developer.skatelescope.org Documentation

Release 10.2.0

Marco Bartolini

Aug 10, 2023

TABLE OF CONTENTS

CHAPTER

ONE

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

TWO

BACKGROUND

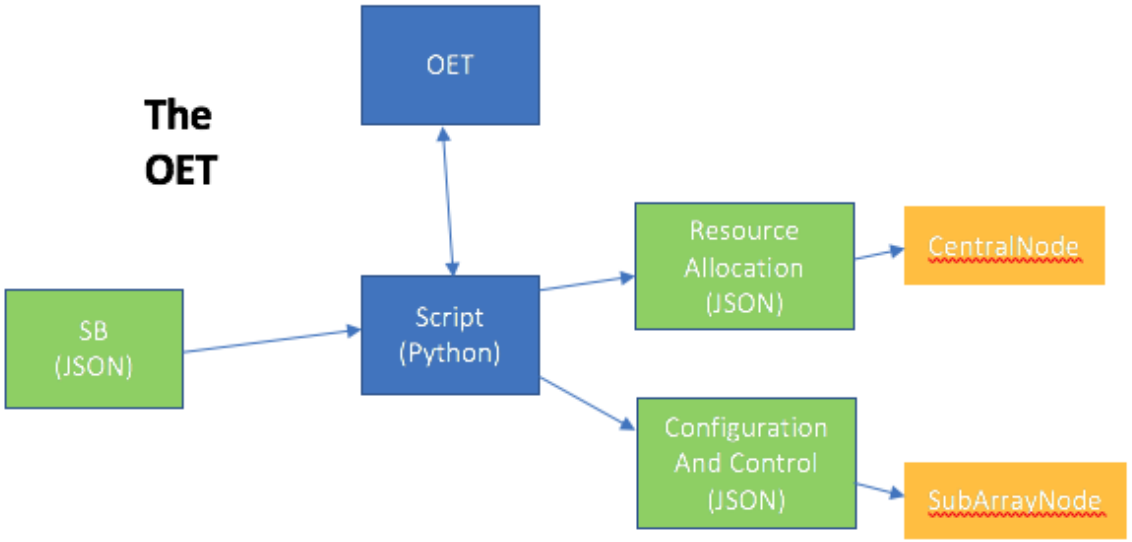
The SKA Project Data Model (PDM) defines a logical data model with entities and relationships that represent all the information required to schedule and observe a fully calibratable observation on an SKA telescope.

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.

CHAPTER

THREE

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.

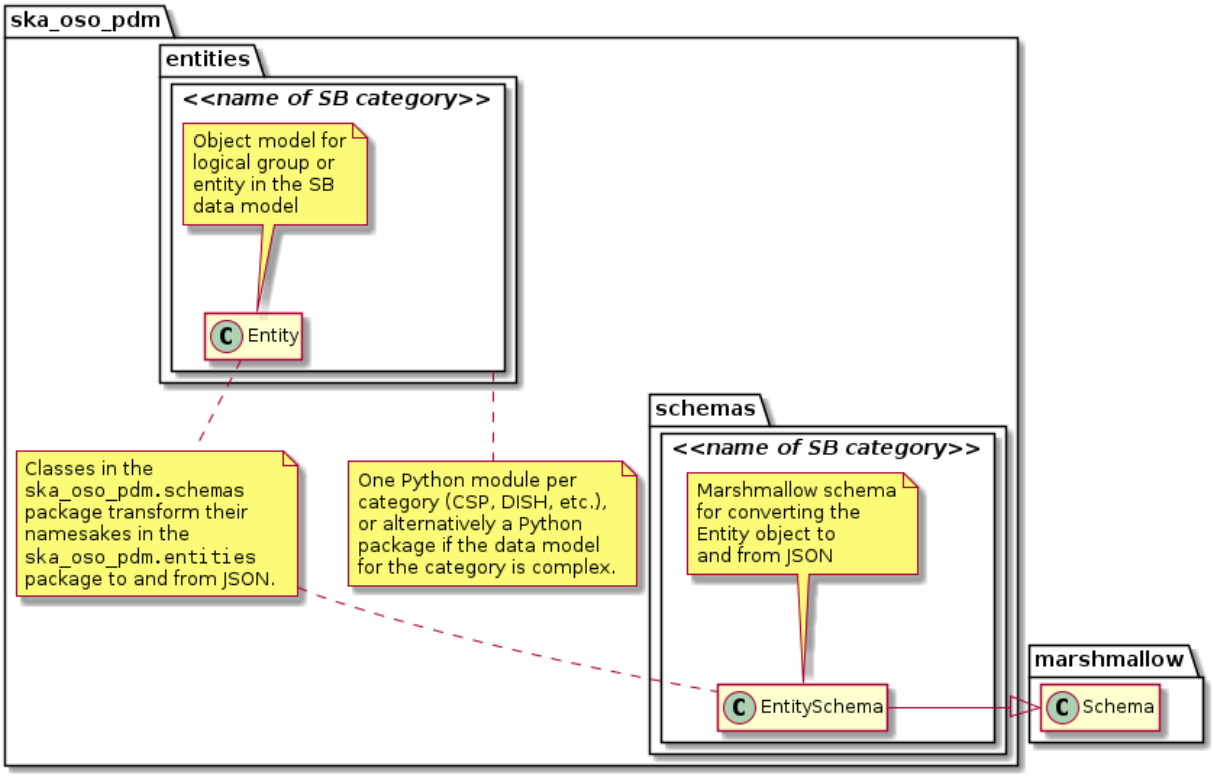


Fig. 1: Project layout and naming conventions.

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

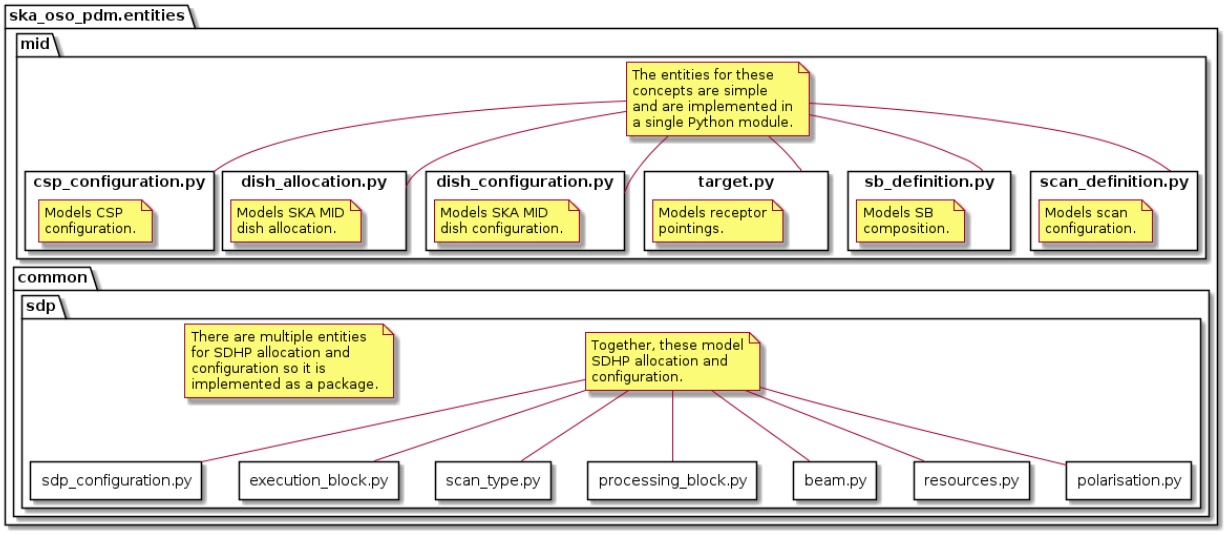


Fig. 2: Overview of PDM entities

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.

