            1342
          ],
          "rfi_weighted": 0.87
        }
      ]
    }
```


(continues on next page)

(continued from previous page)

```
    }
  },
  "tmc": {
    "scan_duration": 10.0
  }
}
```

CHAPTER

TEN

MCCSSUBARRAY

10.1 Overview

MCCS configuration and scan control is achieved via communication with a MCCSSubarray Tango device. Additionally, the MCCSSubarray device presents a device attribute that lists the resources allocated to that subarray.

The diagram below shows the packages and high-level object model used for communication with an MCCSSubarray device.

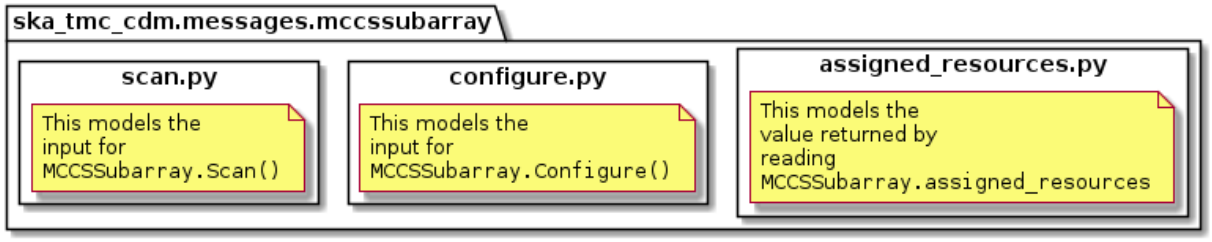


Fig. 1: High-level object model for communication with a MCCSSubarray device.

Classes in the *assigned_resources.py* module model the resource allocation status JSON string returned by the `MCCSSubarray.assigned_resources` attribute.

Classes in the *configure.py* module model the arguments for the `MCCSSubarray.Configure()` command.

Classes in the *scan.py* module model the arguments for the `MCCSSubarray.Scan()` command.

10.2 assigned_resources.py

The `assigned_resources.py` module models the the JSON returned by reading the `MCCSSubarray.assigned_resources` attribute.

Example JSON returned by `MCCSSubarray.assigned_resources`:

```
{
  "interface": "https://schema.skao.int/ska-low-mccs-assignedresources/2.0",
  "subarray_beam_ids": [1],
  "station_ids": [[1,2]],
  "channel_blocks": [3]
}
```

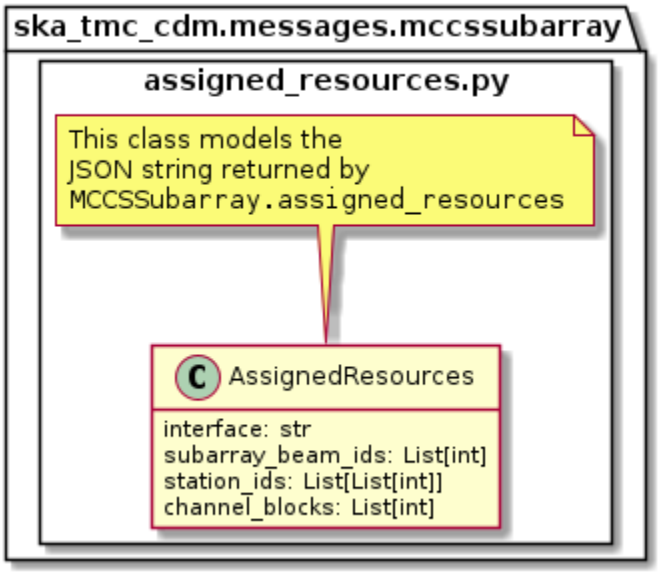


Fig. 2: High-level overview of the assigned_resources module

10.3 configure.py

The `configure.py` module models the the JSON input for an `MCCSSubarray.configure()` command.

Example JSON input for an `MCCSSubarray.Configure` call:

```
{
  "interface": "https://schema.skao.int/ska-low-mccs-configure/2.0",
  "stations":[
    {
      "station_id": 1
    },
    {
      "station_id":2
    }
  ],
  "subarray_beams": [
    {
      "subarray_beam_id":1,
      "station_ids": [1, 2],
      "update_rate": 0.0,
      "channels": [
        [0,8,1,1],
        [8,8,2,1],
        [24,16,2,1]
      ],
      "sky_coordinates": [0.0, 180.0, 0.0, 45.0, 0.0],
      "antenna_weights": [1.0, 1.0, 1.0],
      "phase_centre": [0.0, 0.0]
    }
```

(continues on next page)

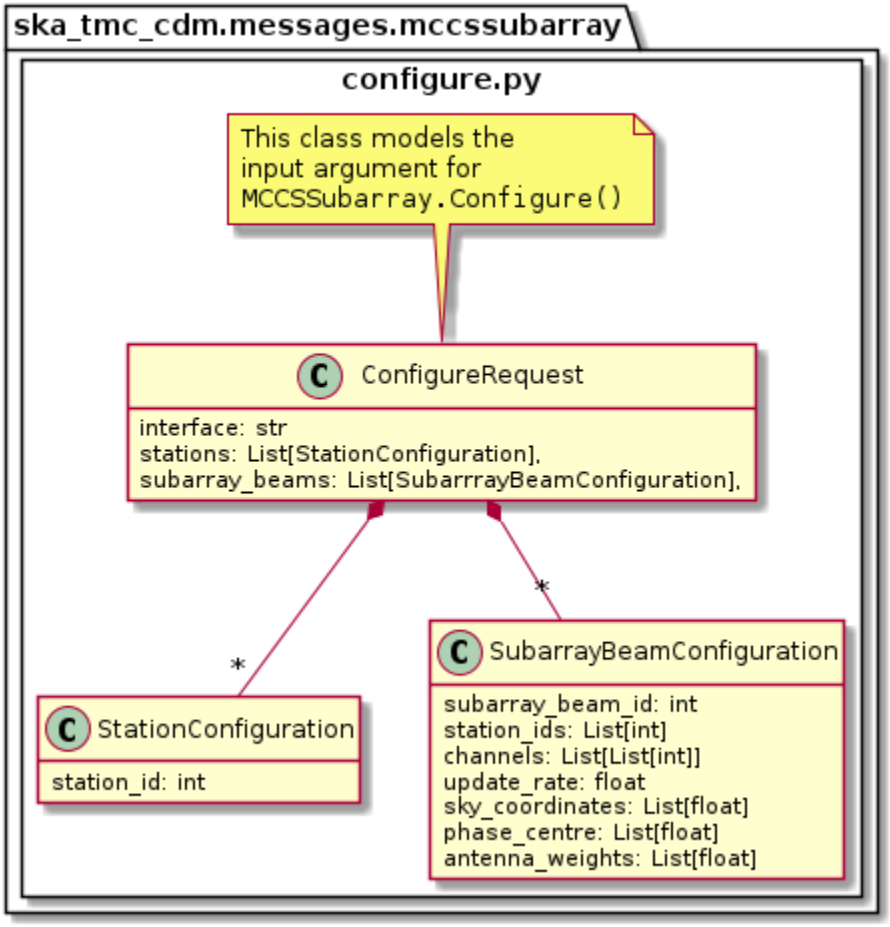


Fig. 3: High-level overview of the configure module

(continued from previous page)

```
  ]
}
```

10.4 scan.py

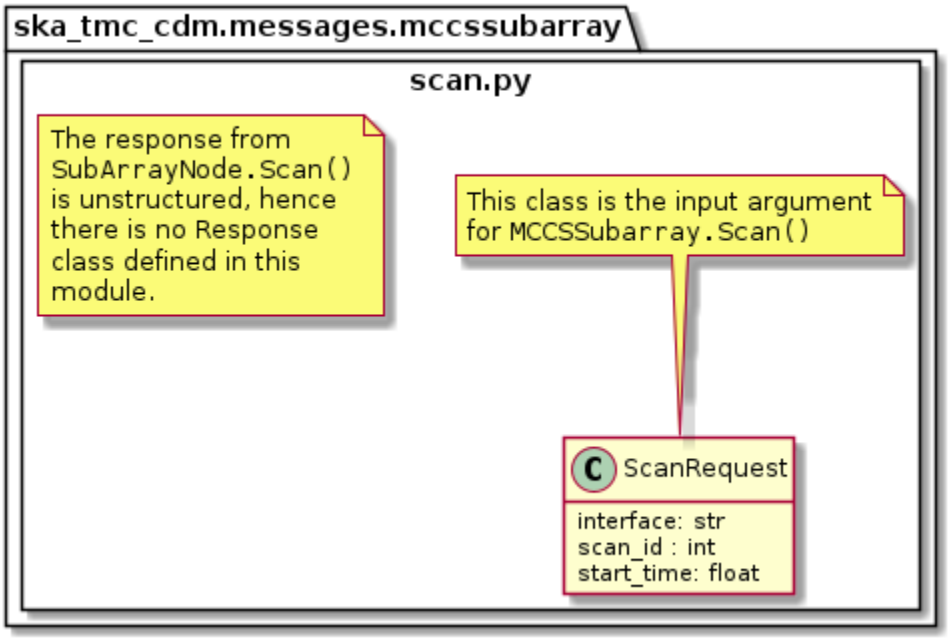


Fig. 4: scan.py object model

The `scan.py` module models the argument for the `MCCSSubarray.scan()` command.

