developer.skatelescope.org Documentation

Release 8.1.1

Marco Bartolini

Sep 16, 2022

Table of Contents

CHAPTER 1

Quickstart

This project uses Docker containers for development and testing, and `make` to provide a consistent UI.

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

```
make test
```

Launch an interactive shell inside a container, with your workspace visible inside the container, with:

```
make interactive
```

To list all available targets, execute make without any arguments, e.g.,

```
tangodev:ska-oso-pdm $ make

build                                          build the application image

down                                           stop develop/test environment and any␣
↪interactive session

help                                           show this help.

interactive                                    start an interactive session using the␣
↪project image
                                               (caution: R/W mounts source directory to /
↪app)

lint                                           lint the application (static code␣
↪analysis)

test                                           test the application

up                                             start develop/test environment
```

CHAPTER 2

Background

The SKA Project Data Model (PDM) defines a logical data model with entities and relationships that represent all the information required to schedule and observe a fully calibratable observation on an SKA telescope.

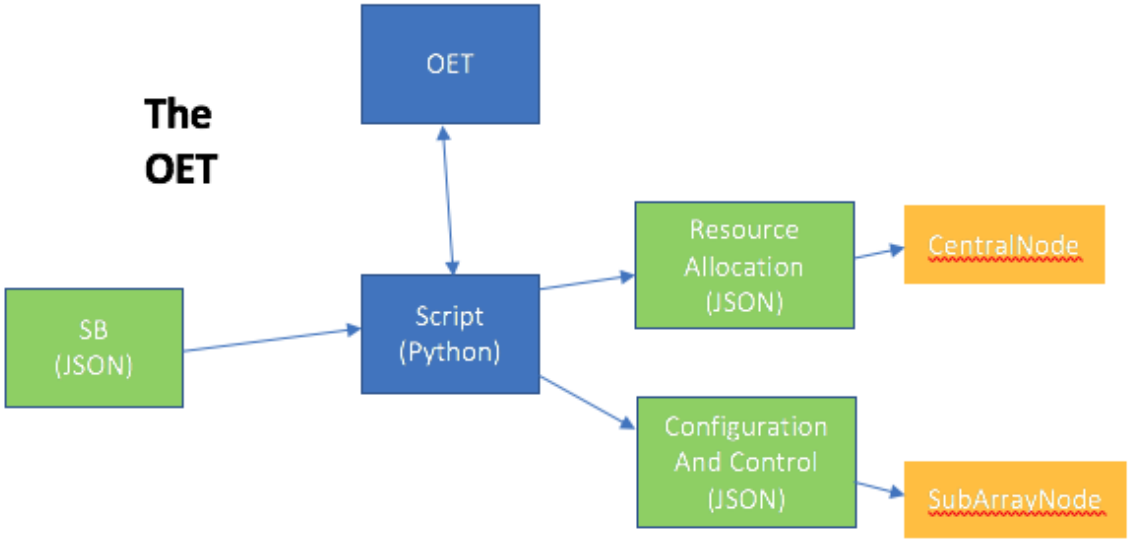
Currently, the sole consumer of this library is the Observation Execution Tool (OET), the application which provides high-level scripting facilities and a high-level scripting UI for the SKA. The Python scripts run by the OET convert Scheduling Blocks, as modelled by this library, into Configuration Data Model (CDM) JSON payloads that configure and control the telescope.

2.1 Context

The implementation context is as follows: a design has been developed for the SKA Project Data Model (SPDM); the Minimum Viable Product (MVP) provides simulated implementations of all important SKA systems; a working version of the Observation Execution Tool (OET) is available.

In operation, the OET will be given a Scheduling Block (SB), the atomic unit of an observing program, and will run a script to execute it at the telescope.

The script executed under the OET will, among other things, parse the SB JSON and split the information into the Configuration Data Model (CDM) JSON components required to configure the telescope.


The
OET
OET
SB
(JSON)
Script
(Python)
Resource
Allocation
(JSON)
CentralNode
Configuration
And Control
(JSON)
SubArrayNode

CHAPTER 3

Project layout

The PDM project contains two top-level packages, `ska_oso_pdm.entities` and `ska_oso_pdm.schemas` as shown in the figure below. The `ska_oso_pdm.entities` package contains Python object models for the logical entities in the SB data model. The `ska_oso_pdm.schemas` package contains code to transform the classes defined in `ska_oso_pdm.entities` to and from JSON.

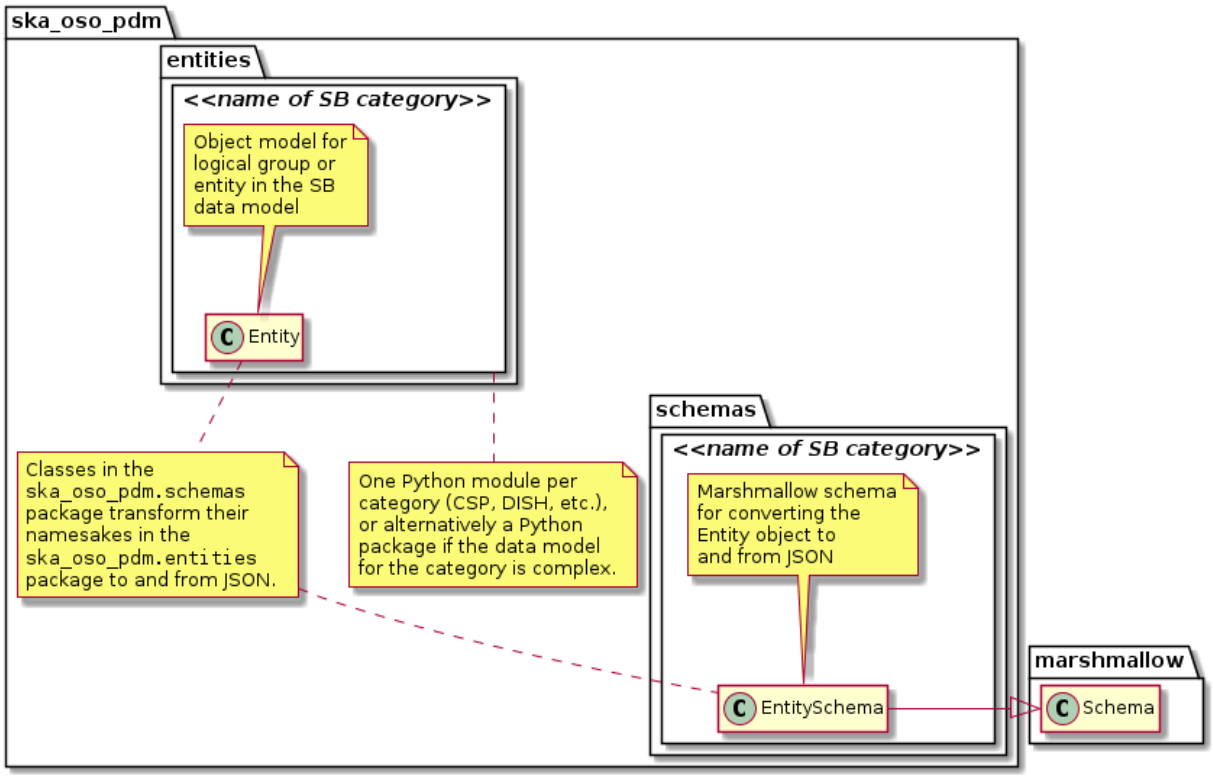


Fig. 1: Project layout and naming conventions.

The project layout and naming conventions are:

- The SB data model is divided into logical groups that collect related entities together.
- For each SB entity grouping, a corresponding Python module is created in `ska_oso_pdm.entities` and `ska_oso_pdm.schemas`. If the grouping is complex and contains many entities, the modules may be located inside a dedicated package for the group.
- Python classes representing the SB data model entities are found in the corresponding `ska_oso_pdm.entities` module/package.
- Marshmallow schema to transform `ska_oso_pdm.entities` objects to and from JSON are found in `ska_oso_pdm.schemas`.

3.1 Entities overview

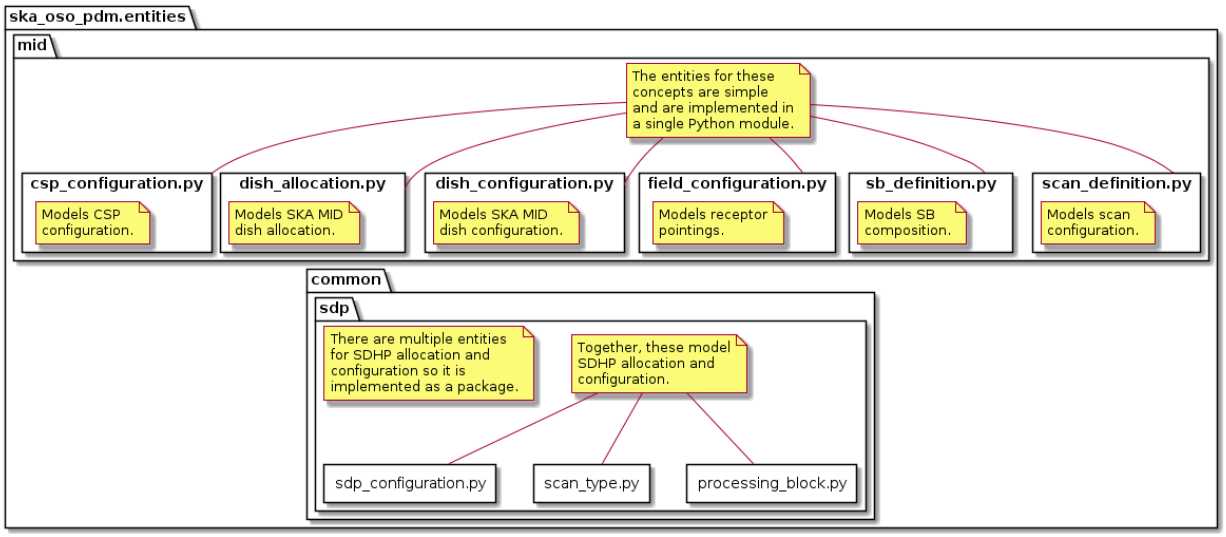


Fig. 2: Overview of PDM entities

The Python object model for the entities defined in the SB data model are located in the `ska_oso_pdm.entities` package. In general, each SB entity is represented as a Python class and each entity attribute presented as a class attribute or property.