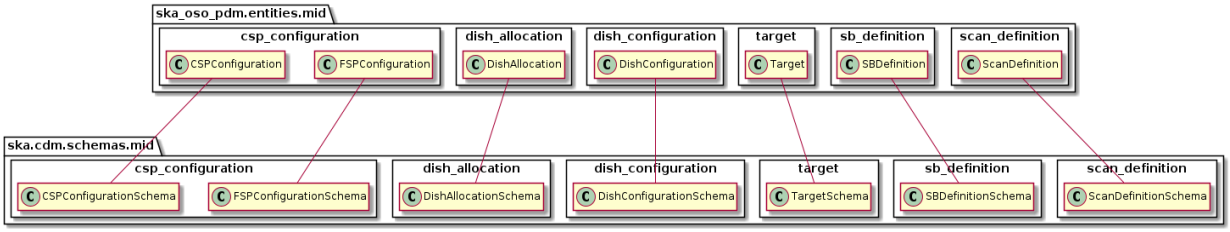


Fig. 3: Schema mapping for ska_oso_pdm.entities package

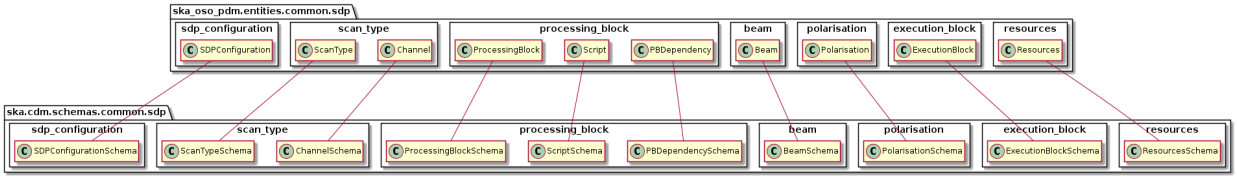


Fig. 4: Schema mapping for ska_oso_pdm.entities.sdp package

CHAPTER

FOUR

JSON REPRESENTATION

4.1 MID JSON representation

A full example of a simple MID SB serialised to JSON is given below. This SB is not self consistent (i.e. does not correctly describe an observation "end-to-end") but gives an example of the structure and possible contents of a MID SB.

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+00:00",
    "last_modified_on": "2022-03-28T15:43:53.971548+00:00",
    "last_modified_by": "Liz Bartlett"
  },
  "activities": {
    "allocate": {
      "kind": "filesystem",
      "path": "file:///path/to/allocatescript.py",
      "function_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "main": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
```

(continues on next page)

(continued from previous page)


```
          }
        }
      }
    },
    "observe": {
      "kind": "git",
      "path": "git://relative/path/to/scriptinsiderepo.py",
      "repo": "https://gitlab.com/script_repo/operational_scripts",
      "branch": "main",
      "function_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "main": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    }
  },
  "scan_definitions": [
    {
      "scan_definition_id": "calibrator scan",
      "scan_duration": 60000,
      "target": "Polaris Australis",
      "dish_configuration": "dish config 123",
      "scan_type": "calibration_B",
      "csp_configuration": "csp config 123"
    },
    {
      "scan_duration": 60000,
      "target": "M83",
      "dish_configuration": "dish config 123",
      "scan_type": "science_A",
      "scan_definition_id": "science scan",
      "csp_configuration": "csp config 123"
    }
  ],
  "scan_sequence": [
    "calibrator scan",
    "science scan",
```


(continues on next page)

(continued from previous page)

```
    "science scan",
    "calibrator scan"
  ],
  "targets": [
    {
      "target_id": "Polaris Australis",
      "pointing_pattern": {
        "active": "FivePointParameters",
        "parameters": [
          {
            "kind": "FivePointParameters",
            "offset_arcsec": 5.0
          },
          {
            "kind": "RasterParameters",
            "row_length_arcsec": 1.23,
            "row_offset_arcsec": 4.56,
            "n_rows": 2,
            "pa": 7.89,
            "unidirectional": true
          },
          {
            "kind": "StarRasterParameters",
            "row_length_arcsec": 1.23,
            "n_rows": 2,
            "row_offset_angle": 4.56,
            "unidirectional": true
          }
        ]
      },
      "reference_coordinate": {
        "kind": "equatorial",
        "ra": "21:08:47.92",
        "dec": "-88:57:22.9",
        "reference_frame": "ICRS",
        "unit": [
          "hourangle",
          "deg"
        ]
      }
    },
    {
      "target_id": "M83",
      "pointing_pattern": {
        "active": "SinglePointParameters",
        "parameters": [
          {
            "kind": "SinglePointParameters",
            "offset_x_arcsec": 0.0,
            "offset_y_arcsec": 0.0
          }
        ]
```

(continues on next page)

(continued from previous page)


```
      },
      "reference_coordinate": {
        "kind": "equatorial",
        "ra": "13:37:00.919",
        "dec": "-29:51:56.74",
        "reference_frame": "ICRS",
        "unit": [
          "hourangle",
          "deg"
        ]
      }
    }
  ],
  "sdp_configuration": {
    "execution_block": {
      "eb_id": "eb-mvp01-20200325-00001",
      "max_length": 100.0,
      "context": {
        "foo": "bar",
        "baz": 123
      },
      "beams": [
        {
          "beam_id": "vis0",
          "function": "visibilities"

        },
        {
          "beam_id": "pss1",
          "search_beam_id": 1,
          "function": "pulsar search"
        },
        {
          "beam_id": "pss2",
          "search_beam_id": 2,
          "function": "pulsar search"
        },
        {
          "beam_id": "pst1",
          "timing_beam_id": 1,
          "function": "pulsar search"
        },
        {
          "beam_id": "pst2",
          "timing_beam_id": 2,
          "function": "pulsar search"
        },
        {
          "beam_id": "vlbi",
          "vlbi_beam_id": 1,
          "function": "vlbi"
        }
```


(continues on next page)

(continued from previous page)

```
    ],
    "scan_types": [
      {
        "scan_type_id": ".default",
        "beams": [
          {
            "beam_id": "vis0",
            "channels_id": "vis_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pss1",
            "field_id": "M83",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pss2",
            "field_id": "Polaris Australis",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pst1",
            "field_id": "M83",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pst2",
            "field_id": "Polaris Australis",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "vlbi",
            "field_id": "Polaris Australis",
            "channels_id": "vlbi_channels",
            "polarisations_id": "all"
          }
        ]
      },
      {
        "scan_type_id": ".default",
        "derive_from": ".default",
        "beams": [
          {
            "beam_id": "vis0",
            "field_id": "M83"
          }
        ]
      }
```

(continues on next page)

(continued from previous page)

```
    ],
    "channels": [
      {
        "channels_id": "vis_channels",
        "spectral_windows": [
          {
            "spectral_window_id": "fsp_1_channels",
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000,
            "freq_max": 368000000,
            "link_map": [
              [
                0,
                0
              ],
              [
                200,
                1
              ],
              [
                744,
                2
              ],
              [
                944,
                3
              ]
            ]
          },
          {
            "spectral_window_id": "fsp_2_channels",
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 360000000,
            "freq_max": 368000000,
            "link_map": [
              [
                2000,
                4
              ],
              [
                2200,
                5
              ]
            ]
          },
          {
            "spectral_window_id": "zoom_window_1",
            "count": 744,
```

(continues on next page)

(continued from previous page)