Example JSON input for an `MCCSSubarray.scan()` call:

```
{
  "interface": "https://schema.skao.int/ska-low-mccs-scan/2.0",
  "scan_id":1,
  "start_time": 0.0
}
```

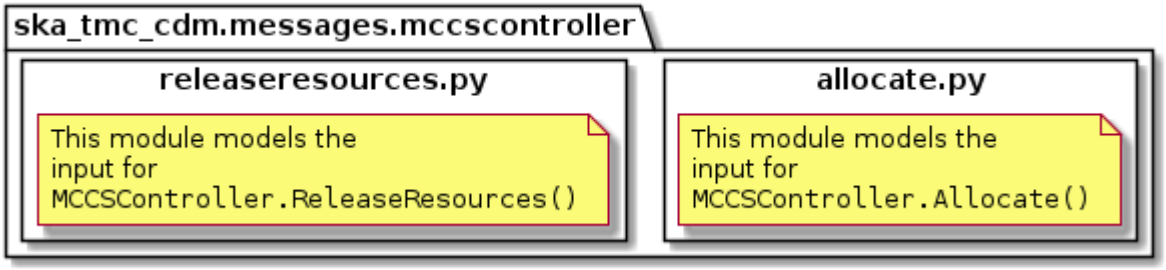
CHAPTER

ELEVEN

MCCSCONTROLLER

11.1 Overview

MCCS resource allocation is achieved via communication with an MCCSController device. The `mccscontroller` package models the JSON input and for MCCSController commands. The contents of this package are shown in the figure below.



Classes in the *allocate.py* module model the arguments for the `MCCSController.Allocate()` command.

Classes in the *releaseresources.py* module model the arguments for the `MCCSController.ReleaseResources()` command.

11.2 allocate.py

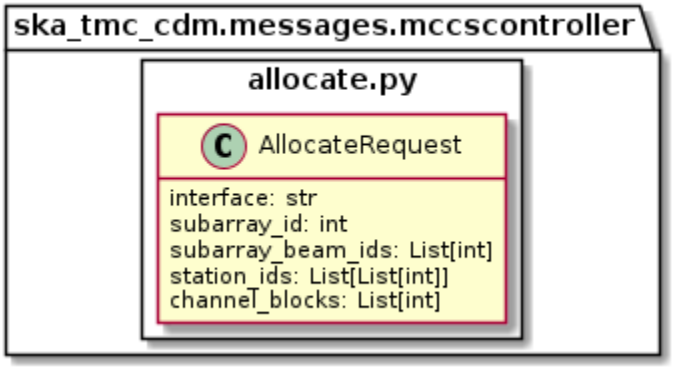


Fig. 1: allocate.py object model

The `allocate.py` module models the the JSON input for an `MCCSController.Allocate()` command.

Example JSON input modelled by `MCCSController.Allocate`:

```
{
  "interface": "https://schema.skao.int/ska-low-mccs-assignresources/2.0",
  "subarray_id": 1,
  "subarray_beam_ids": [1],
  "station_ids": [[1,2]],
  "channel_blocks": [3]
}
```

11.3 releaseresources.py

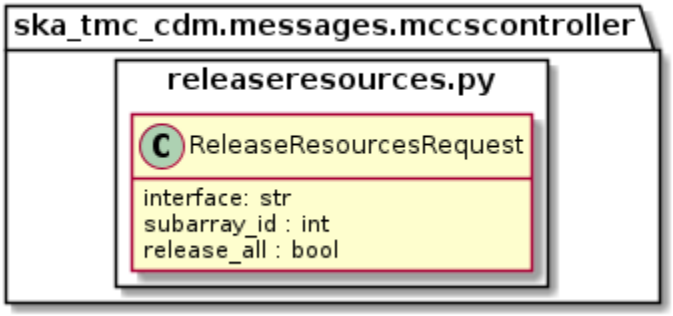


Fig. 2: releaseresources.py object model

The `releaseresources.py` module models the input JSON for a `MCCSController.ReleaseResources()` command.

Example ReleaseResourcesRequest JSON that requests all resources be released from sub-array #1:

```
{
  "interface": "https://schema.skao.int/ska-low-mccs-releaseresources/2.0",
  "subarray_id": 1,
  "release_all": true
}
```

CHAPTER

TWELVE

USING THE CDM

To use this library in your project, create objects using the classes defined in `ska_tmc_cdm.messages` and convert them to/from JSON using `ska_tmc_cdm.schemas.CODEC`.

The Python snippet below is an example of constructing a JSON argument for a `CentralNode.ReleaseResources()` command. The resulting JSON can be sent to the device using a `DeviceProxy`.

```
# import the classes for ReleaseResources commands and CODEC for serialisation
from ska_tmc_cdm.messages import ReleaseResourcesRequest
from ska_tmc_cdm.schemas import CODEC

# create an object for a command that will release all resources on subarray #2
cmd_arg = ReleaseResourcesRequest(2, release_all=True)
# convert the argument to JSON, ready for use in a DeviceProxy call
as_json = CODEC.dumps(cmd_arg)
```

Below is an example of converting the JSON response from a `CentralNode.AssignResources()` command to Python objects. The example assumes you have the string response from the command call at hand.

```
# import the classes for ReleaseResources commands and CODEC for serialisation
from ska_tmc_cdm.messages import AssignResourcesResponse
from ska_tmc_cdm.schemas import CODEC

# assume that you have some JSON-formatted string returned by AssignResources()
json_response = ...
# convert the JSON to a Python object. This requires you to provide the class
# you want to convert to
unmarshalled = CODEC.loads(AssignResourcesResponse, json_response)
# This object can hold other objects, as defined by the schema. For example,
# the response for an AssignResources command includes the dish IDs of the
# dishes that were assigned to it. The schema converts this into a
# DishAllocation object we can inspect and manipulate
print(f'Dish IDs allocated: {unmarshalled.dish.receptor_ids}')
```

CHAPTER

THIRTEEN

API

13.1 ska_tmc_cdm.jsonschema

13.1.1 ska_tmc_cdm.jsonschema.json_schema

The JSON Schema module contains methods for fetching version-specific JSON schemas using interface uri and validating the structure of JSON against these schemas.

class JsonSchema

JSON Schema use for validating the structure of JSON data

static get_schema_by_uri(*uri: str*) → ska_telmodel.schema.Schema

Retrieve JSON Schemas from remote server.

Parameters
uri – Interface Version URI

Returns
Interface schema

Raises
SchemaNotFound if URI does not resolve to a schema

static validate_schema(*uri: str*, *instance: dict*, *strictness=None*) → None

Validate an instance dictionary under the given schema.