PDM 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 entities modelled by this library, see the pages within the API documentation.

3.2 Schemas overview

Classes to marshall the `ska_oso_pdm.messages` objects to and from JSON are defined in the `ska_oso_pdm.schemas` package. This project uses Marshmallow for JSON serialisation. Classes in the `ska_oso_pdm.schemas` define Marshmallow schemas which are used by Marshmallow during JSON conversion.

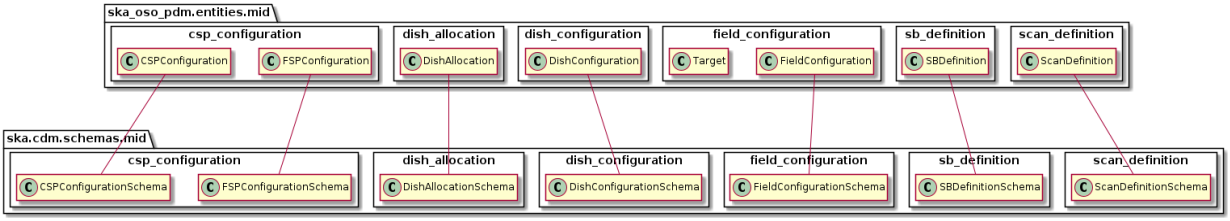


Fig. 3: Schema mapping for ska_oso_pdm.entities package

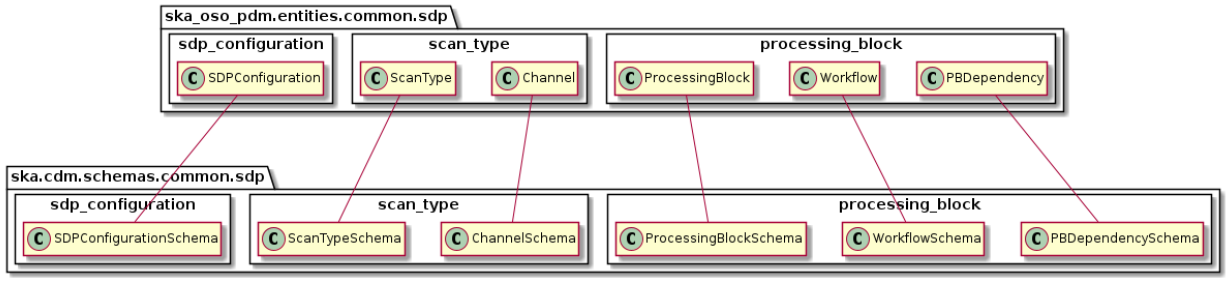


Fig. 4: Schema mapping for ska_oso_pdm.entities.sdp package

CHAPTER 4

JSON representation

4.1 MID JSON representation

A full example of a simple MID SB serialised to JSON is given below. This SBDefinition, identified as 'sbi-mvp01-20200325-00001', defines two field configurations identified as 'science field' and 'calibrator field' respectively. Field configuration 'science field' contains 2 sidereal targets, identified as beam #1 and beam #2 which point at two sidereal positions. Field configuration 'calibrator field' contains one target with ID 'bandpass calibrator' which points at Mars. Two scan definitions are defined: a scan identified as *calibrator scan*, which would observe the field *calibrator field* (=Mars) for 10 seconds, and a scan identified as *science scan*, which would observe the field *science field* (=beam #1 and beam #2) for 60 seconds. The scan_sequence says to observe *calibrator scan*, *science scan*, *science scan*, *calibrator scan*, so the final observing sequence would be:

1. Observe Mars for 10 seconds
2. Observe beam #1 and beam #2 for 60 seconds
3. Observe beam #1 and beam #2 again for 60 seconds
4. Observe Mars for 10 seconds

SBDefinition also allows to optionally include SDP and CSP configurations.

```
{
  "interface": "https://schema.skao.int/ska-oso-pdm-sbd/0.1",
  "sbd_id": "sbi-mvp01-20200325-00001",
  "telescope": "ska_mid",
  "metadata": {
    "version": 1,
    "created_by": "Liz Bartlett",
    "created_on": "2022-03-28T15:43:53.971548",
    "last_modified_on": "2022-03-28T15:43:53.971548",
    "last_modified_by": "Liz Bartlett"
  },
  "activities": {
    "allocate": {
```

(continues on next page)

(continued from previous page)


```
      "procedure_type": "filesystemscript",
      "path": "/path/to/allocatescript.py",
      "default_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "run": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    },
    "observe": {
      "procedure_type": "gitscript",
      "path": "/relative/path/to/scriptinsiderepo.py",
      "repo": "https://gitlab.com/script_repo/operational_scripts",
      "branch": "main",
      "default_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "run": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    }
  },
  "scan_definitions": [
    {
      "scan_definition_id": "calibrator scan",
      "scan_duration": 60000,
      "field_configuration": "calibrator field",
      "dish_configuration": "dish config 123",
      "scan_type": "calibration_B",
```


(continues on next page)

(continued from previous page)


```
      "csp_configuration": "csp-mvp01-20220329-00001"
    },
    {
      "scan_duration": 60000,
      "field_configuration": "science field",
      "dish_configuration": "dish config 123",
      "scan_type": "science_A",
      "scan_definition_id": "science scan"
    }
  ],
  "scan_sequence": [
    "calibrator scan",
    "science scan",
    "science scan",
    "calibrator scan"
  ],
  "field_configurations": [
    {
      "field_id": "calibrator field",
      "targets": [
        {
          "target_id": "my calibrator target",
          "kind": "sidereal",
          "target_name": "Polaris Australis",
          "reference_frame": "ICRS",
          "ra": "21:08:47.92",
          "dec": "-88:57:22.9"
        }
      ]
    },
    {
      "field_id": "science field",
      "targets": [
        {
          "target_id": "my science target",
          "kind": "sidereal",
          "target_name": "Polaris Australis",
          "reference_frame": "ICRS",
          "ra": "21:08:47.92",
          "dec": "-88:57:22.9"
        }
      ]
    }
  ],
  "sdp_configuration": {
    "eb_id": "eb-mvp01-20200325-00001",
    "max_length": 100.0,
    "scan_types": [
      {
        "scan_type_id": "science_A",
        "target": "my science target",
        "channels": [
          {
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 0.35e9,
```


(continues on next page)

(continued from previous page)

```
            "freq_max": 0.368e9,
            "link_map": [
              [
                0,
                0
              ],
              [
                200,
                1
              ],
              [
                744,
                2
              ],
              [
                944,
                3
              ]
            ]
          },
          {
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 0.36e9,
            "freq_max": 0.368e9,
            "link_map": [
              [
                2000,
                4
              ],
              [
                2200,
                5
              ]
            ]
          }
        ]
      },
      {
        "scan_type_id": "calibration_B",
        "target": "my calibrator target",
        "channels": [
          {
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 0.35e9,
            "freq_max": 0.368e9,
            "link_map": [
              [
                0,
                0
              ],
              [
                200,
                1
```

(continues on next page)

(continued from previous page)


```
            ],
            [
              744,
              2
            ],
            [
              944,
              3
            ]
          ]
        },
        {
          "count": 744,
          "start": 2000,
          "stride": 1,
          "freq_min": 0.36e9,
          "freq_max": 0.368e9,
          "link_map": [
            [
              2000,
              4
            ],
            [
              2200,
              5
            ]
          ]
        }
      ]
    }
  ],
  "processing_blocks": [
    {
      "pb_id": "pb-mvp01-20200325-00001",
      "workflow": {
        "name": "vis_receive",
        "kind": "realtime",
        "version": "0.1.0"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00002",
      "workflow": {
        "name": "test_receive_addresses",
        "kind": "realtime",
        "version": "0.3.2"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00003",
      "workflow": {
        "name": "ical",
        "kind": "batch",
        "version": "0.1.0"
      },
```


(continues on next page)

(continued from previous page)


```
        "parameters": {},
        "dependencies": [
          {
            "pb_id": "pb-mvp01-20200325-00001",
            "kind": [
              "visibilities"
            ]
          }
        ]
      },
      {
        "pb_id": "pb-mvp01-20200325-00004",
        "workflow": {
          "name": "dpreb",
          "kind": "batch",
          "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [
          {
            "kind": [
              "calibration"
            ],
            "pb_id": "pb-mvp01-20200325-00003"
          }
        ]
      }
    ]
  },
  "csp_configurations": [
    {
      "config_id": "csp-mvp01-20220329-00001",
      "subarray": {
        "subarray_name": "science period 23"
      },
      "common": {
        "subarray_id": 1,
        "band_5_tuning": [
          5.85,
          7.25
        ]
      },
      "cbf": {
        "fsp": [
          {
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
              [
                0,
                2
              ],
              [
                744,
```


(continues on next page)

(continued from previous page)


```
                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,
            "zoom_window_tuning": 650000
          }
        ]
      }
    }
  ],
  "dish_allocations": {
    "receptor_ids": [
      "0001",
      "0002"
    ]
  },
  "dish_configurations": [
    {
      "dish_configuration_id": "dci_mvp01-20220329-00001",
      "receiver_band": "1"
    }
  ]
}
```


4.2 LOW JSON representation

A full example of a simple SKA LOW SB serialised to JSON is given below. This LOW SBDefinition, identified as "sbi-mvp01-20200325-00001", defines a MCCS subarray beam, *beam A*, composed of stations 1 and 2, configured to output on one channel block. One field consisting of two targets is defined. The targets are drift scan targets, at 45 degree and 85 degree elevation respectively. One subarray beam configuration is defined. If other scans required different subarray beam configurations, they would be defined here too. The SB then defines two target beam configurations that link subarray beam configurations to targets: each configuration points beam A at 45 degree elevation and 85 degree elevation targets respectively. Two scan definitions are defined that each perform a drift scan. The first is defined as the 'calibrator scan' that is using the subarray beam configuration for the first 'target' at 45 degrees elevation, the second is the 'science scan' which uses the same subarray beam configuration with the second target at 85 degrees. Finally, the scan sequence declares the observation to consist of four scans, with a calibrator scan bookending two science scans performed back-to-back.

```
{
  "interface": "https://schema.skao.int/ska-oso-pdm-sbd/0.1",
  "sbd_id": "sbi-mvp01-20200325-00001",
  "telescope": "ska_low",
  "metadata": {
    "version": 1,
    "created_by": "Liz Bartlett",
    "created_on": "2022-03-28T15:43:53.971548"
  },
  "activities": {
    "allocate": {
      "procedure_type": "filesystemscript",
      "path": "/path/to/allocatescript.py",
      "default_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "run": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    },
    "observe": {
      "procedure_type": "gitscript",
      "path": "/relative/path/to/scriptinsiderepo.py",
      "repo": "https://gitlab.com/script_repo/operational_scripts",
      "branch": "main",
      "commit": "d234c257dadd18b3edcd990b8194c6ad94fc278a",
      "default_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "run": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
```