```
            "start": 4000,
            "stride": 1,
            "freq_min": 360000000,
            "freq_max": 361000000,
            "link_map": [
              [
                4000,
                6
              ],
              [
                4200,
                7
              ]
            ]
          }
        ]
      },
      {
        "channels_id": "pulsar_channels",
        "spectral_windows": [
          {
            "spectral_window_id": "pulsar_fsp_channels",
            "count": 744,
            "start": 0,
            "freq_min": 350000000,
            "freq_max": 368000000
          }
        ]
      }
    ],
    "polarisations": [
      {
        "polarisations_id": "all",
        "corr_type": [
          "XX",
          "XY",
          "YY",
          "YX"
        ]
      }
    ]
  },
  "processing_blocks": [
    {
      "pb_id": "pb-mvp01-20200325-00001",
      "sbi_ids": [
        "sbi-mvp01-20200325-00001"
      ],
      "script": {
        "version": "0.1.0",
        "name": "vis_receive",
        "kind": "realtime"
```


(continues on next page)

(continued from previous page)


```
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00002",
      "sbi_ids": [
        "sbi-mvp01-20200325-00001"
      ],
      "script": {
        "version": "0.1.0",
        "name": "test_realtime",
        "kind": "realtime"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00003",
      "sbi_ids": [
        "sbi-mvp01-20200325-00001"
      ],
      "script": {
        "version": "0.1.0",
        "name": "ical",
        "kind": "batch"
      },
      "parameters": {},
      "dependencies": [
        {
          "pb_id": "pb-mvp01-20200325-00001",
          "kind": [
            "visibilities"
          ]
        }
      ]
    },
    {
      "pb_id": "pb-mvp01-20200325-00004",
      "sbi_ids": [
        "sbi-mvp01-20200325-00001"
      ],
      "script": {
        "version": "0.1.0",
        "name": "dpreb",
        "kind": "batch"
      },
      "parameters": {},
      "dependencies": [
        {
          "pb_id": "pb-mvp01-20200325-00003",
          "kind": [
            "calibration"
          ]
```


(continues on next page)

(continued from previous page)


```
        }
      ]
    }
  ],
  "resources": {
    "csp_links": [
      1,
      2,
      3,
      4
    ],
    "receptors": [
      "FS4",
      "FS8",
      "FS16",
      "FS17",
      "FS22",
      "FS23",
      "FS30",
      "FS31",
      "FS32",
      "FS33",
      "FS36",
      "FS52",
      "FS56",
      "FS57",
      "FS59",
      "FS62",
      "FS66",
      "FS69",
      "FS70",
      "FS72",
      "FS73",
      "FS78",
      "FS80",
      "FS88",
      "FS89",
      "FS90",
      "FS91",
      "FS98",
      "FS108",
      "FS111",
      "FS132",
      "FS144",
      "FS146",
      "FS158",
      "FS165",
      "FS167",
      "FS176",
      "FS183",
      "FS193",
      "FS200",
```


(continues on next page)

(continued from previous page)


```
        "FS345",
        "FS346",
        "FS347",
        "FS348",
        "FS349",
        "FS350",
        "FS351",
        "FS352",
        "FS353",
        "FS354",
        "FS355",
        "FS356",
        "FS429",
        "FS430",
        "FS431",
        "FS432",
        "FS433",
        "FS434",
        "FS465",
        "FS466",
        "FS467",
        "FS468",
        "FS469",
        "FS470"
      ],
      "receive_nodes": 10
    }
  },
  "csp_configurations": [
    {
      "config_id": "csp config 123",
      "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,
```


(continues on next page)

(continued from previous page)


```
                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
          }
        ]
      }
    }
  ],
  "dish_allocations": {
    "receptor_ids": [
      "0001",
      "0002"
    ]
  },
  "dish_configurations": [
    {
      "dish_configuration_id": "dish config 123",
      "receiver_band": "5a"
    }
  ]
}
```

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+00:00"
  },
  "activities": {
    "allocate": {
      "kind": "filesystem",
      "path": "/path/to/allocatescript.py",
      "function_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "main": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    },
    "observe": {
      "kind": "git",
      "path": "/relative/path/to/scriptinsiderepo.py",
      "repo": "https://gitlab.com/script_repo/operational_scripts",
      "branch": "main",
```

(continues on next page)

(continued from previous page)


```
      "commit": "d234c257dadd18b3edcd990b8194c6ad94fc278a",
      "function_args": {
        "init": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        },
        "main": {
          "args": [
            "posarg1",
            "posarg2"
          ],
          "kwargs": {
            "argname": "argval"
          }
        }
      }
    }
  },
  "scan_definitions": [
    {
      "scan_definition_id": "sbi-mvp01-20220328-00001",
      "scan_duration": 64000,
      "target_beam_configurations": [
        "target #1 with beam A config 1"
      ],
      "target": "target #1"
    },
    {
      "scan_definition_id": "sbi-mvp01-20220328-00002",
      "scan_duration": 64000,
      "target_beam_configurations": [
        "target #2 with beam A config 1"
      ],
      "target": "target #2"
    }
  ],
  "scan_sequence": [
    "sbi-mvp01-20220328-00001",
    "sbi-mvp01-20220328-00002",
    "sbi-mvp01-20220328-00002",
    "sbi-mvp01-20220328-00001"
  ],
  "targets": [
    {
      "target_id": "target #1",
      "pointing_pattern": {
        "active": "SinglePointParameters",
```


(continues on next page)

(continued from previous page)


```
        "parameters": [
          {
            "kind": "SinglePointParameters",
            "offset_x_arcsec": 0.0,
            "offset_y_arcsec": 0.0
          }
        ]
      },
      "reference_coordinate": {
        "kind": "horizontal",
        "az": 180.0,
        "el": 45.0,
        "reference_frame": "ALTAZ",
        "unit": ["deg"]
      }
    },
    {
      "target_id": "target #2",
      "pointing_pattern": {
        "active": "SinglePointParameters",
        "parameters": [
          {
            "kind": "SinglePointParameters",
            "offset_x_arcsec": 0.0,
            "offset_y_arcsec": 0.0
          }
        ]
      },
      "reference_coordinate": {
        "kind": "horizontal",
        "az": 180.0,
        "el": 85.0,
        "reference_frame": "ALTAZ",
        "unit": ["deg"]
      }
    }
  ],
  "mccs_allocation": {
    "subarray_beam_ids": [
      "beam A"
    ],
    "station_ids": [
      [
        1,
        2
      ]
    ],
    "channel_blocks": [
      1
    ]
  },
  "target_beam_configurations": [
```


(continues on next page)

(continued from previous page)


```
    {
      "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

FIVE

EXECUTION BLOCKS

SKA needs a way to link data to the observations that create the data. To summarise the purpose of Execution Blocks (EBs), EBs exist to link SDP data to the observing session that created the data, be it an observation driven by the OET using an SB Instance or an interactive session where control was exerted via other interface.

The Execution Block defined in the PDM is used to record references to the OSO entities whose execution created the data (Scheduling Block Instances, Scheduling Block Definitions, etc.), the sequence of commands and responses that led to the data, and to link to the SBD Status entity, which is the OSO entity that defines the current state of the SB Definition within the SB Definition lifecycle.

A current example of the Execution Block json, with examples of handling success and error responses from commands, can be seen below:


```
{
  "interface": "https://schema.skao.int/ska-oso-pdm-eb/0.1",
  "eb_id": "eb-mvp01-20220923-00001",
  "telescope": "ska_mid",
  "sbd_id": "sbd-mvp01-20220923-00001",
  "sbd_version": 1,
  "metadata": {
    "version": 1,
    "created_by": "TestUser",
    "created_on": "2022-09-23T15:43:53.971548+00:00",
    "last_modified_on": "2022-09-23T15:43:53.971548+00:00",
    "last_modified_by": "TestUser"
  },
  "request_responses": [
    {
      "request": "ska_oso_scripting.functions.devicecontrol.release_all_resources",
      "request_args": "{'args': None, 'kwargs': {'subarray_id': '1'}}",
      "status": "OK",
      "response": {
        "result": "this is a result"
      },
      "request_sent_at": "2022-09-23T15:43:53.971548+00:00",
      "response_received_at": "2022-09-23T15:43:53.971548+00:00"
    },
    {
      "request": "ska_oso_scripting.functions.devicecontrol.scan",
      "status": "ERROR",
      "error": {
        "detail": "this is an error"
```


(continues on next page)

(continued from previous page)

```
      },
      "request_sent_at": "2022-09-23T15:43:53.971548+00:00"
    }
  ]
}
```

CHAPTER

SIX

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.

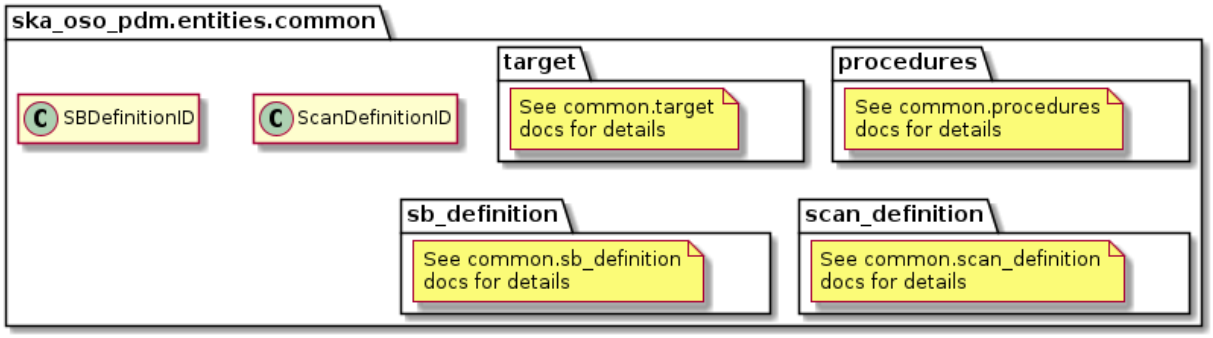


Fig. 1: Class diagram for the __init__ module

CHAPTER

SEVEN

SKA_OSO_PDM.ENTITIES.COMMON.TARGET

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

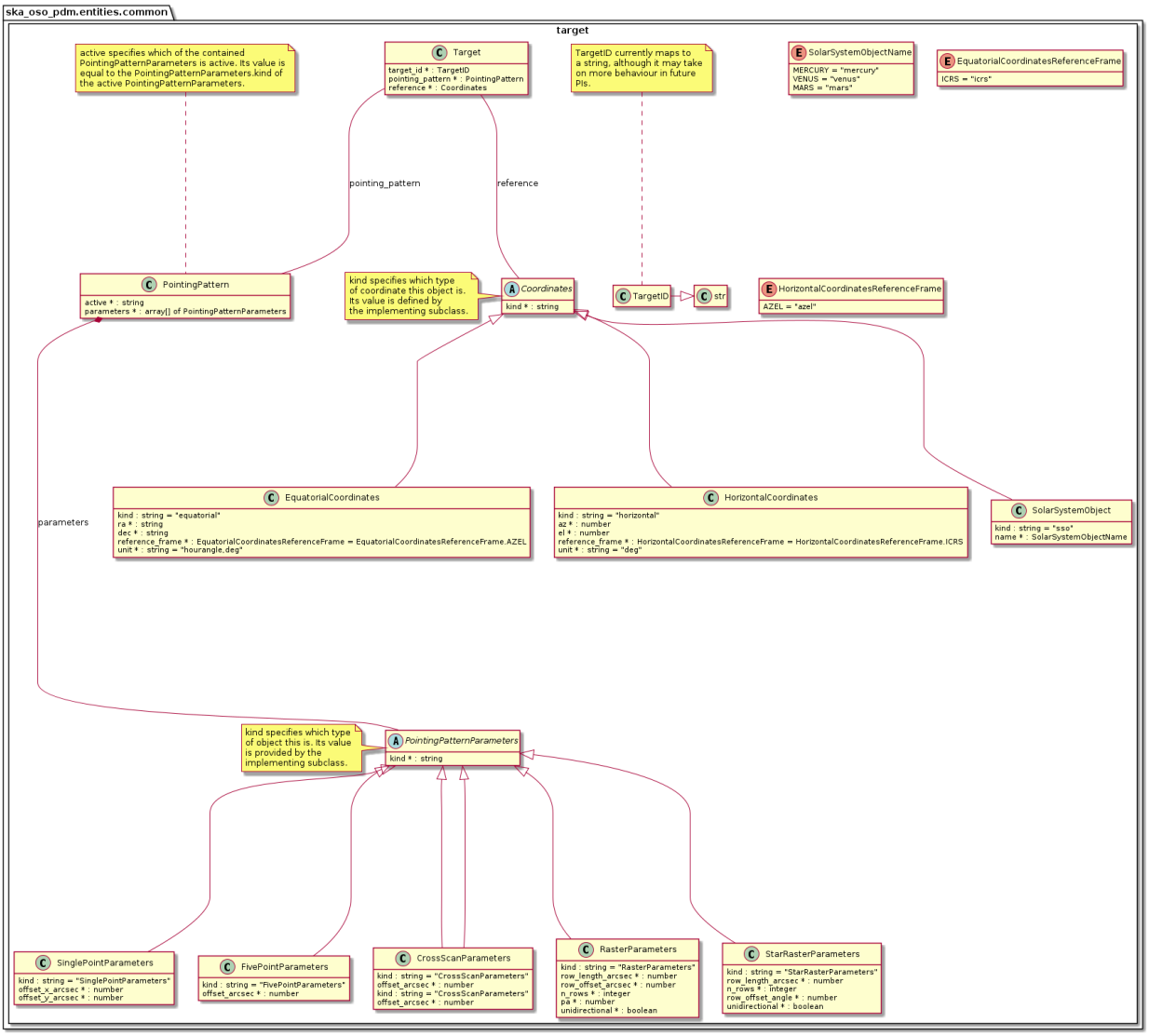


Fig. 1: Class diagram for the target module

An example serialisation of this model to JSON is shown below. This describes two target objects: one, a five-point observation centred on Polaris Australis using 5 arcsec offsets between individual pointings, and two, a single pointing observation centred on M83. In both cases, the coordinates are defined using equatorial coordinates using an ICRS reference frame.