strictness can be set from 0-2. Values equal:

0: permissive warnings 1: permissive errors and strict warnings 2: strict errors

Parameters

- **uri** – The schema to validate with
- **instance** – The instance to validate
- **strictness** – strictness level

Returns
None, in case of valid data otherwise, it raises an exception.

13.2 ska_tmc_cdm.messages

The ska_tmc_cdm.messages package contains modules that maps Tango structured arguments to Python.

13.2.1 ska_tmc_cdm.messages.central_node

The ska_tmc_cdm.messages.central_node package holds modules that translate TMC Central Node requests and responses to and from Python.

13.2.2 ska_tmc_cdm.messages.central_node.assign_resources

The messages module provides simple Python representations of the structured request and response for the TMC CentralNode.AssignResources command.

class AssignResourcesRequest(*subarray_id: Optional[int] = None*, *dish_allocation: Optional[DishAllocation] = None*, *sdp_config: Optional[SDPConfiguration] = None*, *mccs: Optional[MCCSAllocate] = None*, *interface: Optional[str] = None*, *transaction_id: Optional[str] = None*)

AssignResourcesRequest is a Python representation of the structured argument for a TMC CentralNode.AssignResourcesRequest request.

classmethod from_dish(*subarray_id: int*, *dish_allocation: DishAllocation*, *sdp_config: Optional[SDPConfiguration] = None*, *interface: Optional[str] = None*, *transaction_id: Optional[str] = None*)

Create a new AssignResourcesRequest object.

Parameters

- **subarray_id** – the numeric SubArray ID (1..16)
- **dish_allocation** – object holding the DISH resource allocation for this request.
- **sdp_config** – sdp configuration

Returns

AssignResourcesRequest object

classmethod from_mccs(*subarray_id: int*, *mccs: MCCSAllocate*, *sdp_config: Optional[SDPConfiguration] = None*, *interface: Optional[str] = None*, *transaction_id: Optional[str] = None*)

Create a new AssignResourcesRequest object.

Parameters

- **subarray_id** – the numeric SubArray ID (1..16)
- **mccs** – MCCS subarray allocation
- **sdp_config** – SDP configuration
- **interface** – url string to determine JsonSchema version

Returns

AssignResourcesRequest object

class AssignResourcesResponse(*dish_allocation: Optional[DishAllocation] = None*)

AssignResourcesResponse is a Python representation of the structured response from a TMC CentralNode.AssignResources request.

13.2.3 ska_tmc_cdm.messages.central_node.release_resources

The release_resources module provides simple Python representations of the structured request and response for a TMC CentralNode.ReleaseResources call.

class ReleaseResourcesRequest(**_*, *interface: Optional[str] = None*, *transaction_id: Optional[str] = None*, *subarray_id: Optional[int] = None*, *release_all: bool = False*, *dish_allocation: Optional[*DishAllocation*] = None*)

ReleaseResourcesRequest is a Python representation of the structured request for a TMC CentralNode.ReleaseResources call.

13.2.4 ska_tmc_cdm.messages.central_node.common

The messages module provides simple Python representations of the structured request and response for the TMC CentralNode.AssignResources command.

class DishAllocation(*receptor_ids: Optional[List[str]] = None*)

DishAllocation represents the DISH allocation part of an AssignResources request and response.

13.2.5 ska_tmc_cdm.messages.central_node.sdp

The messages module provides simple Python representations of the structured request and response for the TMC CentralNode.AssignResources command.

class BeamConfiguration(*beam_id: Optional[str] = None*, *function: Optional[str] = None*, *search_beam_id: Optional[int] = None*, *timing_beam_id: Optional[int] = None*, *vlbi_beam_id: Optional[int] = None*)

Class to hold Dependencies for Beam Configuration

class Channel(*count: int*, *start: int*, *stride: int*, *freq_min: float*, *freq_max: float*, *link_map: List[List]*, *spectral_window_id: Optional[str] = None*)

Class to hold Channels for ScanType

class ChannelConfiguration(*channels_id: Optional[str] = None*, *spectral_windows: Optional[List[*Channel*]] = None*)

Class to hold Dependencies for Channel Configuration

class EBScanType(*scan_type_id: Optional[str] = None*, *beams: Optional[Dict[str,* EBScanTypeBeam*]] = None*, *derive_from: Optional[str] = None*)

Class to hold EBScanType configuration

class EBScanTypeBeam(*field_id: Optional[str] = None*, *channels_id: Optional[str] = None*, *polarisations_id: Optional[str] = None*)

Class to hold EBScanTypeBeam Configuration

class ExecutionBlockConfiguration(*eb_id: Optional[str] = None*, *max_length: Optional[float] = None*, *context: Optional[Dict] = None*, *beams: Optional[List[*BeamConfiguration*]] = None*, *channels: Optional[List[*ChannelConfiguration*]] = None*, *polarisations: Optional[List[*PolarisationConfiguration*]] = None*, *fields: Optional[List[*FieldConfiguration*]] = None*, *scan_types: Optional[List[*EBScanType*]] = None*)

Class to hold ExecutionBlock configuration

class FieldConfiguration(*field_id: Optional[str] = None*, *pointing_fqdn: Optional[str] = None*, *phase_dir: Optional[PhaseDir] = None*)

Class to hold Field configuration

class PbDependency(*pb_id: str*, *kind: List[str]*)

Class to hold Dependencies for ProcessingBlock

class PhaseDir(*ra: Optional[List] = None*, *dec: Optional[List] = None*, *reference_time: Optional[str] = None*, *reference_frame: Optional[str] = None*)

Class to hold PhaseDir configuration

class PolarisationConfiguration(*polarisations_id: Optional[str] = None*, *corr_type: Optional[List[str]] = None*)

Class to hold Dependencies for Polarisation Configuration

class ProcessingBlockConfiguration(*pb_id: Optional[str] = None*, *workflow: Optional[SDPWorkflow] = None*, *parameters: Optional[Dict] = None*, *dependencies: Optional[List[PbDependency]] = None*, *sbi_ids: Optional[List] = None*, *script: Optional[ScriptConfiguration] = None*)

Class to hold ProcessingBlock configuration

class SDPConfiguration(*eb_id: Optional[str] = None*, *max_length: Optional[float] = None*, *scan_types: Optional[List[ScanType]] = None*, *processing_blocks: Optional[List[ProcessingBlockConfiguration]] = None*, *execution_block: Optional[ExecutionBlockConfiguration] = None*, *resources: Optional[Dict] = None*, *interface: Optional[str] = None*)

Class to hold SDP Configuration

class SDPWorkflow(*name: str*, *kind: str*, *version: str*)

Class to hold SDPWorkflows for ProcessingBlock

class ScanType(*scan_type_id*, *reference_frame: str*, *ra: str*, *dec: str*, *channels: List[Channel]*)

Class to hold ScanType configuration

class ScriptConfiguration(*kind: Optional[str] = None*, *name: Optional[str] = None*, *version: Optional[str] = None*)

Class to hold ScriptConfiguration

13.2.6 ska_tmc_cdm.messages.central_node.mccs

class MCCSAllocate(*station_ids: Sequence[Sequence[int]]*, *channel_blocks: Sequence[int]*, *subarray_beam_ids: Sequence[int]*)

MCCSAllocate is a Python representation of the structured argument for a TMC CentralNode.AssignResourcesRequest.

13.2.7 ska_tmc_cdm.messages.mccscontroller.allocate

The allocate module defines a Python object model for the structured JSON given in an MCCSController.Allocate call.

class AllocateRequest(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-mccs-assignresources/2.0'*, *subarray_id: int*, *subarray_beam_ids: Optional[List[int]] = None*, *station_ids: Optional[List[List[int]]] = None*, *channel_blocks: Optional[List[int]] = None*)

AssignResourcesRequest is the object representation of the JSON argument for an MCCSController.Allocate command.

13.2.8 ska_tmc_cdm.messages.mccscontroller.releaseresources

The allocate module defines a Python object model for the structured JSON that forms the argument for an MCCSController.ReleaseResources call.