(continues on next page)

(continued from previous page)


```
          }
        }
      }
    }
  },
  "scan_definitions": [
    {
      "scan_definition_id": "sbi-mvp01-20220328-00001",
      "scan_duration": 64000,
      "target_beam_configurations": [
        "target #1 with beam A config 1"
      ],
      "field_configuration": "target field"
    },
    {
      "scan_definition_id": "sbi-mvp01-20220328-00002",
      "scan_duration": 64000,
      "target_beam_configurations": [
        "target #2 with beam A config 1"
      ],
      "field_configuration": "target field"
    }
  ],
  "scan_sequence": [
    "sbi-mvp01-20220328-00001",
    "sbi-mvp01-20220328-00002",
    "sbi-mvp01-20220328-00002",
    "sbi-mvp01-20220328-00001"
  ],
  "field_configurations": [
    {
      "field_id": "target field",
      "targets": [
        {
          "target_id": "target #1",
          "kind": "driftscan",
          "target_name": "Drift scan at 45 degs",
          "reference_frame": "HORIZON",
          "az": 180.0,
          "el": 45.0
        },
        {
          "target_id": "target #2",
          "kind": "driftscan",
          "target_name": "Drift scan at 85 degs",
          "reference_frame": "HORIZON",
          "az": 180.0,
          "el": 85.0
        }
      ]
    }
  ],
  "mccs_allocation": {
    "subarray_beam_ids": [
      "beam A"
    ],
    "station_ids": [
```


(continues on next page)

(continued from previous page)


```
        [
          1,
          2
        ]
      ],
      "channel_blocks": [
        1
      ]
    },
    "target_beam_configurations": [
      {
        "target_beam_id": "target #1 with beam A config 1",
        "target": "target #1",
        "subarray_beam_configuration": "beam A config 1"
      },
      {
        "target_beam_id": "target #2 with beam A config 1",
        "target": "target #2",
        "subarray_beam_configuration": "beam A config 1"
      }
    ],
    "subarray_beam_configurations": [
      {
        "subarray_beam_configuration_id": "beam A config 1",
        "subarray_beam_id": "beam A",
        "update_rate": 0.0,
        "antenna_weights": [
          1.0,
          1.0,
          1.0
        ],
        "phase_centre": [
          0.0,
          0.0
        ],
        "channels": [
          [
            0,
            8,
            1,
            1
          ],
          [
            8,
            8,
            2,
            1
          ]
        ]
      }
    ]
}
```

CHAPTER 5

ska_oso_pdm.entities.common

The common package contains modules and packages that model entities common to MID and LOW Scheduling Blocks. The contents of the module are presented in the diagram below.

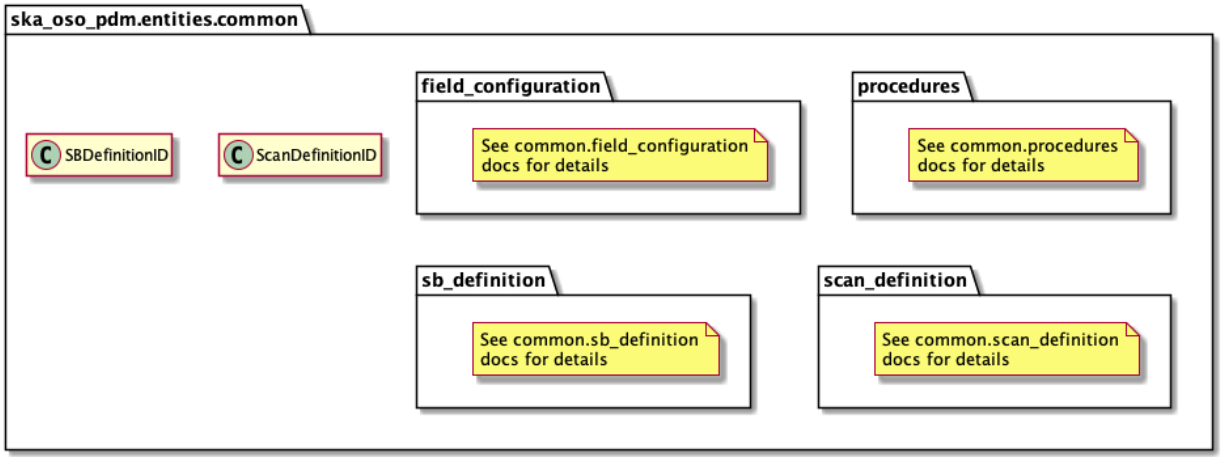


Fig. 1: Class diagram for the __init__ module

CHAPTER 6

ska_oso_pdm.entities.common.field_configuration

The field_configuration module models SB data model entities concerned with receptor pointing and mapping (source coordinates, survey fields, etc.). The contents of the module are presented in the diagram below.

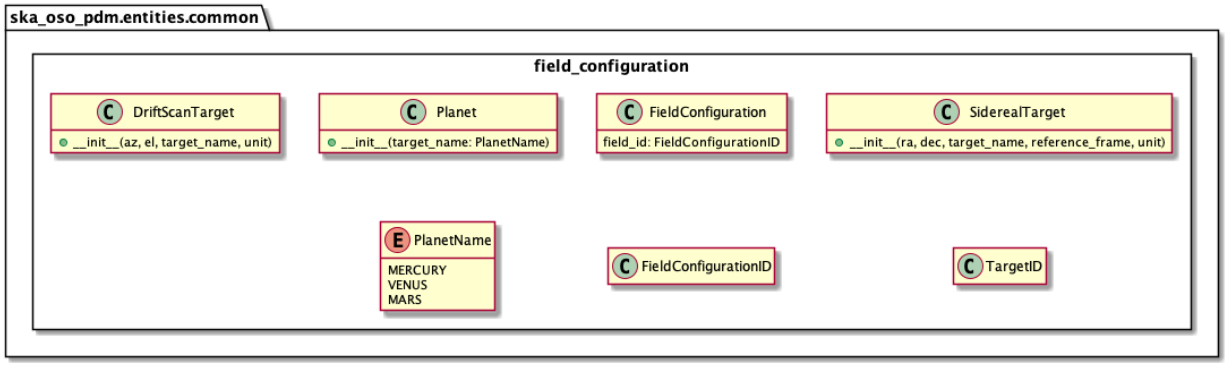


Fig. 1: Class diagram for the field_configuration module

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by field_configuration
...
"field_configurations": [
  {
    "field_id":"science fields",
      "targets": [
        {
          "target_id": "my first science target",
          "kind": "SiderealTarget",
          "target_name": "NGC6251",
          "reference_frame": "ICRS",
          "ra": "23:45:45.67",
          "dec": "-21:23:45.67"
```

(continues on next page)

(continued from previous page)

```
        },
        {
          "target_id": "my second science target",
          "kind": "SiderealTarget",
          "target_name": "M51",
          "reference_frame": "ICRS",
          "ra": "23:45:45.93",
          "dec": "-21:23:44.92"
        }
      ]
    }
],
...
```

Another JSON example with DriftScan target is shown below.

```
...
"field_configurations": [
  {
    "field_id": "target field",
    "targets": [
      {
        "target_id": "target #1",
        "kind": "driftscan",
        "target_name": "Drift scan at 45 degs",
        "reference_frame": "HORIZON",
        "az": 180.0,
        "el": 45.0
      },
      {
        "target_id": "target #2",
        "kind": "driftscan",
        "target_name": "Drift scan at 85 degs",
        "reference_frame": "HORIZON",
        "az": 180.0,
        "el": 85.0
      }
    ]
  }
],
...
```

The entities.common.field_configuration module defines a Python representation of the target of the observation.

class FieldConfiguration(*field_id: Optional[str] = None, targets: Optional[Sequence[ska_oso_pdm.entities.common.field_configuration.Target]] = None*)

FieldConfiguration represents the argument for SKA scheduling block.

FieldConfigurationID

alias of `builtins.str`

class SiderealTarget(*ra: Union[float, str], dec: Union[float, str], target_name: Optional[str] = None, reference_frame: Optional[str] = 'icrs', target_id: Optional[str] = None, unit: Union[str, Tuple[str, str]] = ('hourangle', 'deg')*)

SiderealTarget represents the argument for SKA scheduling block.

coord

abstract method : Target encapsulates source coordinates and source metadata.

class DriftScanTarget(*az: float*, *el: float*, *target_name: Optional[str] = None*, *target_id: Optional[str] = None*, *unit: str = 'deg'*)

DriftScanTarget defines AltAz target for SKA scheduling block.

coord

abstract method : Target encapsulates source coordinates and source metadata.

class Planet(*target_name: ska_oso_pdm.entities.common.field_configuration.PlanetName*, *target_id: Optional[str] = None*)

Planet represents the argument for SKA scheduling block.

coord

abstract method : Target encapsulates source coordinates and source metadata.

class PlanetName

PlanetName represents name of the planet.

TargetID

alias of `builtins.str`

CHAPTER 7

ska_oso_pdm.entities.common.procedures

The procedures modules models SB entities concerned with the script execution requirements. This includes the scripts that should process the SB and the git identifier, including the repository and branch. The contents of the modules are presented in the diagram below.

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by procedures
...
"activities":{
  "allocate": {
    "procedure_type": "filesystemscript",
    "path": "/path/to/allocatescript.py",
    "default_args": {
      "init": {"args": ["posarg1","posarg2"],
        "kwargs": {"argname": "argval"}
      },
      "run": {
        "args": ["posarg1","posarg2"],
        "kwargs": {"argname": "argval"
        }
      }
    }
  },
  "observe": {
    "procedure_type": "gitscript",
    "path": "/relative/path/to/scriptinsiderepo.py",
    "repo": "https://gitlab.com/script_repo/operational_scripts",
    "branch": "main",
    "default_args": {
      "init": {"args": ["posarg1","posarg2"],
        "kwargs": {"argname": "argval"}
      },
      "run": {
        "args": ["posarg1","posarg2"],
```