```
# JSON modelled specifically by target module
...
"targets": [
  {
    "target_id": "Polaris Australis",
    "pointing_pattern": {
      "active": "FivePointParameters",
      "parameters": [
        {
          "kind": "FivePointParameters",
          "offset_arcsec": 5.0
        }
      ]
    },
    "reference_coordinate": {
      "kind": "equatorial",
      "ra": "21:08:47.92",
      "dec": "-88:57:22.9",
      "reference_frame": "ICRS",
      "unit": ["hourangle","deg"]
    }
  },
  {
    "target_id": "M83",
    "pointing_pattern": {
      "active": "SinglePointParameters",
      "parameters": [
        {
          "kind": "SinglePointParameters",
          "offset_x_arcsec": 0.0,
          "offset_y_arcsec": 0.0
        }
      ]
    },
    "reference_coordinate": {
      "kind": "equatorial",
      "ra": "13:37:00.919",
      "dec": "-29:51:56.74",
      "reference_frame": "ICRS",
      "unit": ["hourangle","deg"]
    }
  }
],
...
```

Another JSON example defining one drift scan target with position specified as azimuth and elevation is shown below.

```
...
"targets": [
```

(continues on next page)

(continued from previous page)

```
  {
    "target_id": "target #1",
    "pointing_pattern": {
      "active": "SinglePointParameters",
      "parameters": [
        {
          "kind": "SinglePointParameters",
          "offset_x_arcsec": 0.0,
          "offset_y_arcsec": 0.0
        }
      ]
    },
    "reference_coordinate": {
      "kind": "horizontal",
      "az": 180.0,
      "el": 45.0,
      "reference_frame": "ALTAZ",
      "unit": ["hourangle","deg"]
    }
  }
],
...
```

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

TargetID

alias of `str`

CHAPTER

EIGHT

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.

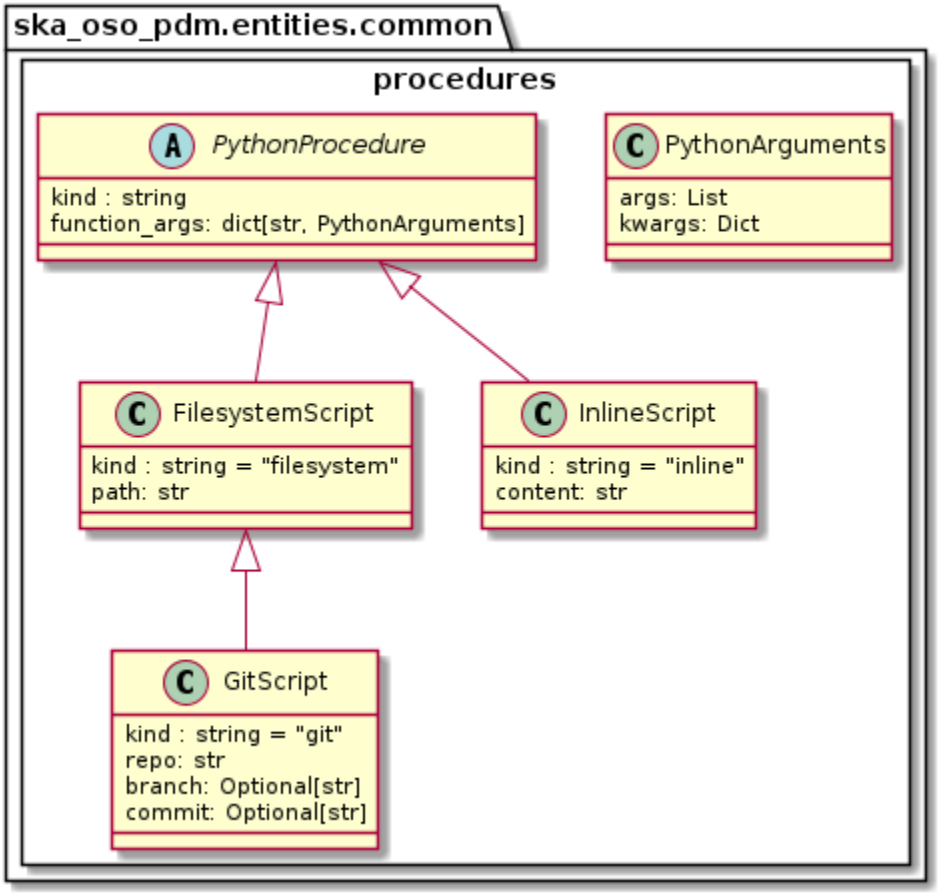


Fig. 1: Class diagram for the procedure module

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

```
# JSON modelled specifically by procedures
...
```

(continues on next page)

(continued from previous page)

```
"activities":{
  "allocate": {
    "procedure_type": "filesystemscript",
    "path": "/path/to/allocatescript.py",
    "function_args": {
      "init": {"args": ["posarg1","posarg2"],
        "kwargs": {"argname": "argval"}
      },
      "main": {
        "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",
    "function_args": {
      "init": {"args": ["posarg1","posarg2"],
        "kwargs": {"argname": "argval"}
      },
      "main": {
        "args": ["posarg1","posarg2"],
        "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 FilesystemScript(*path: str*, *function_args: Optional[Dict[str, PythonArguments]] = None*)

Represents an FilesystemScript to be run as an activity in an SKA scheduling block.

class GitScript(*repo: str*, *path: str*, *branch: Optional[str] = None*, *commit: Optional[str] = None*, *function_args: Optional[Dict[str, PythonArguments]] = None*)

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

class InlineScript(*content: str*, *function_args: Optional[Dict[str, PythonArguments]] = None*)