class ReleaseResourcesRequest(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-mccs-releaseresources/2.0'*, *subarray_id: int*, *release_all: bool*)

ReleaseResourcesRequest is the object representation of the JSON argument for an MCCSController.ReleaseResources command.

13.2.9 ska_tmc_cdm.messages.mccssubarray.assigned_resources

class AssignedResources(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-mccs-assignedresources/2.0'*, *subarray_beam_ids: Optional[List[int]] = None*, *station_ids: Optional[List[List[int]]] = None*, *channel_blocks: Optional[List[int]] = None*)

AssignedResources is the object representation of the JSON returned by the MCCSSubarray.assigned_resources attribute.

13.2.10 ska_tmc_cdm.messages.mccssubarray.configure

The mccssubarray.configure module contains a Python object model for the various structured bits of JSON given in an MCCSSubarray.Configure call.

class ConfigureRequest(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-mccs-configure/2.0'*, *stations: List[*StationConfiguration*]*, *subarray_beams: List[*SubarrayBeamConfiguration*]*)

Class to hold all subarray configuration.

class StationConfiguration(*station_id: int*)

A class to hold station configuration

class SubarrayBeamConfiguration(*, *subarray_beam_id: int*, *station_ids: List[int]*, *update_rate: float*, *channels: List[List[int]]*, *sky_coordinates: List[float]*, *antenna_weights: List[float]*, *phase_centre: List[float]*)

A class to hold subarray beam configuration attributes

13.2.11 ska_tmc_cdm.messages.mccssubarray.scan

The scan module defines Python object representations of the structured request for an MCCSSubarray.Scan command.

class ScanRequest(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-mccs-scan/2.0'*, *scan_id: int*, *start_time: float*)

ScanRequest represents the request argument for MCCSSubarray.Scan call.

13.2.12 ska_tmc_cdm.messages.subarray_node

The ska_tmc_cdm.messages.subarray_node package holds modules containing classes that represent arguments, requests, and responses for TMC SubArrayNode devices.

13.2.13 ska_tmc_cdm.messages.subarray_node.assigned_resources

TMC Assigned Resources

class AssignedResources(*, *interface: Optional[str] = 'https://schema.skao.int/ska-low-tmc-assignedresources/2.0'*, *mccs: MCCSAllocation*)

AssignedResources models the structured JSON returned when the MCCSSubarray.assigned_resources Tango attribute is read.

is_empty() → bool

Determine that the current MCCSAllocation instance is empty (none of the attribute Lists are populated)

class MCCSAllocation(*subarray_beam_ids: List[int]*, *station_ids: List[List[int]]*, *channel_blocks: List[int]*)

MCCSAllocation is a Python representation of the structured JSON representing the resources assigned to an MCCS subarray.

is_empty()

Determine that the current MCCSAllocation instance is empty (none of the attribute Lists are populated)

13.2.14 ska_tmc_cdm.messages.subarray_node.configure

The configure package contains modules that define Python classes for all of the permissible arguments for a SubArrayNode.configure() call.

class ConfigureRequest(*pointing: Optional[PointingConfiguration] = None*, *dish: Optional[DishConfiguration] = None*, *sdp: Optional[SDPConfiguration] = None*, *csp: Optional[CSPConfiguration] = None*, *mccs: Optional[MCCSConfiguration] = None*, *tmc: Optional[TMCConfiguration] = None*, *interface: Optional[str] = 'https://schema.skao.int/ska-tmc-configure/2.1'*, *transaction_id: Optional[str] = None*)

ConfigureRequest encapsulates the arguments required for the TMC SubArrayNode.Configure() command.

13.2.15 ska_tmc_cdm.messages.subarray_node.configure.core

The configure.common module contains simple Python representations of the structured request and response for the TMC SubArrayNode.Configure command.

As configurations become more complex, they may be rehomed in a submodule of this package.

class DishConfiguration(*receiver_band:* ReceiverBand)

DishConfiguration specifies how SKA MID dishes in a sub-array should be configured. At the moment, this is limited to setting the receiver band.

class PointingConfiguration(*target:* Target)

PointingConfiguration specifies where the subarray receptors are going to point.

class ReceiverBand(*value*)

ReceiverBand is an enumeration of SKA MID receiver bands.

class Target(*ra*, *dec*, *target_name=''*, *reference_frame='icrs'*, *unit=('hourangle', 'deg')*)

Target encapsulates source coordinates and source metadata.

The SubArrayNode ICD specifies that RA and Dec must be provided, hence non-ra/dec frames such as galactic are not supported.

13.2.16 ska_tmc_cdm.messages.subarray_node.configure.csp

The configure.csp module contains Python classes that represent the various aspects of CSP configuration that may be specified in a SubArrayNode.configure command.

class BeamConfiguration(*pst_beam_id: Optional[int] = None*, *stn_beam_id: Optional[int] = None*, *offset_dly_poly: Optional[str] = None*, *stn_weights: Optional[List[float]] = None*, *jones: Optional[str] = None*, *dest_chans: Optional[List[int]] = None*, *rfi_enable: Optional[List[bool]] = None*, *rfi_static_chans: Optional[List[int]] = None*, *rfi_dynamic_chans: Optional[List[int]] = None*, *rfi_weighted: Optional[float] = None*)

Class to hold Beams Configuration.

class CBFConfiguration(*fsp_configs: List[*FSPConfiguration*]*, *vlbi_config: Optional[dict] = None*)

Class to hold all FSP and VLBI configurations.

class CSPConfiguration(*interface: Optional[str] = None*, *subarray: Optional[*SubarrayConfiguration*] = None*, *common: Optional[*CommonConfiguration*] = None*, *cbf_config: Optional[*CBFConfiguration*] = None*, *pst_config: Optional[dict] = None*, *pss_config: Optional[dict] = None*, *lowcbf: Optional[*LowCBFConfiguration*] = None*)

Class to hold all CSP configuration.

class CommonConfiguration(*config_id: str*, *frequency_band: Optional[*ReceiverBand*] = None*, *subarray_id: Optional[int] = None*, *band_5_tuning: Optional[List[float]] = None*)

Class to hold the CSP sub-elements.

class FSPConfiguration(*fsp_id: int*, *function_mode:* FSPFunctionMode, *frequency_slice_id: int*, *integration_factor: int*, *zoom_factor: int*, *channel_averaging_map: Optional[List[Tuple]] = None*, *output_link_map: Optional[List[Tuple]] = None*, *channel_offset: Optional[int] = None*, *zoom_window_tuning: Optional[int] = None*)

FSPConfiguration defines the configuration for a CSP Frequency Slice Processor.

class FSPFunctionMode(*value*)

FSPFunctionMode is an enumeration of the available FSP modes.

class LowCBFConfiguration(*stations: Optional[*StationConfiguration*] = None*, *timing_beams: Optional[*TimingBeamConfiguration*] = None*)

Class to hold Low CBF Configuration.

class StationConfiguration(*stns: Optional[List[List[int]]] = None*, *stn_beams: Optional[List[*StnBeamConfiguration*]] = None*)

Class to hold Stations Configuration.

class StnBeamConfiguration(*beam_id: Optional[int] = None*, *freq_ids: Optional[List[int]] = None*, *boresight_dly_poly: Optional[str] = None*)

Class to hold Stations Beam Configuration.

class SubarrayConfiguration(*subarray_name: str*)

Class to hold the parameters relevant only for the current sub-array device.

class TimingBeamConfiguration(*beams: Optional[List[*BeamConfiguration*]] = None*)

Class to hold Timing Beams Configuration.

13.2.17 ska_tmc_cdm.messages.subarray_node.configure.sdp

The configure.sdp module contains Python classes that represent the various aspects of SDP configuration that may be specified in a SubArrayNode.configure command.

class SDPConfiguration(**, interface: Optional[str] = 'https://schema.skao.int/ska-sdp-configure/0.3'*, *scan_type: str*)

Message class to hold SDP configuration aspect of a TMC SubArrayNode.Configure call.