(continues on next page)

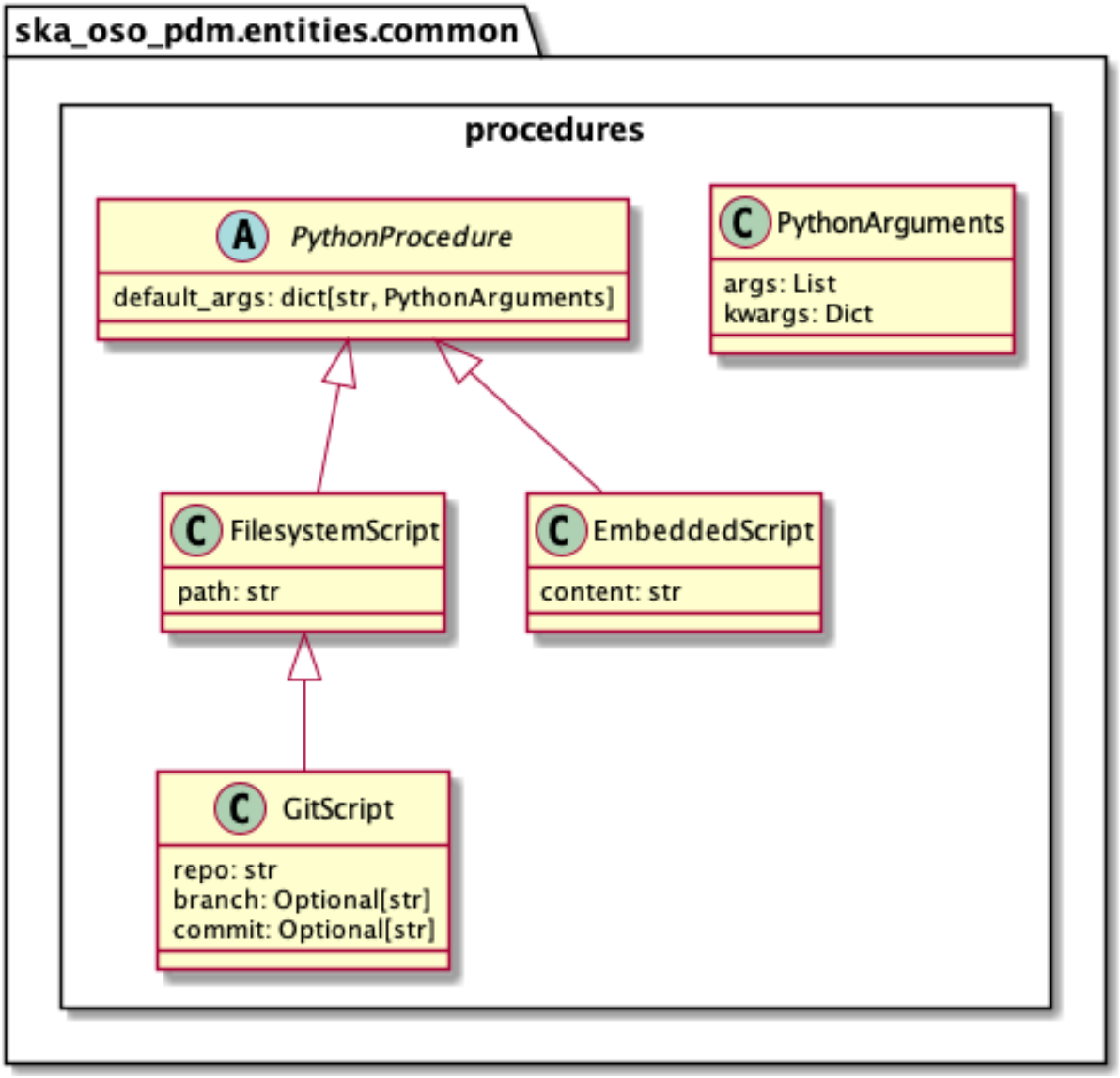


Fig. 1: Class diagram for the procedure module

(continued from previous page)

```
          "kwargs": {"argname": "argval"
          }
        }
      }
    }
}
```

The entities.common.procedures module defines a Python representation of the procedures listed in the activities of the SKA scheduling block.

class EmbeddedScript(*content: str*, *default_args: Dict[str, ska_oso_pdm.entities.common.procedures.PythonArguments] = None*)

Represents an EmbeddedScript to be ran as an activity in an SKA scheduling block.

class FileSystemScript(*path: str*, *default_args: Dict[str, ska_oso_pdm.entities.common.procedures.PythonArguments] = None*)

Represents an FileSystemScript to be ran as an activity in an SKA scheduling block.

class GitScript(*repo: str*, *path: str*, *branch: Optional[str] = None*, *commit: Optional[str] = None*, *default_args: Dict[str, ska_oso_pdm.entities.common.procedures.PythonArguments] = None*)

Represents an GitScript to be ran as an activity in an SKA scheduling block.

CHAPTER 8

ska_oso_pdm.entities.common.sb_definition

The sb_definition module models SB data model entities concerned with the high-level composition of a Scheduling Block. An SB defines everything needed to schedule and perform an observation, for instance:

- Target telescope
- Dish allocations to sub-arrays (for SKA MID);
- Dish configurations (receiver bands, etc., for SKA MID);
- MCCS resource allocations to sub-arrays (for SKA LOW);
- Sub-array Beam configurations (channels, antenna weights etc., for SKA LOW);
- Target Beam configuration, defining the required Target for subarray beam configuration (for SKA LOW);
- Targets and field positions, describing which points to observe on the sky;
- CSP (Central Signal Processing) correlator configurations to be used;
- SDHP configuration, defining the required pipeline workflows for the observation;
- Scan information, which describes which CSP/SDHP/dish/target configurations to be used for each scan;
- Scan sequence describing the sequence of scans constituting the observation.

as well as:

- SB metadata (author, creation date and version number, as well as edit history);
- SB script execution requirements;

The contents of the module are presented in the diagram below.

```
# JSON modelled specifically by sb_definition
{
  "sbd_id": "sbi-mvp01-20200325-00001",
  "telescope": "ska_mid",
  "interface": "https://schema.skao.int/ska-oso-pdm-sbd/0.1",
  "metadata": {
    "version": 1,
```

(continues on next page)

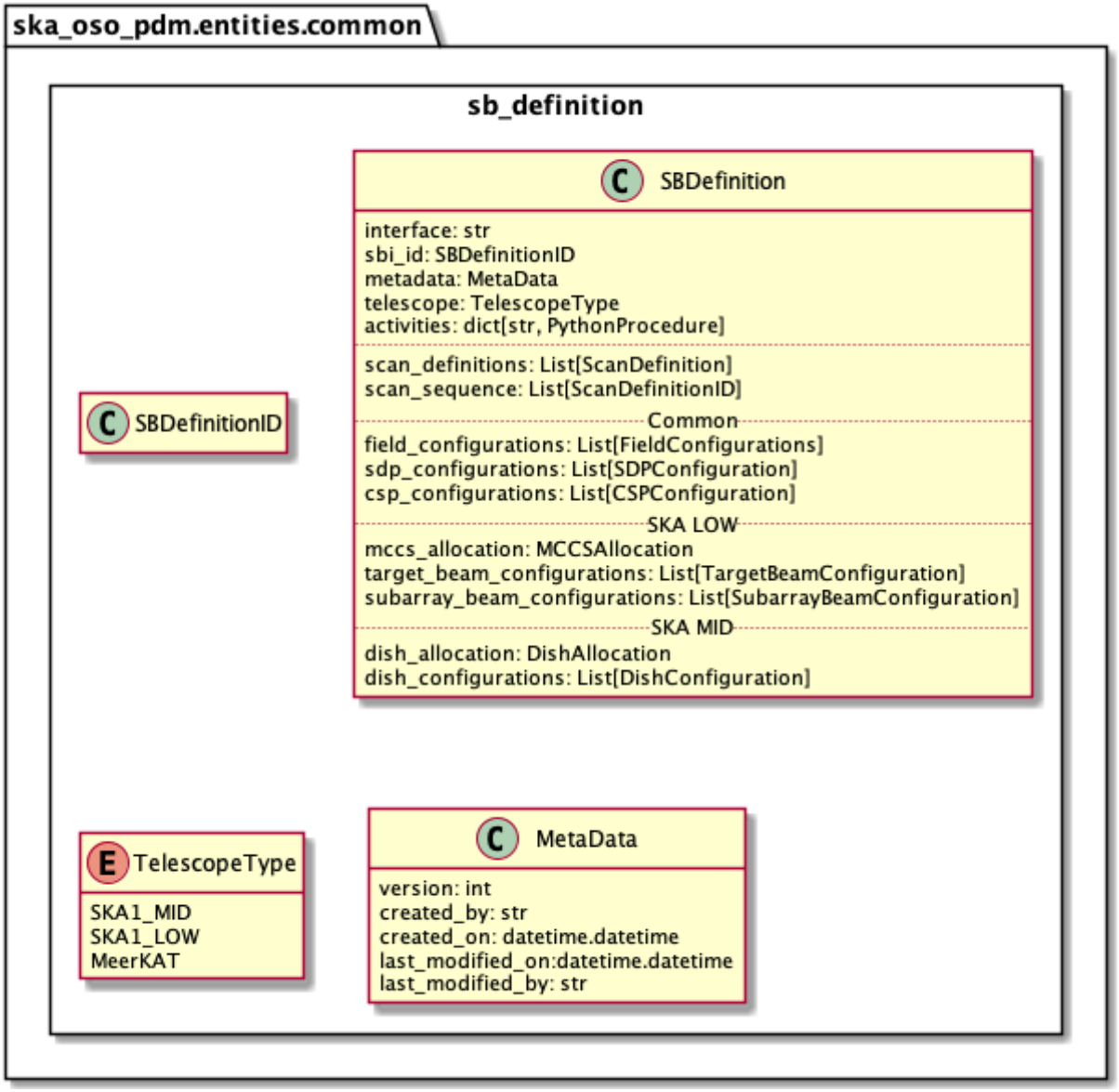


Fig. 1: Class diagram for the sb_definition module

(continued from previous page)

```
    "created_by": "Liz Bartlett",
    "created_on": "2022-03-28T15:43:53.971548",
    "last_modified_on": null,
    "last_modified_by": null
    }
  "activities": ...
  "dish_allocations": ...
  "csp_configurations": ...
  "dish_configurations" : ...
  "field_configurations": ...
  "sdp_configuration": ...
  "scan_definitions": ...
  "scan_sequence": [
    "A", "B", "B", "A"
  ]
}
```