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

CHAPTER

NINE

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,
    "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": ...
```

(continues on next page)

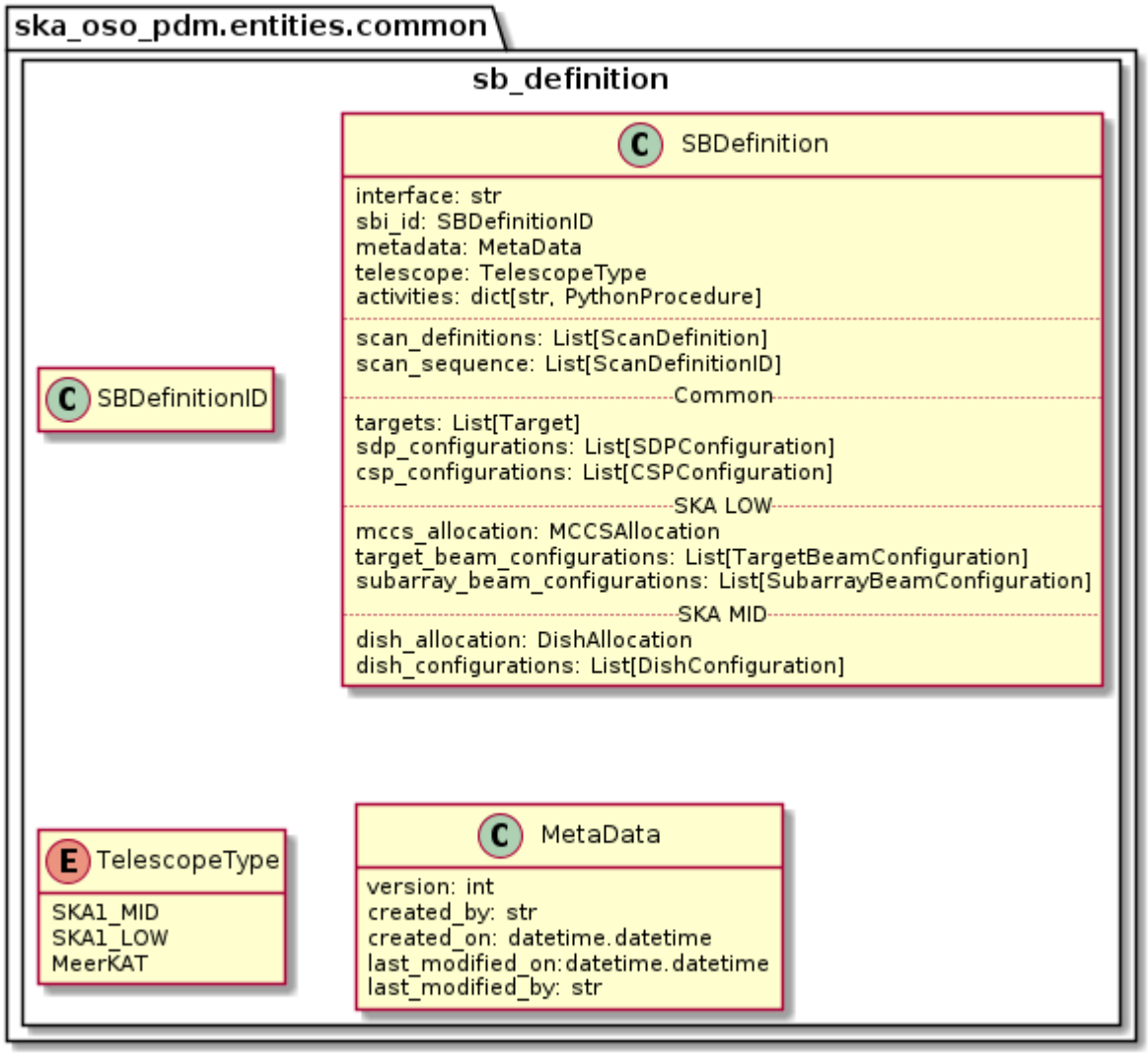


Fig. 1: Class diagram for the sb_definition module

(continued from previous page)

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

MetaData Class

class SBDefinition(*, *interface: Optional[str] = 'https://schema.skao.int/ska-oso-pdm-sbd/0.1'*, *sbd_id: Optional[str] = None*, *telescope: Optional[TelescopeType] = None*, *metadata: Optional[MetaData] = None*, *activities: Optional[Dict[str, PythonProcedure]] = None*, *targets: Optional[List[Target]] = None*, *scan_definitions: Optional[List[ScanDefinition]] = None*, *scan_sequence: Optional[List[str]] = None*, *sdp_configuration: Optional[SDPConfiguration] = None*, *dish_configurations: Optional[List[DishConfiguration]] = None*, *csp_configurations: Optional[List[CSPConfiguration]] = None*, *dish_allocations: Optional[DishAllocation] = None*, *mccs_allocation: Optional[MCCSAllocation] = None*, *subarray_beam_configurations: Optional[List[SubarrayBeamConfiguration]] = None*, *target_beam_configurations: Optional[List[TargetBeamConfiguration]] = None*)

SKA scheduling block

class TelescopeType(*value*)

TelescopeType represents which telescope is being used

CHAPTER

TEN

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.

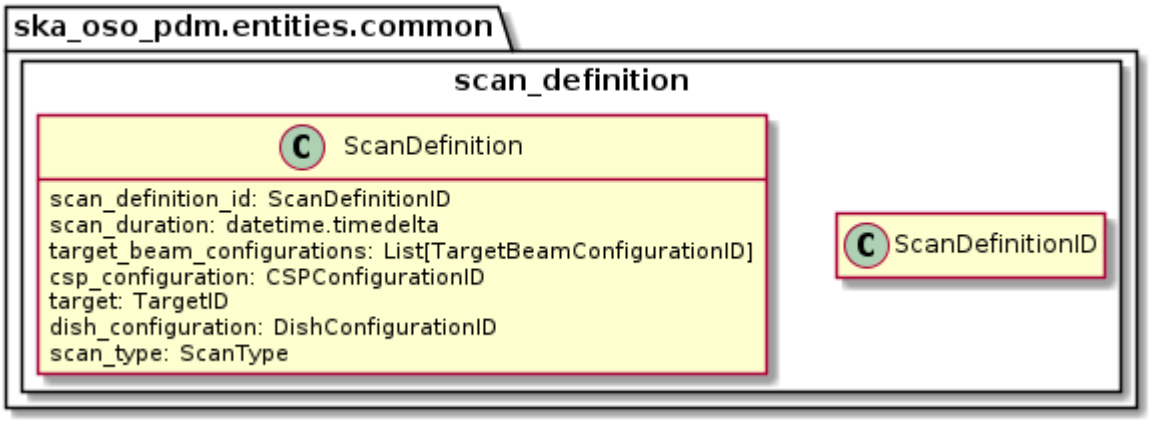


Fig. 1: Class diagram for the scan_definition module

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,
    "target": "calibrator target",
    "dish_configuration": "dish config 123",
    "scan_type": "calibration_B",
    "csp_configuration": "csp-mvp01-20220329-00001"
  },
  {
    "scan_duration": 60000,
    "target": "science target",
    "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: timedelta*, *target_id: Optional[str] = None*, *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 `str`

CHAPTER

ELEVEN

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.

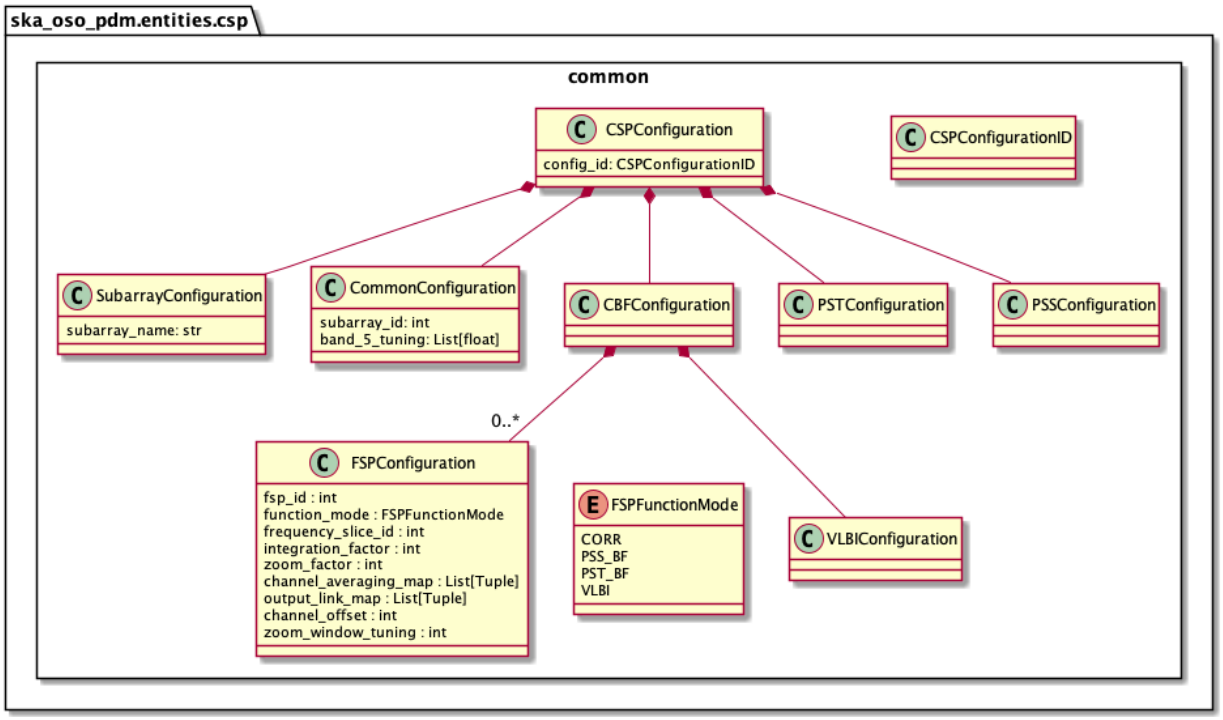


Fig. 1: Class diagram for the common module within csp

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": {
```