13.2.18 ska_tmc_cdm.messages.subarray_node.configure.mccs

The configure.mccs module contains Python classes that represent the various aspects of MCCS configuration that may be specified in a SubArray.configure command.

class MCCSConfiguration(**_, station_configs: List[*StnConfiguration*]*, *subarray_beam_configs: List[*SubarrayBeamConfiguration*]*)

Class to hold all subarray configuration.

class StnConfiguration(*station_id: int*)

A class to hold station configuration configuration

class SubarrayBeamConfiguration(*subarray_beam_id: int*, *station_ids: List[int]*, *channels: List[List[int]]*, *update_rate: float*, *target:* SubarrayBeamTarget, *antenna_weights: List[float]*, *phase_centre: List[float]*)

A class to hold subarray_beam configuration attributes

class SubarrayBeamTarget(*az: float*, *el: float*, *target_name: str*, *reference_frame: str*)

Target encapsulates source coordinates and source metadata.

The SubArrayNode ICD specifies that az and el must be provided

13.2.19 ska_tmc_cdm.messages.subarray_node.configure.tmc

Configuration specific to TMC. scan_duration (in seconds) is the duration to be used for all scan commands following this configuration.

class TMCConfiguration(*scan_duration: timedelta*)

Class to hold TMC configuration

13.2.20 ska_tmc_cdm.messages.subarray_node.scan

The scan module defines simple Python representations of the structured request for a TMC SubArrayNode.Scan command.

class ScanRequest(*, *interface: Optional[str] = 'https://schema.skao.int/ska-tmc-scan/2.1'*, *transaction_id: Optional[str] = None*, *scan_id: int*)

ScanRequest represents the JSON for a SubArrayNode.scan call.

13.3 ska_tmc_cdm.schemas

The schemas for the SKA Configuration Data Model (CDM).

13.3.1 ska_tmc_cdm.schemas.central_node

The schemas.central_node package contains Marshmallow schemas that convert JSON to/from the Python classes contained in ska_tmc_cdm.messages.central_node.

class AssignResourcesRequestSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the AssignResourcesRequest class.

class Meta

marshmallow directives for AssignResourcesRequestSchema.

create_request(*data*, ***_*)

Convert parsed JSON back into an AssignResources request object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns

AssignResources object populated from data

dish

alias of *DishAllocationSchema*

mccs

alias of *MCCSAllocateSchema*

sdp_config

alias of *SDPConfigurationSchema*

validate_on_dump(*data*, **_)

Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

class AssignResourcesResponseSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the AssignResourcesResponse class.

class Meta

Marshmallow directives for AssignResourcesResponseSchema.

create_response(*data*, **_)

Convert parsed JSON from an AssignResources response back into an AssignResourcesResponse object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
AssignResourcesResponse object populated from data

dish

alias of *DishAllocationResponseSchema*

class DishAllocationResponseSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the DishAllocation class when received in the response.

create(*data*, **_)

Convert parsed JSON from an AssignResources response back into a DishAllocation object.

This 'duplicate' schema is required as the DishAllocation is found under a different JSON key in the response as compared to the request.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishAllocation object populated from data

class DishAllocationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the DishAllocation class.

create(*data*, **_)

Convert parsed JSON back into a DishAllocation object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishAllocation object populated from data

class MCCSAllocateSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the MCCSAllocate class.

create_mccs_allocate(*data*, **_)

Convert parsed JSON back into a MCCSAllocate object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
MCCSAllocate object populated from data

class ReleaseResourcesRequestSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the ReleaseResourcesRequest class.

class Meta

Marshmallow directives for ReleaseResourcesRequestSchema.

create_request(*data*, **_)

Convert parsed JSON from an ReleaseResources request back into an ReleaseResourcesRequest object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ReleaseResourcesRequest object populated from data

dish

alias of *DishAllocationSchema*

filter_args(*data*, **_)

Filter Marshmallow's JSON based on the value of release_all.

If release_all is True, other resource definitions should be stripped from the request. If release_all for MID set to False, the 'release_all' key itself should be stripped. If release_all_low for LOW set to False, the 'release_all_low' key itself should be stripped.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for request submission

class SDPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the SDPConfiguration class

create_sdp_config(*data*, **_)

Convert parsed JSON back into a SDPConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values

- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

execution_block
alias of *ExecutionBlockConfigurationSchema*

filter_nulls(*data*, **_)
Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

processing_blocks
alias of *ProcessingBlockSchema*

scan_types
alias of *ScanTypeSchema*

13.3.2 ska_tmc_cdm.schemas.central_node.assign_resources

The schemas.central_node module defines Marshmallow schemas that map TMC Central Node message classes to/from a JSON representation.

class AssignResourcesRequestSchema(**args: Any*, ***kwargs: Any*)
Marshmallow schema for the AssignResourcesRequest class.

class Meta
marshmallow directives for AssignResourcesRequestSchema.

create_request(*data*, **_)
Convert parsed JSON back into an AssignResources request object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
AssignResources object populated from data

dish
alias of *DishAllocationSchema*

mccs
alias of *MCCSAllocateSchema*

sdp_config
alias of *SDPConfigurationSchema*

validate_on_dump(*data*, **_)

Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

class AssignResourcesResponseSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the AssignResourcesResponse class.

class Meta

Marshmallow directives for AssignResourcesResponseSchema.

create_response(*data*, **_)

Convert parsed JSON from an AssignResources response back into an AssignResourcesResponse object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
AssignResourcesResponse object populated from data

dish

alias of *DishAllocationResponseSchema*

13.3.3 ska_tmc_cdm.schemas.central_node.common

The schemas.central_node module defines Marshmallow schemas that map TMC Central Node message classes to/from a JSON representation.

class DishAllocationResponseSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the DishAllocation class when received in the response.

create(*data*, **_)

Convert parsed JSON from an AssignResources response back into a DishAllocation object.

This 'duplicate' schema is required as the DishAllocation is found under a different JSON key in the response as compared to the request.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishAllocation object populated from data

class DishAllocationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the DishAllocation class.

create(*data*, **_)

Convert parsed JSON back into a DishAllocation object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishAllocation object populated from data

13.3.4 ska_tmc_cdm.schemas.central_node.sdp

The schemas.central_node module defines Marshmallow schemas that map TMC Central Node message classes to/from a JSON representation.

class BeamConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the BeamConfiguration class

create_beam_config(*data*, **_)

Convert parsed JSON back into a BeamConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class ChannelConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the ChannelConfiguration class

create_channel_config(*data*, **_)

Convert parsed JSON back into a ChannelConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

spectral_windows

alias of *ChannelSchema*

class ChannelSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the SubBand class.

create_channel(*data*, **_)

Convert parsed JSON back into a Channel object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SubBand object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class EBScanTypeBeamSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the EBScanTypeBeam class

create_ebscantypebeams_config(*data*, **_)

Convert parsed JSON back into a EBScanTypeBeam object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class **EBScanTypeSchema**(**args: Any*, ***kwargs: Any*)

Marsmallow class for the EBScanTypeBeam class

create_ebscantype_config(*data*, **_)

Convert parsed JSON back into a EBScanType object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class **ExecutionBlockConfigurationSchema**(**args: Any*, ***kwargs: Any*)

Marsmallow class for the ExecutionBlockConfiguration class

create_executionblock_config(*data*, **_)

Convert parsed JSON back into a ExecutionBlockConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class **FieldConfigurationSchema**(**args: Any*, ***kwargs: Any*)

Marsmallow class for the FieldConfiguration class

create_polarisation_config(*data*, **_)

Convert parsed JSON back into a FieldConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

phase_dir

alias of *PhaseDirSchema*

class PbDependencySchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the PbDepedency class.