The entities.scheduling_block_entity module defines a simple Python representation of the scheduling block that contains the details of the observation

class SBDefinition(*, *interface: Optional[str] = 'https://schema.skao.int/ska-oso-pdm-sbd/0.1'*, *sbd_id: Optional[str] = None*, *telescope: Optional[ska_oso_pdm.entities.common.sb_definition.TelescopeType] = None*, *metadata: Optional[ska_oso_pdm.entities.common.sb_definition.MetaData] = None*, *activities: Optional[Dict[str, ska_oso_pdm.entities.common.procedures.PythonProcedure]] = None*, *field_configurations: Optional[List[ska_oso_pdm.entities.common.field_configuration.FieldConfig = None*, *scan_definitions: Optional[List[ska_oso_pdm.entities.common.scan_definition.ScanDefinition]] = None*, *scan_sequence: Optional[List[str]] = None*, *sdp_configuration: Optional[ska_oso_pdm.entities.sdp.sdp_configuration.SDPConfiguration] = None*, *dish_configurations: Optional[List[ska_oso_pdm.entities.dish.dish_configuration.DishConfiguration]] = None*, *csp_configurations: Optional[List[ska_oso_pdm.entities.csp.common.CSPConfiguration]] = None*, *dish_allocations: Optional[ska_oso_pdm.entities.dish.dish_allocation.DishAllocation] = None*, *mccs_allocation: Optional[ska_oso_pdm.entities.mccs.mccs_allocation.MCCSAllocation] = None*, *subarray_beam_configurations: Optional[List[ska_oso_pdm.entities.mccs.subarray_beam_configuration.SubarrayBeamConfiguration]] = None*, *target_beam_configuration = None*)

SKA scheduling block

class TelescopeType

TelescopeType represents which telescope is being used

class MetaData(*, *version: Optional[int] = 1*, *created_on: Optional[datetime.datetime] = None*, *created_by: Optional[str] = None*, *last_modified_on: Optional[datetime.datetime] = None*, *last_modified_by: Optional[str] = None*)

MetaData Class

CHAPTER 9

ska_oso_pdm.entities.common.scan_definition

The scan_definition module models SB entities concerned with the selection of which element configurations should take effect for the duration of a scan. The contents of the module are presented in the diagram below.

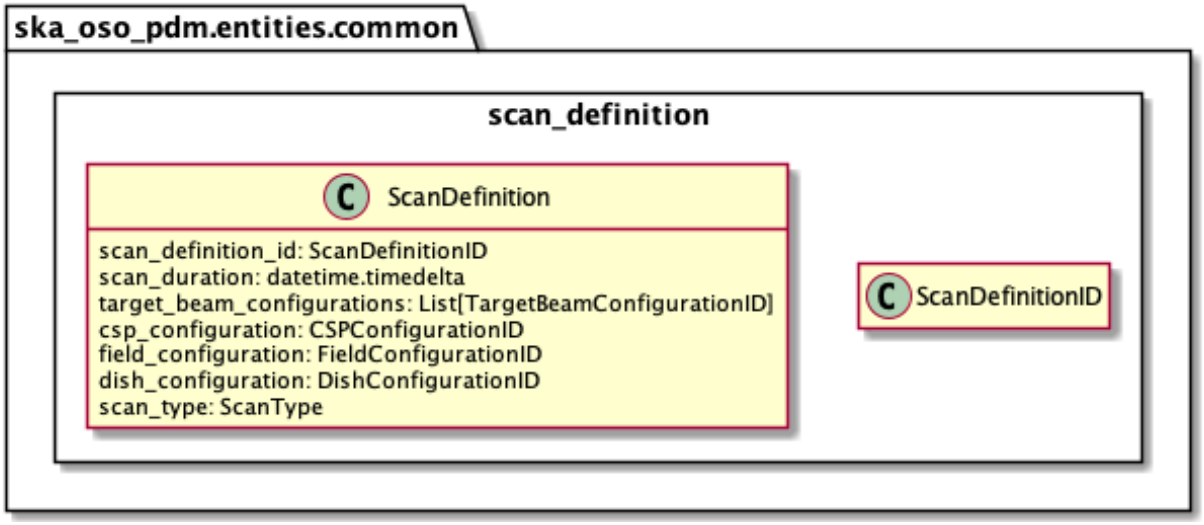


Fig. 1: Class diagram for the scan_definition module

An example serialisation of this model to JSON for SKA MID is shown below.

```
# JSON modelled specifically by scan_definition
...
"scan_definitions": [
  {
    "scan_definition_id": "calibrator scan",
    "scan_duration": 60000,
    "field_configuration": "calibrator field",
    "dish_configuration": "dish config 123",
```

(continues on next page)

(continued from previous page)

```
    "scan_type": "calibration_B",
    "csp_configuration": "csp-mvp01-20220329-00001"
  },
  {
    "scan_duration": 60000,
    "field_configuration": "science field",
    "dish_configuration": "dish config 123",
    "scan_type": "science_A",
    "scan_definition_id": "science scan"
  }
],
```

A JSON code example for SKA LOW is shown below.

```
# JSON modelled specifically by scan_definition

"scan_definitions": [
  {
    "scan_definition_id": "sbi-mvp01-20220328-00001",
    "scan_duration": 64000,
    "target_beam_configurations": [
      "target #1 with beam A config 1"
    ],
```

The entities.scan_definition_entity module defines simple Python representation of a single observation scan

class ScanDefinition(*scan_definition_id: Optional[str], scan_duration: datetime.timedelta, field_configuration_id: str, target_beam_configuration_ids: Optional[List[str]] = None, dish_configuration_id: Optional[str] = None, scan_type_id: Optional[str] = None, csp_configuration_id: Optional[str] = None*)

ScanDefinition represents the instrument configuration for a single scan.

ScanDefinitionID

alias of `builtins.str`

CHAPTER 10

ska_oso_pdm.entities.csp.common

The csp.common module models SB entities concerned with CSP (Central Signal Processing) beam former and correlator configuration. The contents of the module are presented in the diagram below.

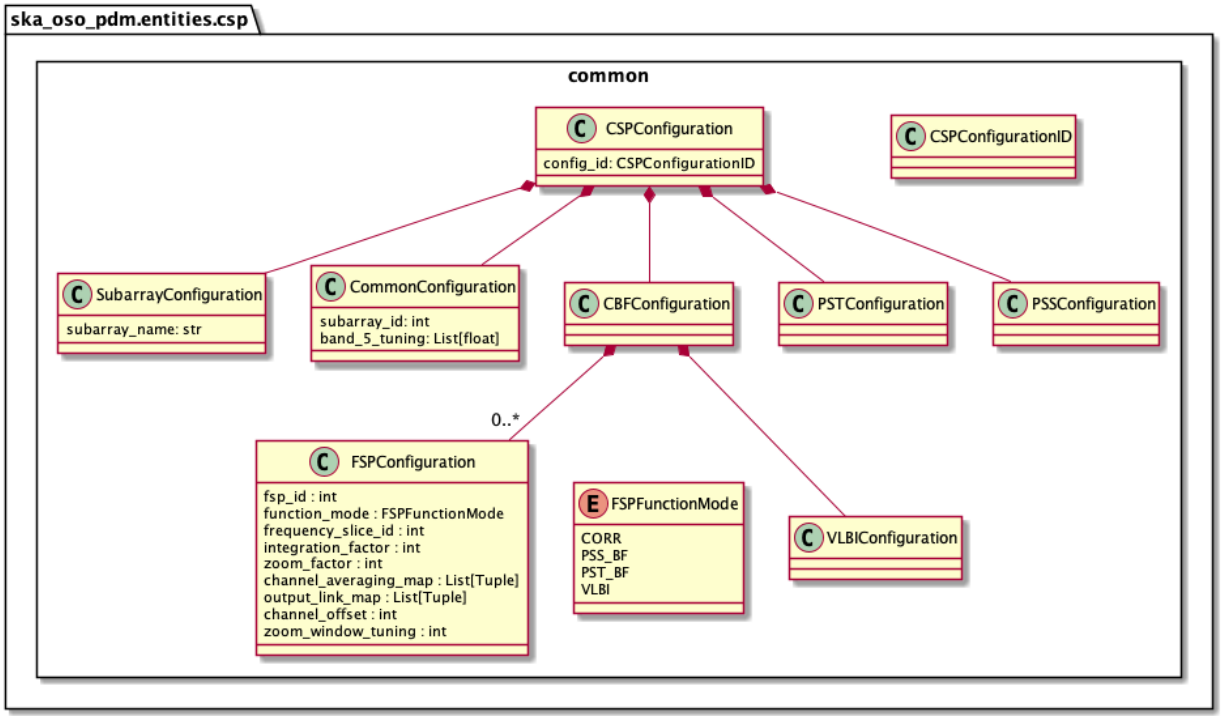


Fig. 1: Class diagram for the common module within csp

An example serialisation of this model to JSON is shown below.

```
  # JSON modelled specifically by csp_configuration
  ...
"csp_configurations": [{
    "config_id": "csp-mvp01-20220329-00001",
    "subarray": {"subarray_name": "science period 23"},
    "common": {
      "subarray_id": 1,
      "band_5_tuning": [5.85,7.25]
    },
    "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]],
          "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,
          "zoom_window_tuning": 650000
        }]
    }
  }
],
  ...
```

The entities.csp_configuration module defines a simple Python representation of CSP and FSP configurations.

class CSPConfiguration(*config_id: str = None*, *subarray_config: ska_oso_pdm.entities.csp.common.SubarrayConfiguration = None*, *common_config: ska_oso_pdm.entities.csp.common.CommonConfiguration = None*, *cbf_config: ska_oso_pdm.entities.csp.common.CBFConfiguration = None*, *pst_config: ska_oso_pdm.entities.csp.common.PSTConfiguration = None*, *pss_config: ska_oso_pdm.entities.csp.common.PSSConfiguration = None*)

Class to hold all CSP configuration.