(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,
          "zoom_window_tuning": 650000
        }]
    }
  }
],
  ...
```

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

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

Class to hold all FSP and VLBI configurations.

class CSPConfiguration(*config_id: Optional[str] = None*, *subarray_config: Optional[SubarrayConfiguration] = None*, *common_config: Optional[CommonConfiguration] = None*, *cbf_config: Optional[CBFConfiguration] = None*, *pst_config: Optional[PSTConfiguration] = None*, *pss_config: Optional[PSSConfiguration] = None*)

Class to hold all CSP configuration.

CSPConfigurationID

alias of `str`

class CommonConfiguration(*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 SubarrayConfiguration(*subarray_name: str*)

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

CHAPTER

TWELVE

SKA_OSO_PDM.ENTITIES.SDP

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

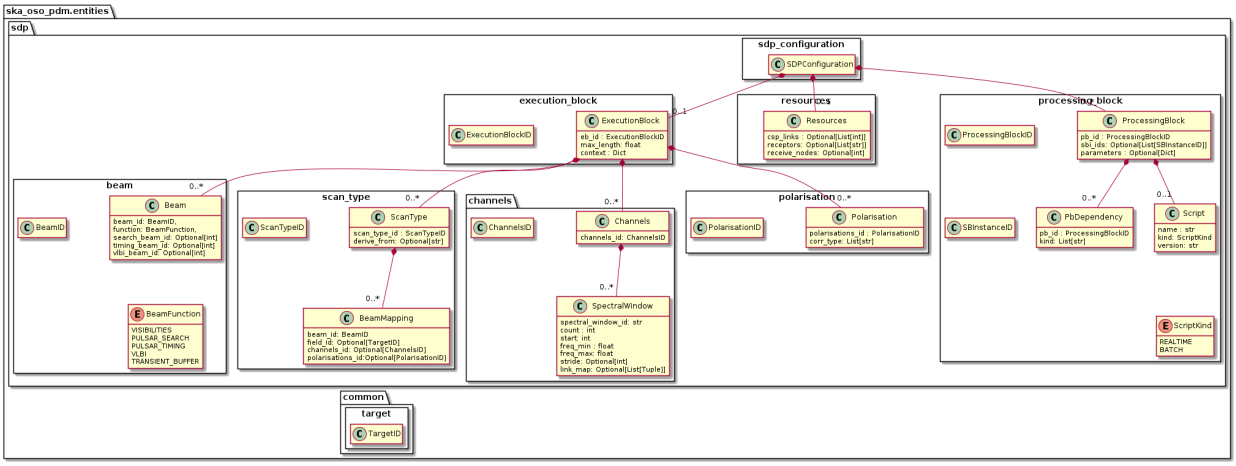


Fig. 1: Class diagram for the sdp package

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

```
# JSON modelled specifically by entities in the sdp package
...
"sdp_configuration": {
  "execution_block": {
    "eb_id": "eb-mvp01-20200325-00001",
    "max_length": 100,
    "context": {"foo": "bar", "baz": 123},
    "beams": [
      {
        "beam_id": "vis0",
        "function": "visibilities"
      },
      {
        "beam_id": "pss1",
        "search_beam_id": 1,
        "function": "pulsar search"
      },
      {
```

(continues on next page)

(continued from previous page)


```
        "beam_id": "pss2",
        "search_beam_id": 2,
        "function": "pulsar search"
      },
      {
        "beam_id": "pst1",
        "timing_beam_id": 1,
        "function": "pulsar search"
      },
      {
        "beam_id": "pst2",
        "timing_beam_id": 2,
        "function": "pulsar search"
      },
      {
        "beam_id": "vlbi",
        "vlbi_beam_id": 1,
        "function": "vlbi"
      }
    ],
    "scan_types": [
      {
        "scan_type_id": ".default",
        "beams": [
          {
            "beam_id": "vis0",
            "channels_id": "vis_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pss1",
            "field_id": "pss_field_0",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pss2",
            "field_id": "pss_field_1",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pst1",
            "field_id": "pst_field_0",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
          },
          {
            "beam_id": "pst2",
            "field_id": "pst_field_1",
            "channels_id": "pulsar_channels",
            "polarisations_id": "all"
```


(continues on next page)

(continued from previous page)

```
        },
        {
          "beam_id": "vlbi",
          "field_id": "vlbi_field",
          "channels_id": "vlbi_channels",
          "polarisations_id": "all"
        }
      ]
    },
    {
      "scan_type_id": ".default",
      "derive_from": ".default",
      "beams": [
        {
          "beam_id": "vis0",
          "field_id": "field_a"
        }
      ]
    }
  ],
  "channels": [
    {
      "channels_id": "vis_channels",
      "spectral_windows": [
        {
          "spectral_window_id": "fsp_1_channels",
          "count": 744,
          "start": 0,
          "stride": 2,
          "freq_min": 350000000,
          "freq_max": 368000000,
          "link_map": [[0,0],[200,1],[744,2],[944,3]]
        },
        {
          "spectral_window_id": "fsp_2_channels",
          "count": 744,
          "start": 2000,
          "stride": 1,
          "freq_min": 360000000,
          "freq_max": 368000000,
          "link_map": [[2000,4],[2200,5]]
        },
        {
          "spectral_window_id": "zoom_window_1",
          "count": 744,
          "start": 4000,
          "stride": 1,
          "freq_min": 360000000,
          "freq_max": 361000000,
          "link_map": [[4000,6],[4200,7]]
        }
      ]
```

(continues on next page)

(continued from previous page)


```
      },
      {
        "channels_id": "pulsar_channels",
        "spectral_windows": [
          {
            "spectral_window_id": "pulsar_fsp_channels",
            "count": 744,
            "start": 0,
            "freq_min": 350000000,
            "freq_max": 368000000
          }
        ]
      }
    ],
    "polarisations": [
      {
        "polarisations_id": "all",
        "corr_type": ["XX","XY","YY","YX"]
      }
    ]
  },
  "processing_blocks": [
    {
      "pb_id": "pb-mvp01-20200325-00001",
      "sbi_ids": ["sbi-mvp01-20200325-00001"],
      "script": {
        "version": "0.1.0",
        "name": "vis_receive",
        "kind": "realtime"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00002",
      "sbi_ids": ["sbi-mvp01-20200325-00001"],
      "script": {
        "version": "0.1.0",
        "name": "test_realtime",
        "kind": "realtime"
      },
      "parameters": {}
    },
    {
      "pb_id": "pb-mvp01-20200325-00003",
      "sbi_ids": ["sbi-mvp01-20200325-00001"],
      "script": {
        "version": "0.1.0",
        "name": "ical",
        "kind": "batch"
      },
      "parameters": {},
      "dependencies": [
```


(continues on next page)

(continued from previous page)