create_pb_dependency(*data*, **_)

Convert parsed JSON back into a PbDependency object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
PbDependency object populated from data

class PhaseDirSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the PhaseDir class

create_phase_dir_config(*data*, **_)

Convert parsed JSON back into a PhaseDir object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class PolarisationConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the PolarisationConfiguration class

create_polarisation_config(*data*, ***_*)

Convert parsed JSON back into a PolarisationConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

filter_nulls(*data*, ***_*)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

class ProcessingBlockSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the ProcessingBlock class.

create_processing_block_config(*data*, ***_*)

Convert parsed JSON back into a ProcessingBlock object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
PB object populated from data

filter_nulls(*data*, ***_*)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

script

alias of *ScriptConfigurationSchema*

workflow

alias of *SDPWorkflowSchema*

class SDPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marsmallow class for the SDPConfiguration class

create_sdp_config(*data*, **_)

Convert parsed JSON back into a SDPConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

execution_block

alias of *ExecutionBlockConfigurationSchema*

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

processing_blocks

alias of *ProcessingBlockSchema*

scan_types

alias of *ScanTypeSchema*

class SDPWorkflowSchema(**args: Any*, ***kwargs: Any*)

Represents the type of workflow being configured on the SDP

create_sdp_wf(*data*, **_)

Convert parsed JSON back into a SDP Workflow object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDP Workflow object populated from data

class ScanTypeSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the ScanType class.

channels

alias of *ChannelSchema*

create_scan_type(*data*, **_)

Convert parsed JSON back into a ScanType object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ScanType object populated from data

`filter_nulls`(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

`class ScriptConfigurationSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the ScriptConfiguration class.

`create_executionblock_config`(*data*, **_)

Convert parsed JSON back into a ScriptConfiguration object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SDPConfiguration object populated from data

`filter_nulls`(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for PB configuration

13.3.5 ska_tmc_cdm.schemas.central_node.mccs

The schemas.central_node module defines Marshmallow schemas that map TMC Central Node message classes to/from a JSON representation.

`class MCCSAllocateSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the MCCSAllocate class.

`create_mccs_allocate`(*data*, **_)

Convert parsed JSON back into a MCCSAllocate object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
MCCSAllocate object populated from data

13.3.6 ska_tmc_cdm.schemas.codec

The codec module contains classes used by clients to marshall CDM classes to and from JSON. This saves the clients having to instantiate and manipulate the Marshmallow schema directly.

class MarshmallowCodec

MarshmallowCodec marshalls and unmarshalls CDM classes.

The mapping of CDM classes to Marshmallow schema is defined in this class.

dumps(*obj*, *validate: bool = True*, *strictness: Optional[int] = None*)

Return a string JSON representation of a CDM instance.

The default strictness of the Telescope Model schema validator can be overridden by supplying the validate argument.

Parameters

- **obj** – the instance to marshall to JSON
- **validate** – True to enable schema validation
- **strictness** – optional validation strictness level (0=min, 2=max)

Returns
JSON representation of obj

load_from_file(*cls*, *path*, *validate: bool = True*, *strictness: Optional[int] = None*)

Load an instance of a CDM class from disk.

Parameters

- **cls** – the class to create from the file
- **path** – the path to the file
- **validate** – True to enable schema validation
- **strictness** – optional validation strictness level (0=min, 2=max)

Returns
an instance of cls

loads(*cdm_class*, *json_data*, *validate: bool = True*, *strictness: Optional[int] = None*)

Create an instance of a CDM class from a JSON string.

The default strictness of the Telescope Model schema validator can be overridden by supplying the validate argument.

Parameters

- **cdm_class** – the class to create from the JSON
- **json_data** – the JSON to unmarshall
- **validate** – True to enable schema validation
- **strictness** – optional validation strictness level (0=min, 2=max)

Returns
an instance of cls

register_mapping(*cdm_class*)

A decorator that is used to register the mapping between a Marshmallow schema and the CDM class it serialises.

Parameters
cdm_class – the CDM class this schema maps to

Returns
the decorator

`set_schema`(*cdm_class*, *schema_class*)

Set the schema for a CDM class.

Parameters

- **schema_class** – Marshmallow schema to map
- **cdm_class** – CDM class the schema maps to

13.3.7 ska_tmc_cdm.schemas.mccscontroller.allocate

The schemas.central_node module defines Marshmallow schemas that map MCCSController AllocateRequest message classes to/from their JSON representation.

`class AllocateRequestSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the MCCSController AllocateRequest class.

`create_allocaterequest`(*data*, **_) → *AllocateRequest*

Convert parsed JSON back into an AllocateRequest object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
AllocateRequest object populated from data

13.3.8 ska_tmc_cdm.schemas.mccscontroller.releaseresources

The releaseresources module defines Marshmallow schemas that map MCCSController ReleaseResourcesRequest objects to/from their JSON representation.

`class ReleaseResourcesRequestSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the ReleaseResourcesRequest class.

`create_request`(*data*, **_)

Convert parsed JSON from an ReleaseResources request back into an ReleaseResourcesRequest object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ReleaseResourcesRequest object populated from data

13.3.9 ska_tmc_cdm.schemas.mccssubarray.assigned_resources

The assigned_resources module defines Marshmallow schemas that maps the MCCSSubarray.assigned_resources attribute to/from a JSON representation.

class AssignedResourcesSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the MCCSSubarray AssignedResources class.

create_allocaterequest(*data*, **_) → *AssignedResources*

Convert parsed JSON back into an AssignedResources object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
AssignedResources object populated from data

13.3.10 ska_tmc_cdm.schemas.mccssubarray.configure

The configure module defines Marshmallow schemas that maps the MCCSSubarray.Configure call arguments to/from a JSON representation.

class ConfigureRequestSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the mccssubarray.ConfigureRequest class

create(*data*, **_) → *ConfigureRequest*

Convert parsed JSON back into a ConfigureRequest object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ConfigureRequest instance populated to match JSON

stations

alias of *StationConfigurationSchema*

subarray_beams

alias of SubarrayBeamConfigurationSchema

class StationConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_) → *StationConfiguration*

Convert parsed JSON back into a StationConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
StnConfiguration instance populated to match JSON

class SubarrayBeamConfiguration(*, *subarray_beam_id: int*, *station_ids: List[int]*, *update_rate: float*, *channels: List[List[int]]*, *sky_coordinates: List[float]*, *antenna_weights: List[float]*, *phase_centre: List[float]*)

A class to hold subarray beam configuration attributes

13.3.11 ska_tmc_cdm.schemas.mccssubarray.scan

The schemas module defines Marshmallow schemas that map CDM message classes and data model classes to/from a JSON representation.

class ScanRequestSchema(**args: Any*, ***kwargs: Any*)

Create the Schema for ScanRequest

create_scanrequest(*data*, ***_*)

Convert parsed JSON back into a ScanRequest

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns

ScanRequest instance populated to match JSON

13.3.12 ska_tmc_cdm.schemas.shared

The schemas module defines Marshmallow schemas that are shared by various other serialisation schemas.

class OrderedSchema(**args: Any*, ***kwargs: Any*)

Subclass of Schema, anything inheriting from Schema has the order of its JSON properties respected in the message. Saves adding a Meta class to everything individually

class Meta

marshmallow directive to respect order of JSON properties in message.

class UpperCasedField(**args: Any*, ***kwargs: Any*)

Field that serializes to an upper-case string and deserializes to a lower-case string.

class ValidatingSchema(**args: Any*, ***kwargs: Any*)

ValidatingSchema is a marshmallow schema that calls the appropriate Telescope Model schema validation functions when serialising or deserialising JSON.

validate_json(*data*, *process_fn*)

Validate JSON using the Telescope Model schema.