CSPConfigurationID

alias of `builtins.str`

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

FSPConfiguration defines the configuration for a CSP Frequency Slice Processor.

class FSPFunctionMode

FSPFunctionMode is an enumeration of the available FSP modes.

class CBFConfiguration(*fsp_configs: List[ska_oso_pdm.entities.csp.common.FSPConfiguration]*, *vlbi_config: Optional[ska_oso_pdm.entities.csp.common.VLBIConfiguration] = None*)

Class to hold all FSP and VLBI configurations.

class SubarrayConfiguration(*subarray_name: str*)

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

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

Class to hold the CSP sub-elements.

CHAPTER 11

ska_oso_pdm.entities.sdp

The sdp package contains modules that model SB entities concerned with SDHP resource allocation and pipeline workflow configuration. The contents of the module are presented in the diagram below.

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by entities in the sdp package
...
"sdp_configuration": {
  "eb_id": "eb-mvp01-20200325-00001",
  "max_length": 100.0,
  "scan_types": [{
      "scan_type_id": "science_A",
      "target": "my science target",
      "channels": [{
          "count": 744,
          "start": 0,
          "stride": 2,
          "freq_min": 0.35e9,
          "freq_max": 0.368e9,
          "link_map": [[0,0],[200,1],[744,2],[944,3]]
        },
        {
          "count": 744,
          "start": 2000,
          "stride": 1,
          "freq_min": 0.36e9,
          "freq_max": 0.368e9,
          "link_map": [[2000,4],[2200,5]]
          }]},
    {
      "scan_type_id": "calibration_B",
      "target": "my calibrator target",
      "channels": [{
          "count": 744,
          "start": 0,
```

(continues on next page)

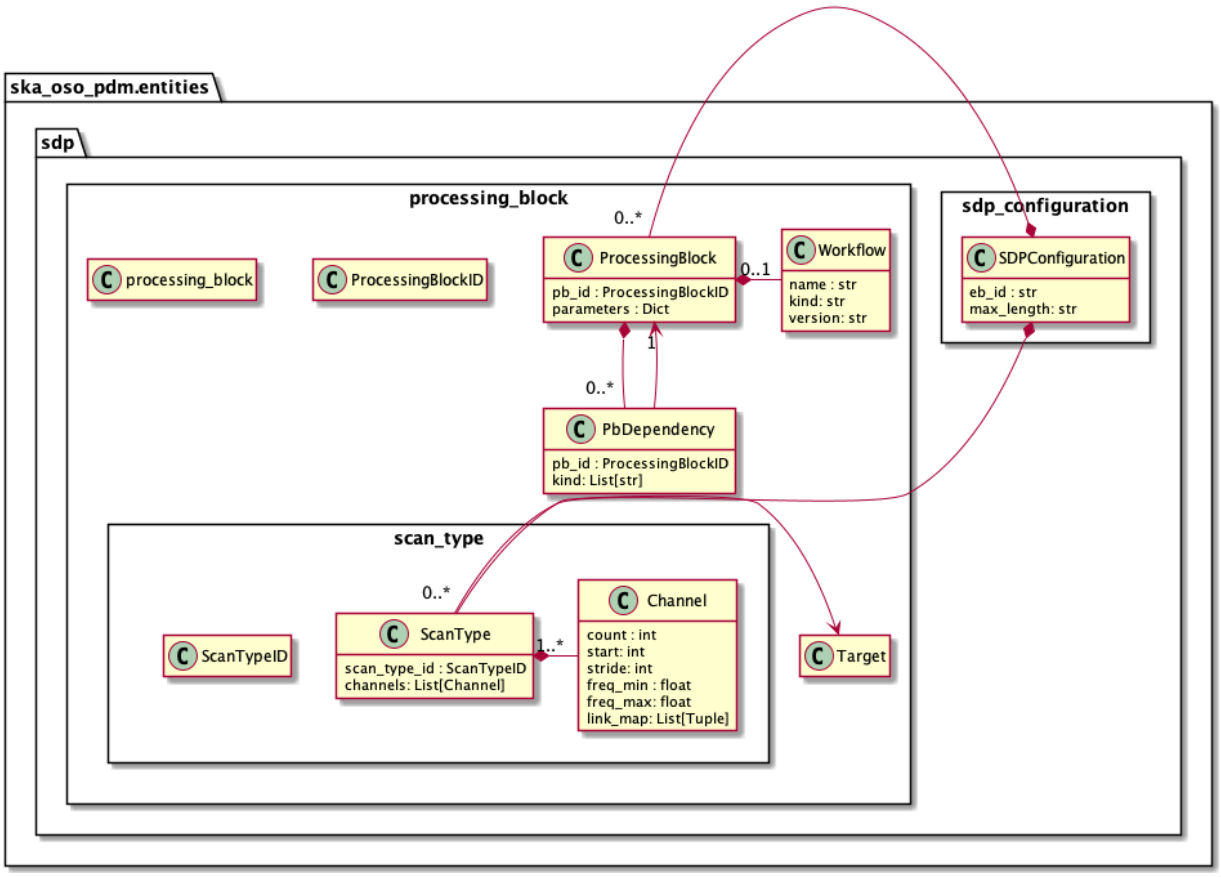


Fig. 1: Class diagram for the sdp package

(continued from previous page)

```
          "stride": 2,
          "freq_min": 0.35e9,
          "freq_max": 0.368e9,
          "link_map": [[0,0],[200,1],[744,2],[944,3]]
        },
        {
          "count": 744,
          "start": 2000,
          "stride": 1,
          "freq_min": 0.36e9,
          "freq_max": 0.368e9,
          "link_map": [[2000,4],[2200,5]]
        }]}],
  "processing_blocks": [{
      "pb_id": "pb-mvp01-20200325-00001",
      "workflow": {
        "name": "vis_receive",
        "kind": "realtime",
        "version": "0.1.0"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00002",
      "workflow": {
        "name": "test_receive_addresses",
        "kind": "realtime",
        "version": "0.3.2"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00003",
      "workflow": {
        "name": "ical",
        "kind": "batch",
        "version": "0.1.0"
      },
      "parameters": {},
      "dependencies": [{
          "pb_id": "pb-mvp01-20200325-00001",
          "kind": ["visibilities"]
        }]
    },
    {
      "pb_id": "pb-mvp01-20200325-00004",
      "workflow": {
        "name": "dpreb",
        "kind": "batch",
        "version": "0.1.0"
      },
      "parameters": {},
      "dependencies": [{
          "kind": ["calibration"],
          "pb_id": "pb-mvp01-20200325-00003"
        }]
    }
```

(continues on next page)

(continued from previous page)

```
  ]
},



"sdpConfiguration": {
  "eb_id": "eb-mvp01-20200325-00001",
  "max_length": 100.0,
  "scan_types": [
    {
      "scan_type_id": "science_A",
      "target": "beam #1",
      "channels": [{
         "count": 744, "start": 0, "stride": 2,
         "freq_min": 0.35e9,
         "freq_max": 0.368e9,
         "link_map": [[0, 0], [200, 1], [744, 2], [944, 3]]
       },
       {
         "count": 744, "start": 2000, "stride": 1,
         "freq_min": 0.36e9,
         "freq_max": 0.368e9,
         "link_map": [[2000, 4], [2200, 5]]
       }]
    },
    {
      "scan_type_id": "calibration_B",
      "target": "bandpass calibrator",
      "channels": [{
        "count": 744, "start": 0, "stride": 2,
        "freq_min": 0.35e9,
        "freq_max": 0.368e9,
        "link_map": [[0, 0], [200, 1], [744, 2], [944, 3]]
      },
      {
        "count": 744, "start": 2000, "stride": 1,
        "freq_min": 0.36e9,
        "freq_max": 0.368e9,
        "link_map": [[2000, 4], [2200, 5]]
      }]
    }
  ],
  "processing_blocks": [
    {
      "pb_id": "pb-mvp01-20200325-00001",
      "workflow": {"name": "vis_receive", "kind": "realtime", "version": "0.1.0"},
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00003",
      "workflow": {"name": "ical", "kind": "batch", "version": "0.1.0"},
      "parameters": {},
      "dependencies": [
        {"pb_id": "pb-mvp01-20200325-00001", "kind": ["visibilities"]}
      ]
    }
  ]
```

(continues on next page)

(continued from previous page)

```
},
...
```

The ska_oso_pdm.entities.common.sdp.sdp_configuration module defines a Python object model for the SDP configuration JSON string passed to CentralNode.AssignResources.

class SDPConfiguration(*eb_id: str, max_length: float, scan_types: List[ska_oso_pdm.entities.sdp.scan_type.ScanType], processing_blocks: List[ska_oso_pdm.entities.sdp.processing_block.ProcessingBlock]*)
SDPConfiguration captures the SDP resources and pipeline configuration required to process an execution block.

The entities.sdp.scan_type module defines a Python representation of a scan type for SDP configuration.

class ScanType(*scan_type_id: str, target_id: str, channels: List[ska_oso_pdm.entities.sdp.scan_type.Channel]*)
Class to hold ScanType configuration

ScanTypeID
alias of `builtins.str`

class Channel(*count: int, start: int, stride: int, freq_min: float, freq_max: float, link_map: List[Tuple]*)
Class to hold Channels for ScanType

The entities.sdp.processing_block module defines a Python representation of a processing block for SDP configuration.

class Workflow(*name: str*, *kind: str*, *version: str*)
Class to hold Workflow for ProcessingBlock

class ProcessingBlock(*pb_id: str, workflow: ska_oso_pdm.entities.sdp.processing_block.Workflow, parameters: Dict[KT, VT], dependencies: List[ska_oso_pdm.entities.sdp.processing_block.PbDependency] = None*)
Class to hold ProcessingBlock configuration

class PbDependency(*pb_id: str, kind: List[str]*)
Class to hold Dependencies for ProcessingBlock

CHAPTER 12

ska_oso_pdm.entities.dish.dish_allocation

The dish_allocation module defines which SKA MID dishes should be allocated to a sub-array prior to an observation.

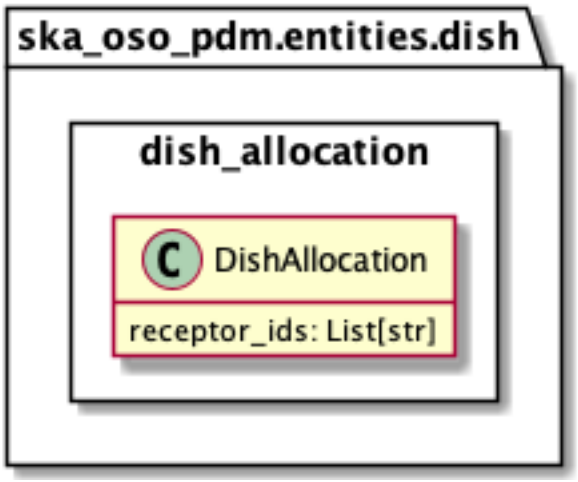


Fig. 1: Class diagram for the dish_allocation module

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by dish_allocation
...
"dish_allocations": {
   "receptor_ids": ["0001","0002"]
},
...
```

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.