The process_fn argument can be used to process semantically correct but schematically invalid Python to something equivalent but valid, e.g., to convert a list of Python tuples to a list of lists.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **process_fn** – data processing function called before validation

Returns

validate_on_dump(*data*, *process_fn=<function ValidatingSchema.<lambda>>*, ***_*)

Validate the serialised object against the relevant Telescope Model schema.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **process_fn** – function to process data before validation
- **_** – unused kwargs passed by Marshmallow

Returns

dict suitable for writing as a JSON string

validate_on_load(*data*, *process_fn=<function ValidatingSchema.<lambda>>*, ***_*)

Validate the JSON string to deserialise.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **process_fn** – function to process data before validation
- **_** – unused kwargs passed by Marshmallow

Returns

dict suitable for object constructor.

13.3.13 ska_tmc_cdm.schemas.subarray_node

The schemas.subarray_node package contains Marshmallow schemas that convert JSON to/from the Python classes contained in ska_tmc_cdm.messages.subarray_node.

13.3.14 ska_tmc_cdm.schemas.subarray_node.assigned_resources

This module defines Marshmallow schemas that map CDM classes to/from JSON.

class AssignedResourcesSchema(**args: Any*, ***kwargs: Any*)

AssignedResourcesSchema maps the AssignedResources class to/from a JSON representation.

create_assigned_resources(*data*, ***_*)

Convert parsed JSON back into an AssignedResources object

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns

AssignedResources object populated from data

mccs

alias of *MCCSAllocationSchema*

class MCCSAllocationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the MCCSAllocation class.

create_mccs_allocation(*data*, **_)

Convert parsed JSON back into a MCCSAllocation object.

Parameters

- **data** – Marshmallow-provided dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
MCCSAllocation object populated from data

13.3.15 ska_tmc_cdm.schemas.subarray_node.configure

The schemas module defines Marshmallow schemas that map CDM message classes and data model classes to/from a JSON representation.

class CBFConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_)

Convert parsed JSON back into a CBFConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
CBFConfiguration instance populated to match JSON

Return type
CBFConfiguration

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for CBF configuration

fsp_configs

alias of *FSPConfigurationSchema*

class CSPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.CSPConfiguration class

cbf_config

alias of *CBFConfigurationSchema*

common

alias of *CommonConfigurationSchema*

create(*data*, **_)

Convert parsed JSON back into a CSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
CSPConfiguration instance populated to match JSON

lowcbf

alias of *LowCBFConfigurationSchema*

subarray

alias of *SubarrayConfigurationSchema*

validate_on_dump(*data*, **_)

Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

class CommonConfigurationSchema(**args: Any*, ***kwargs: Any*)

convert(*common_configuration:* CommonConfiguration, **_)

Process CommonConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **CommonConfiguration** – Common configuration to process
- **_** – kwargs passed by Marshmallow

Returns
CommonConfiguration instance populated to match JSON

create(*data*, **_)

Convert parsed JSON back into a CSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
CommonConfiguration instance populated to match JSON

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values

- **_** – kwargs passed by Marshmallow

Returns
dict suitable for FSP configuration

class **ConfigureRequestSchema**(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.ConfigureRequest class.

create_configuration(*data*, **_)

Converted parsed JSON backn into a subarray_node.ConfigureRequest object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ConfigurationRequest instance populated to match JSON

csp

alias of *CSPConfigurationSchema*

dish

alias of *DishConfigurationSchema*

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

mccs

alias of *MCCSConfigurationSchema*

pointing

alias of *PointingSchema*

sdp

alias of *SDPConfigurationSchema*

tmc

alias of *TMCConfigurationSchema*

class **DishConfigurationSchema**(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.DishConfiguration class.

convert(*dish_configuration:* DishConfiguration, **_)

Process DishConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **dish_configuration** – the dish configuration
- **_** – kwargs passed by Marshmallow

Returns
DishConfiguration instance populated to match JSON

create_dish_configuration(*data*, **_)

Converted parsed JSON back into a subarray_node.DishConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishConfiguration instance populated to match JSON

class FSPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.FSPConfiguration class

convert(*fsp_configuration:* FSPConfiguration, **_)

Process FSPConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **fsp_configuration** – FSP configuration to process
- **_** – kwargs passed by Marshmallow

Returns
FspConfiguration instance populated to match JSON

create(*data*, **_)

Convert parsed JSON back into a FSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
FSPConfiguration instance populated to match JSON

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for FSP configuration

class LowCBFConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.LowCBFConfiguration class

create(*data*, **_)

Convert parsed JSON back into a LowCBFConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
LowCBFConfiguration instance populated to match JSON

`stations`
alias of `StationConfigurationSchema`

`timing_beams`
alias of `TimingBeamConfigurationSchema`

`validate_on_dump`(*data*, **_)
Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters
- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

`class MCCSConfigurationSchema`(**args: Any*, ***kwargs: Any*)
Marshmallow schema for the subarray_node.MCCSConfiguration class

`create`(*data*, **_)

Convert parsed JSON back into a MCCSConfiguration object.

Parameters
- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
MCCSConfiguration instance populated to match JSON

Return type
MCCSConfiguration

`filter_nulls_and_validate_schema`(*data*, **_)
validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters
- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

`station_configs`
alias of *StnConfigurationSchema*

`subarray_beam_configs`
alias of *SubarrayBeamConfigurationSchema*

`validate_json`(*data*, *process_fn=<function MCCSConfigurationSchema.<lambda>>*)
validating the structure of JSON against schemas

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **function** (*lambda*) – use for converting list of tuples to list of list

Returns

validate_schema(*data*, **_)

validating the structure of JSON against schemas

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for CSP configuration

class PointingSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.Pointing class.

create(*data*, **_)

Convert parsed JSON back into a subarray_node.Pointing object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
Pointing instance populated to match JSON

target

alias of *TargetSchema*

class SDPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow class for the SDPConfiguration class

create_sdp_configuration(*data*, **_)

Convert parsed JSON back into a set containing all the scans :param data: dict containing parsed JSON values :param _: kwargs passed by Marshmallow :return: SDPConfiguration instance populated to match JSON

class StnConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_)

Convert parsed JSON back into a StnConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
StnConfiguration instance populated to match JSON

Return type
StnConfiguration

class SubarrayBeamConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_) → *SubarrayBeamConfiguration*

Convert parsed JSON back into a SubarrayBeamConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SubarrayBeamConfiguration instance populated to match JSON

target

alias of *SubarrayBeamTargetSchema*

class SubarrayConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_)

Convert parsed JSON back into a SubarrayConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SubarrayConfiguration instance populated to match JSON

Return type
SubarrayConfiguration

class TargetSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.Target class

convert_to_icrs(*target:* Target, **_)

Process Target co-ordinates by converting them to ICRS frame before the JSON marshalling process begins.

Parameters

- **target** – Target instance to process
- **_** – kwargs passed by Marshallow

Returns
SexagesimalTarget with ICRS ra/dec expressed in hms/dms

create_target(*data*, **_)

Convert parsed JSON back into a Target object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
Target instance populated to match JSON

13.3.16 ska_tmc_cdm.schemas.subarray_node.configure.core

The schemas module defines Marshmallow schemas that map shared CDM message classes for SubArrayNode configuration to/from a JSON representation.

class ConfigureRequestSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.ConfigureRequest class.

create_configuration(*data*, **_)

Converted parsed JSON backn into a subarray_node.ConfigureRequest object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ConfigurationRequest instance populated to match JSON

csp

alias of *CSPConfigurationSchema*

dish

alias of *DishConfigurationSchema*

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

mccs

alias of *MCCSConfigurationSchema*

pointing

alias of *PointingSchema*

sdp

alias of *SDPConfigurationSchema*

tmc

alias of *TMCConfigurationSchema*

class DishConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.DishConfiguration class.

convert(*dish_configuration:* DishConfiguration, **_)

Process DishConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **dish_configuration** – the dish configuration
- **_** – kwargs passed by Marshmallow

Returns
DishConfiguration instance populated to match JSON

`create_dish_configuration`(*data*, **_)

Converted parsed JSON back into a subarray_node.DishConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
DishConfiguration instance populated to match JSON

`class PointingSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.Pointing class.

`create`(*data*, **_)

Convert parsed JSON back into a subarray_node.Pointing object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
Pointing instance populated to match JSON

`target`

alias of *TargetSchema*

`class TargetSchema`(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.Target class

`convert_to_icrs`(*target:* Target, **_)

Process Target co-ordinates by converting them to ICRS frame before the JSON marshalling process begins.

Parameters

- **target** – Target instance to process
- **_** – kwargs passed by Marshallow

Returns
SexagesimalTarget with ICRS ra/dec expressed in hms/dms

`create_target`(*data*, **_)

Convert parsed JSON back into a Target object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
Target instance populated to match JSON

13.3.17 ska_tmc_cdm.schemas.subarray_node.configure.csp

This module defines Marshmallow schemas that map the CDM classes for SubArrayNode CSP configuration to/from JSON.

class CBFConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_)

Convert parsed JSON back into a CBFConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
CBFConfiguration instance populated to match JSON

Return type
CBFConfiguration

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for CBF configuration

fsp_configs

alias of *FSPConfigurationSchema*

class CSPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.CSPConfiguration class

cbf_config

alias of *CBFConfigurationSchema*

common

alias of *CommonConfigurationSchema*

create(*data*, **_)

Convert parsed JSON back into a CSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
CSPConfiguration instance populated to match JSON

lowcbf

alias of *LowCBFConfigurationSchema*

subarray

alias of *SubarrayConfigurationSchema*

validate_on_dump(*data*, **_)

Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns

dict suitable for SubArrayNode configuration

class CommonConfigurationSchema(**args: Any*, ***kwargs: Any*)

convert(*common_configuration:* CommonConfiguration, **_)

Process CommonConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **CommonConfiguration** – Common configuration to process
- **_** – kwargs passed by Marshmallow

Returns

CommonConfiguration instance populated to match JSON

create(*data*, **_)

Convert parsed JSON back into a CSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns

CommonConfiguration instance populated to match JSON

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns

dict suitable for FSP configuration

class FSPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.FSPConfiguration class

convert(*fsp_configuration:* FSPConfiguration, **_)

Process FSPConfiguration instance so that it is ready for conversion to JSON.

Parameters

- **fsp_configuration** – FSP configuration to process
- **_** – kwargs passed by Marshmallow

Returns
FspConfiguration instance populated to match JSON

create(*data*, **_)

Convert parsed JSON back into a FSPConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
FSPConfiguration instance populated to match JSON

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for FSP configuration

class LowCBFConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.LowCBFConfiguration class

create(*data*, **_)

Convert parsed JSON back into a LowCBFConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
LowCBFConfiguration instance populated to match JSON

stations

alias of `StationConfigurationSchema`

timing_beams

alias of `TimingBeamConfigurationSchema`

validate_on_dump(*data*, **_)

Validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

class SubarrayConfigurationSchema(**args: Any*, ***kwargs: Any*)

create(*data*, **_)

Convert parsed JSON back into a SubarrayConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SubarrayConfiguration instance populated to match JSON

Return type
SubarrayConfiguration

13.3.18 ska_tmc_cdm.schemas.subarray_node.configure.sdp

This module defines Marshmallow schemas that map the SDPConfiguration message classes to/from JSON.

class SDPConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow class for the SDPConfiguration class

create_sdp_configuration(*data*, **_)

Convert parsed JSON back into a set containing all the scans :param data: dict containing parsed JSON values :param _: kwargs passed by Marshmallow :return: SDPConfiguration instance populated to match JSON

13.3.19 ska_tmc_cdm.schemas.subarray_node.configure.mccs

This module defines Marshmallow schemas that map the CDM classes for SubArrayNode MCCS configuration to/from JSON.

class MCCSConfigurationSchema(**args: Any*, ***kwargs: Any*)

Marshmallow schema for the subarray_node.MCCSConfiguration class

create(*data*, **_)

Convert parsed JSON back into a MCCSConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
MCCSConfiguration instance populated to match JSON

Return type
MCCSConfiguration

`filter_nulls_and_validate_schema`(*data*, **_)

validating the structure of JSON against schemas and Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

`station_configs`

alias of *StnConfigurationSchema*

`subarray_beam_configs`

alias of *SubarrayBeamConfigurationSchema*

`validate_json`(*data*, *process_fn=<function MCCSConfigurationSchema.<lambda>>*)

validating the structure of JSON against schemas

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **function** (*lambda*) – use for converting list of tuples to list of list

Returns

`validate_schema`(*data*, **_)

validating the structure of JSON against schemas

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for CSP configuration

`class StnConfigurationSchema`(**args: Any*, ***kwargs: Any*)

`create`(*data*, **_)

Convert parsed JSON back into a StnConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
StnConfiguration instance populated to match JSON

Return type
StnConfiguration

class SubarrayBeamConfigurationSchema(*args: Any, **kwargs: Any)

create(data, **_) → SubarrayBeamConfiguration

Convert parsed JSON back into a SubarrayBeamConfiguration object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
SubarrayBeamConfiguration instance populated to match JSON

target

alias of SubarrayBeamTargetSchema

class SubarrayBeamTargetSchema(*args: Any, **kwargs: Any)

Marshmallow schema for the subarray_node.Target class

create_target(data, **_)

Convert parsed JSON back into a Target object.

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
Target instance populated to match JSON

13.3.20 ska_tmc_cdm.schemas.subarray_node.configure.tmc

The schemas module defines Marshmallow schemas that map CDM message classes and data model classes to/from a JSON representation.

class TMCConfigurationSchema(*args: Any, **kwargs: Any)

Create the Schema for ScanDuration using timedelta

convert_scan_duration_number_to_timedelta(data, **_)

Convert parsed JSON back into a TMConfiguration

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
TMCConfiguration instance populated to match JSON

convert_scan_duration_timedelta_to_float(data: TMCConfiguration, **_)

Process scan_duration and convert it to a float

Parameters

- **data** – the scan_duration timedelta
- **_** – kwargs passed by Marshallow

Returns
float converted

13.3.21 ska_tmc_cdm.schemas.subarray_node.scan

The ska_tmc_cdm.schemas.subarray_node.scan module contains Marshmallow schema that map ska_tmc_cdm.schemas.subarray_node.scan message classes to/from JSON.

class ScanRequestSchema(**args: Any*, ***kwargs: Any*)

ScanRequestSchema is the Marshmallow schema that marshals a ScanRequest to/from JSON.

create_scanrequest(*data*, **_)

Convert parsed JSON back into a ScanRequest

Parameters

- **data** – dict containing parsed JSON values
- **_** – kwargs passed by Marshmallow

Returns
ScanRequest instance populated to match JSON

filter_nulls(*data*, **_)

Filter out null values from JSON.

Parameters

- **data** – Marshmallow-provided dict containing parsed object values
- **_** – kwargs passed by Marshmallow

Returns
dict suitable for SubArrayNode configuration

CHAPTER

FOURTEEN

SKA-TMC-CDM DOCUMENTATION

14.1 Project description

ska-tmc-cdm provides a Python object model and serialisation library for resource allocation commands and telescope configuration commands, with a focus on TMC interfaces with other subsystems. an ICD support library, intended to be used by the Tango clients and Tango servers on opposing sides of a telescope control interface.

14.1.1 Status

This library supports control and configuration payloads for the following Tango devices:

- TMC CentralNode
- TMC SubArrayNode
- MCCSController
- MCCSSubArrayNode

14.2 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

D

E

F

O

P

R

S

T

U

V

W