CHAPTER 13

ska_oso_pdm.entities.dish.dish_configuration

The dish_configuration module models SB entities concerned with SKA MID dish configuration.

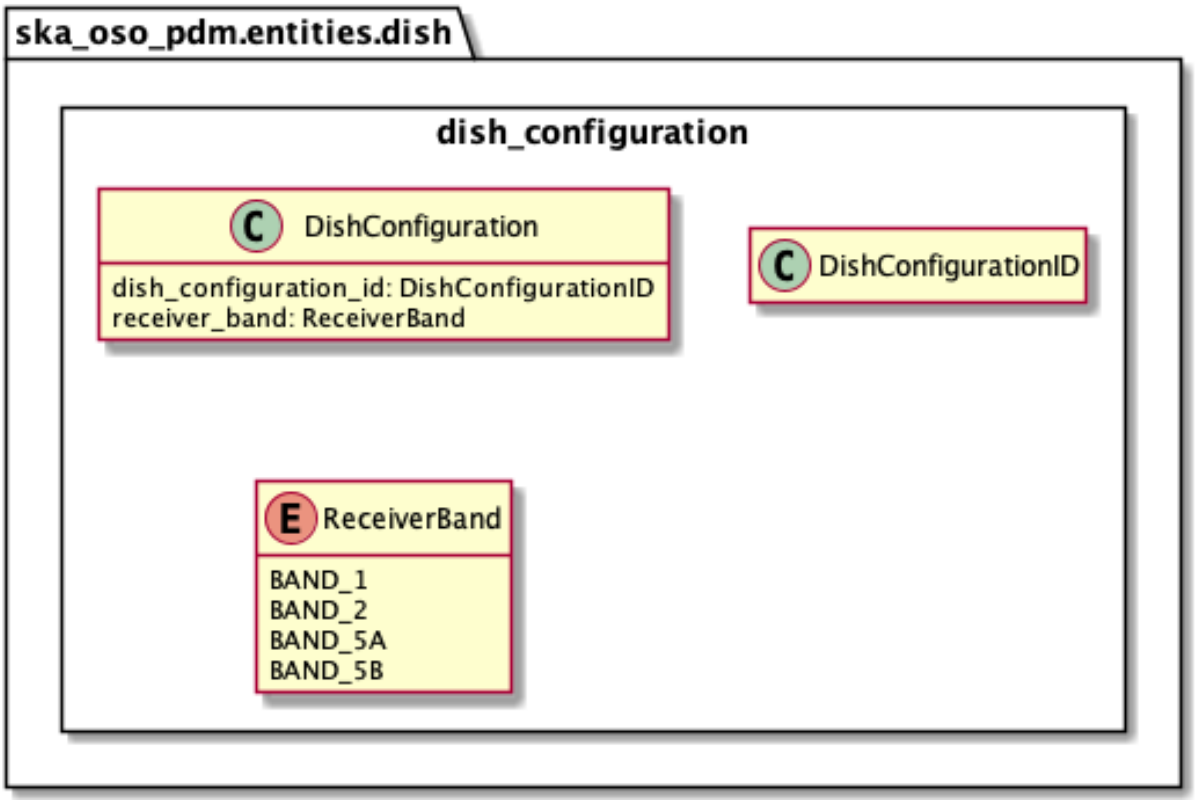


Fig. 1: Class diagram for the dish_configuration module

An example serialisation of this model to JSON is shown below.

```
 # JSON modelled specifically by dish_configuration
 ...
"dish_configurations" : [
    {
        "dish_configuration_id": "dci_mvp01-20220329-00001",
        "receiver_band" : "1"
    }
],
...
```

The entities.dish_configuration module defines simple Python representation of how SKA MID dishes in sub-array should be configured.

DishConfigurationID
alias of `builtins.str`

class ReceiverBand
ReceiverBand is an enumeration of SKA MID receiver bands.

class DishConfiguration(*dish_configuration_id: str*, *receiver_band: ska_oso_pdm.entities.dish.dish_configuration.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.

CHAPTER 14

ska_oso_pdm.entities.mccs.mccs_allocation

The mccs_allocation module defines which SKA LOW stations should be allocated to a sub-array beam prior to an observation.

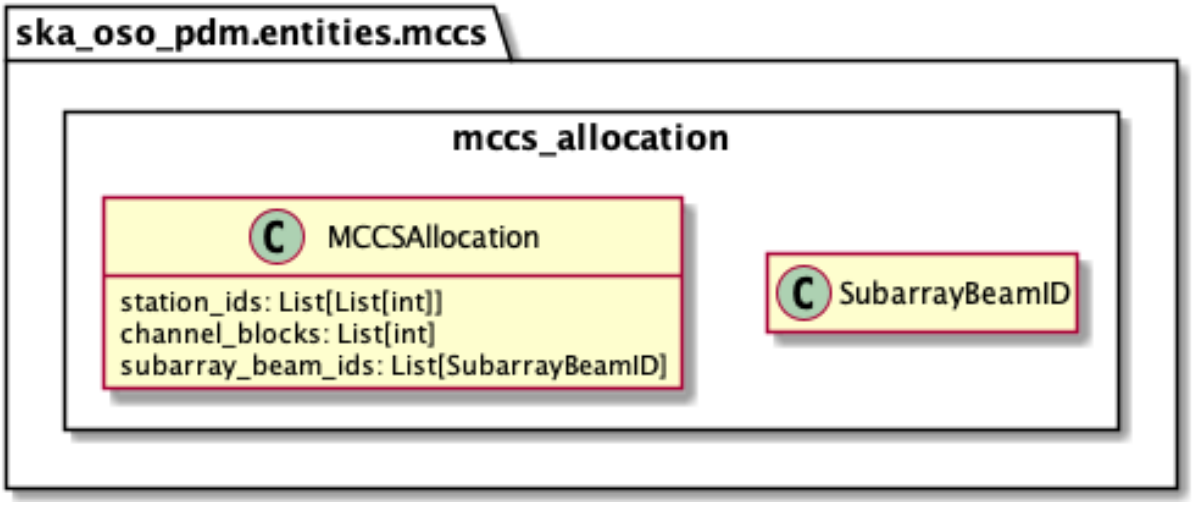


Fig. 1: Class diagram for the mccs_allocation module

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by mccs_allocation
...
"mccs_allocation": {
   "subarray_beam_ids": ["Beam A"],
   "station_ids": [[1,2]],
   "channel_blocks": [3]
},
...
```

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

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

MCCSAllocation is a Python representation of the MCCS allocation segment of a scheduling block.

SubarrayBeamID

alias of `builtins.str`

CHAPTER 15

ska_oso_pdm.entities.mccs.subarray_beam_configuration

The subarray_beam_configuration module models SKA LOW SB entities. The contents of the module are presented in the diagram below.

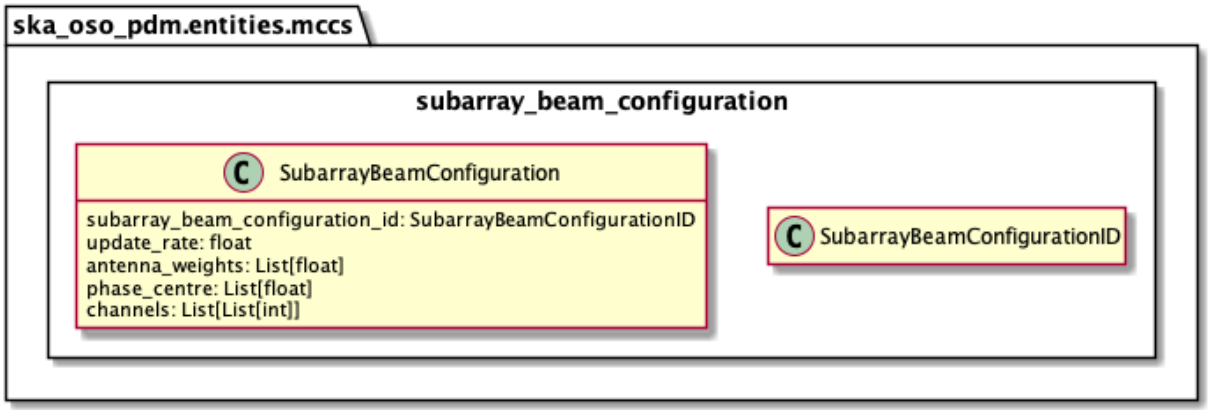


Fig. 1: Class diagram for the subarray_beam_configuration module

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by subarray_beam_configuration
...
"subarray_beam_configuration": {
    "subarray_beam_configuration_id": "Beam A config 1",
    "subarray_beam_id": "Beam A",
    "update_rate": 0.0,
    "antenna_weights": [1.0, 1.0, 1.0],
    "phase_centre": [0.0, 0.0],
    "channels": [
      [0, 8, 1, 1],
      [8, 8, 2, 1],
      [24, 16, 2, 1]
```

(continues on next page)

(continued from previous page)

```
    ]
},
...
```

The entities.subarray_beam_configuration module defines simple Python representation of how SKA Low subarray beam in sub-array should be configured.

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

SubarrayBeamConfiguration specifies how SKA LOW sub-array should be configured.

SubarrayBeamConfigurationID

alias of `builtins.str`

CHAPTER 16

ska_oso_pdm.entities.low.target_beam_configuration

The target_beam_configuration module defines which SKA LOW target will be mapped to the subarray beam configurations.

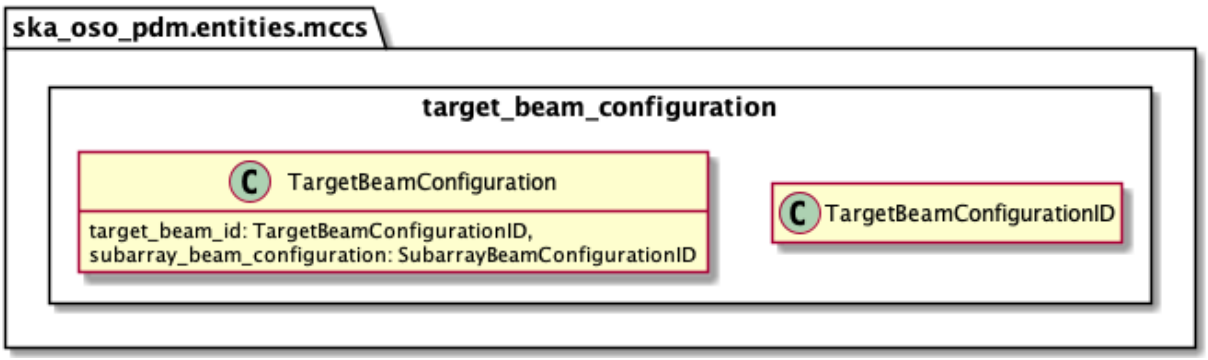


Fig. 1: Class diagram for the target_beam_configuration module

An example serialisation of this model to JSON is shown below.

```
# JSON modelled specifically by target_beam_configuration
...
"target_beam_configurations": [
   {
        "target_beam_id": "science target beam",
        "target": "my science target",
        "subarray_beam_configuration": "subarray beam 1"
   }
],
...
```

The entities.target_beam_configuration module defines simple Python representation of how SKA Low subarray beam in sub-array should be configured.

class TargetBeamConfiguration(*target_beam_id: str*, *target: str*, *subarray_beam_configuration: str*)

TargetBeamConfiguration specifies how SKA LOW subarray beam in a sub-array should be configured.

TargetBeamConfigurationID

alias of `builtins.str`

CHAPTER 17

ska_oso_pdm.schemas.codec

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

class MarshmallowCodec

MarshmallowCodec marshalls and unmarshalls PDM classes.

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

dumps(*obj*)

Return a string JSON representation of a PDM instance.

Parameters **obj** – the instance to marshall to JSON

Returns a JSON string

load_from_file(*cls*, *path*)

Load an instance of a PDM class from disk.

Parameters

- **cls** – the class to create from the file
- **path** – the path to the file

Returns an instance of cls

loads(*pdm_class*, *json_data*)

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

Parameters

- **pdm_class** – the class to create from the JSON
- **json_data** – the JSON to unmarshall

Returns an instance of cls

register_mapping(*pdm_class*)

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

Parameters **pdm_class** – the PDM class this schema maps to

set_schema(*pdm_class*, *schema_class*)
Set the schema for a PDM class.

Parameters

- **schema_class** – Marshmallow schema to map
- **pdm_class** – PDM class the schema maps to

CHAPTER 18

ska_oso_pdm.schemas.shared

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

class UpperCasedField(**args*, ***kwargs*)
Field that serializes to an upper-case string and deserializes to a lower-case string.

class NestedDict(*nested*, *key*, **args*, ***kwargs*)
Field that serialises a list to a dict, with the specified attribute acting as key.

CHAPTER 19

ska_oso_pdm.schemas.common

CHAPTER 20

ska_oso_pdm.schemas.common.field_configuration

The schemas.common.field_configuration defines Marshmallow schema that map the field_configurations section of an SKA scheduling block to/from a JSON representation.

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

The Field Configuration section of an SKA scheduling block

make_fieldconfiguration (*data*, **_)

Convert parsed JSON back into a FieldConfiguration object.

Parameters

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

Returns FieldConfiguration instance populated to match JSON

CHAPTER 21

ska_oso_pdm.schemas.common.procedures

The schemas.common.procedures defines Marshmallow schema that map the activities section of an SKA scheduling block to/from a JSON representation.

class EmbeddedScriptSchema(**args*, ***kwargs*)
Schema for an EmbeddedScript, used in the activities section of an SKA scheduling block

make_embeddedscript(*data*, **_)
Convert parsed JSON back into a EmbeddedScript object.

class FileSystemScriptSchema(**args*, ***kwargs*)
Schema for a FileSystemScript, used in the activities section of an SKA scheduling block

make_filesystemscript(*data*, **_)
Convert parsed JSON back into a FileSystemScript object.

class GitScriptSchema(**args*, ***kwargs*)
Schema for a GitScript, used in the activities section of an SKA scheduling block

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

make_gitscript(*data*, **_)
Convert parsed JSON back into a GitScript object.

class PythonArgumentsSchema(**args*, ***kwargs*)
Schema for the PythonArguments, used in the activities section of an SKA scheduling block

make_pythonarguments(*data*, **_)
Convert parsed JSON back into a PythonArguments object.

class PythonProcedureSchema(**args*, ***kwargs*)

Schema for an abstract PythonProcedure, used in the activities section of anSKA scheduling block

CHAPTER 22

ska_oso_pdm.schemas.common.sb_definition

The schemas.scheduling_block_schema defines a Marshmallow schema that maps the scan definition section of an SKA scheduling block to/from a JSON representation.

class MetaDataSchema (**args*, ***kwargs*)
The MetaData section of an SKA scheduling block

create_metadata (*data*, ***_*)
Convert parsed JSON back into a metadata

Parameters

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

Returns SBDefinition 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 metadata

class SBDefinitionSchema (**args*, ***kwargs*)
SKA scheduling block

create_scheduling_block (*data*, ***_*)
Convert parsed JSON back into a ScanRequest

Parameters

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

Returns SBDefinition 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 scan definition

CHAPTER 23

ska_oso_pdm.schemas.common.scan_definition

The schemas.scan_definition_schema defines a Marshmallow schema that maps The scan definition section of an SKA scheduling block to/from a JSON representation.

class ScanDefinitionSchema (**args*, ***kwargs*)
The scan definition section of an SKA scheduling block

create_scan_definition (*data*, ***_*)
Convert parsed JSON back into a ScanDefinition

Parameters

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

Returns ScanDefinitions 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 scan definition

CHAPTER 24

ska_oso_pdm.schemas.csp.common

The schemas.csp_schema defines Marshmallow schemas that map the CSP definition section of an SKA scheduling block to/from JSON.

class CSPConfigurationSchema (**args*, ***kwargs*)
Marshmallow schema for the ska_oso_pdm.CSPConfiguration class

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

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

class FSPConfigurationSchema (**args*, ***kwargs*)
Marshmallow schema for the FSPConfiguration

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 SubarrayConfigurationSchema(**args*, ***kwargs*)
Marshmallow schema for the SubarrayConfigurationSchema

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 CommonConfigurationSchema(**args*, ***kwargs*)
Marshmallow schema for the CommonConfigurationSchema

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 CBFConfigurationSchema(**args*, ***kwargs*)
Marshmallow schema for the CBFConfigurationSchema

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*

CHAPTER 25

ska_oso_pdm.schemas.sdp

The schemas.sdp.sdp_configuration module defines Marshmallow schemas .

class SDPConfigurationSchema(**args*, ***kwargs*)
Marshmallow 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

The schemas.sdp.scan_type module defines Marshmallow schemas .

class ChannelSchema(**args*, ***kwargs*)
Marshmallow schema for the Channel 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

class ScanTypeSchema(**args*, ***kwargs*)
Marshmallow schema for the ScanType class.

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

The schemas.sdp.processing_block module defines Marshmallow schemas .

class WorkflowSchema(**args*, ***kwargs*)
Represents the type of workflow being configured on the SDP

create_sdp_wf(*data*, **_)
Convert parsed JSON back into a Workflow object.

Parameters

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

Returns SDP Workflow object populated from data

class PbDependencySchema(**args*, ***kwargs*)
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 ProcessingBlockSchema(**args*, ***kwargs*)
Marshmallow schema for the ProcessingBlock class.

create_processing_block(*data*, **_)
Convert parsed JSON back into a PB 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

CHAPTER 26

ska_oso_pdm.schemas.dish.dish_allocation

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

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

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

CHAPTER 27

ska_oso_pdm.schemas.dish.dish_configuration

The schemas.dish_configuration_schema defines a Marshmallow schema that maps SKA MID dishes to/from a JSON representation.

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

The dish configuration schema section of an SKA scheduling block

create_scan_definition (*data*, ***_*)

Convert parsed JSON back into a DishConfiguration

Parameters

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

Returns DishConfiguration instance populated to match JSON

CHAPTER 28

ska_oso_pdm.schemas.mccs.mccs_allocation

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

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

Marshmallow schema for the MCCSAllocation class.

create_mccs_allocate(*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

CHAPTER 29

ska_oso_pdm.schemas.mccs.subarray_beam_configuration

The schemas.subarray_beam_configuration defines a Marshmallow schema that maps SKA LOW subarray beams, target to/from a JSON representation.

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

The MCCS subarray beam configuration schema section of an SKA scheduling block.

create_subarray_beam (*data*, **_)

Convert parsed JSON back into a SubarrayBeamConfiguration

Parameters

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

Returns SubarrayBeamConfiguration instance populated to match JSON

CHAPTER 30

ska_oso_pdm.schemas.mccs.target_beam_configuration

The schemas.target_beam_configuration defines a Marshmallow schema that maps SKA LOW subarray beams, target to/from a JSON representation.

class TargetBeamConfigurationSchema (**args*, ***kwargs*)

The Target beam configuration schema section of an SKA scheduling block

create_target_beam (*data*, **_)

Convert parsed JSON back into a TargetBeamConfiguration

Parameters

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

Returns TargetBeamConfiguration instance populated to match JSON

CHAPTER 31

Project description

The SKA Project Data Model (PDM) is the data model used for 'offline' observatory processes. PDM entities such as observing proposals, observing programmes, scheduling blocks, etc., capture all the information required to describe and grade an observing proposal and any associated scheduling blocks. ska-oso-pdm is a Python object model and JSON serialisation library for the PDM.

31.1 Status

The SB definition evolves from PI to PI as more elements are added to the system and greater element configuration space is exposed. Practically speaking, this library should be considered the best reference for the current state of SB design.

For examples of SBs, see the *MID JSON representation* and *LOW JSON representation*.

Historical references can be found here:

- PI3 MID SB
- PI6 MID SB
- PI10 LOW SB
- PI11 MID SB
- PI11 LOW SB

CHAPTER 32

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

C

D

E

F

G

L

M

N

P

R

S

T

U

W