```
        {
          "pb_id": "pb-mvp01-20200325-00001",
          "kind": ["visibilities"]
        }
      ]
    },
    {
      "pb_id": "pb-mvp01-20200325-00004",
      "sbi_ids": ["sbi-mvp01-20200325-00001"],
      "script": {
        "version": "0.1.0",
        "name": "dpreb",
        "kind": "batch"
      },
      "parameters": {},
      "dependencies": [
        {
          "pb_id": "pb-mvp01-20200325-00003",
          "kind": ["calibration"]
        }
      ]
    }
  ],
  "resources": {
    "csp_links": [1,2,3,4],
    "receptors": [
      "FS4",
      "FS8",
      "FS16",
      "FS17",
      "FS22",
      "FS23",
      "FS30",
      "FS31",
      "FS32",
      "FS33",
      "FS36",
      "FS52",
      "FS56",
      "FS57",
      "FS59",
      "FS62",
      "FS66",
      "FS69",
      "FS70",
      "FS72",
      "FS73",
      "FS78",
      "FS80",
      "FS88",
      "FS89",
      "FS90",
```

(continues on next page)

(continued from previous page)

```
      "FS91",
      "FS98",
      "FS108",
      "FS111",
      "FS132",
      "FS144",
      "FS146",
      "FS158",
      "FS165",
      "FS167",
      "FS176",
      "FS183",
      "FS193",
      "FS200",
      "FS345",
      "FS346",
      "FS347",
      "FS348",
      "FS349",
      "FS350",
      "FS351",
      "FS352",
      "FS353",
      "FS354",
      "FS355",
      "FS356",
      "FS429",
      "FS430",
      "FS431",
      "FS432",
      "FS433",
      "FS434",
      "FS465",
      "FS466",
      "FS467",
      "FS468",
      "FS469",
      "FS470"
    ],
    "receive_nodes": 10
  }
}
...
```

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(*, *execution_block: Optional[ExecutionBlock] = None*, *resources: Optional[Resources] = None*, *processing_blocks: Optional[List[ProcessingBlock]] = None*)

SDPConfiguration captures the SDP resources and pipeline configuration required to process an execution block.

The ska_oso_pdm.entities.common.sdp.sdp_configuration module defines a Python object model for the Execution

Block JSON string passed to CentralNode.AssignResources.

class ExecutionBlock(*, *eb_id: str*, *max_length: float*, *context: Optional[Dict] = None*, *beams: List[*Beam*]*, *scan_types: List[*ScanType*]*, *channels: List[*Channels*]*, *polarisations: List[*Polarisation*]*)

ExecutionBlock captures the Execution Block resources and pipeline configuration required to process an execution block.

ExecutionBlockID

alias of `str`

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

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

Class to hold Dependencies for ProcessingBlock

class ProcessingBlock(*, *pb_id: str*, *script:* Script, *sbi_ids: Optional[List[str]] = None*, *parameters: Optional[Dict] = None*, *dependencies: Optional[List[*PbDependency*]] = None*)

Class to hold ProcessingBlock configuration

ProcessingBlockID

alias of `str`

class Script(*, *name: str*, *kind:* ScriptKind, *version: str*)

Class to hold Script for ProcessingBlock

class ScriptKind(*value*)

Enumeration class to hold the kind of processing script

The entities.sdp.resources module defines a Python representation of a Resources for SDP configuration.

class Resources(*, *csp_links: Optional[List[int]] = None*, *receptors: Optional[List[str]] = None*, *receive_nodes: Optional[int] = None*)

Class to hold SDP Resources

The entities.sdp.beam module defines a Python representation of beams for SDP configuration.

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

Class that defines an SDP Beam configuration.

class BeamFunction(*value*)

Enumeration of possible functions for an SDP Beam.

BeamID

alias of `str`

The entities.sdp.channels module defines a Python representation of a channel configuration for SDP configuration.

class Channels(*, *channels_id: str*, *spectral_windows: List[*SpectralWindow*]*)

Class to hold Channel configuration

ChannelsID

alias of `str`

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

Class to hold Spectral Windows configuration

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

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

Class to hold mapping of beam parameters to scans

class ScanType(*, *scan_type_id: str*, *derive_from: Optional[str] = None*, *beams: Optional[List[BeamMapping]] = None*)

Class to hold ScanType configuration

ScanTypeID

alias of `str`

The entities.sdp.polarisation module defines a Python representation of polarisations for SDP configuration.

class Polarisation(*, *polarisations_id: str*, *corr_type: List[str]*)

Class that defines an SDP Polarisation configuration

PolarisationID

alias of `str`

CHAPTER

THIRTEEN

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.

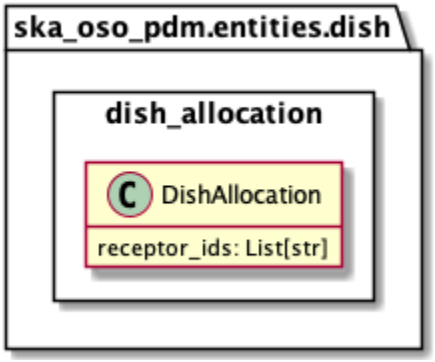


Fig. 1: Class diagram for the dish_allocation module

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

FOURTEEN

SKA_OSO_PDM.ENTITIES.DISH.DISH_CONFIGURATION

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

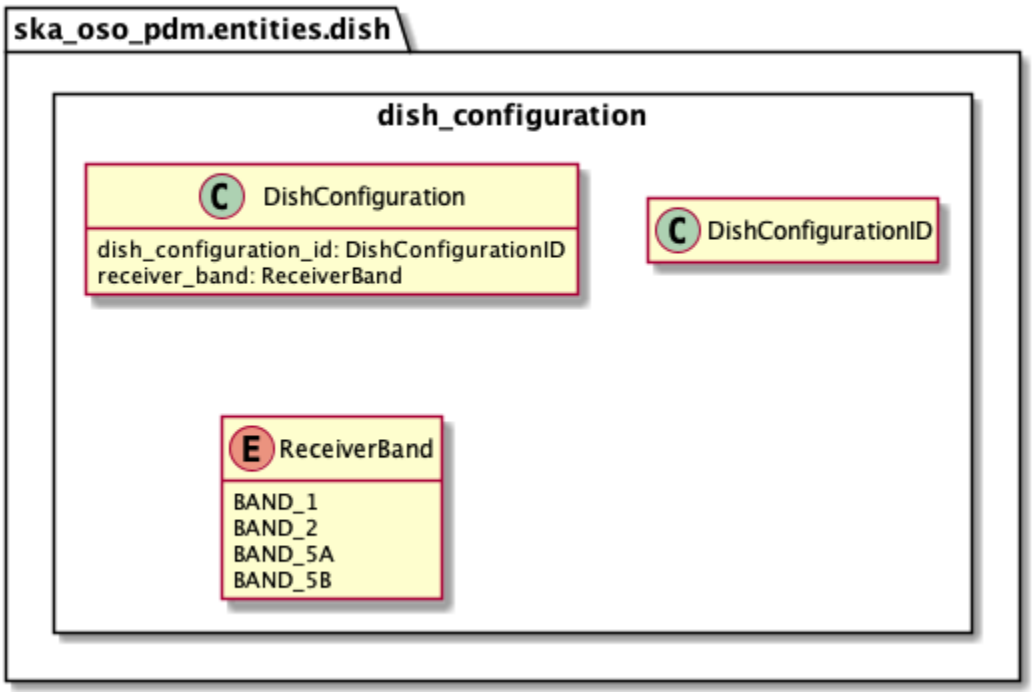


Fig. 1: Class diagram for the dish_configuration module

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.

class DishConfiguration(*dish_configuration_id: str*, *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.

DishConfigurationID

alias of `str`

class ReceiverBand(*value*)

ReceiverBand is an enumeration of SKA MID receiver bands.

CHAPTER

FIFTEEN

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.

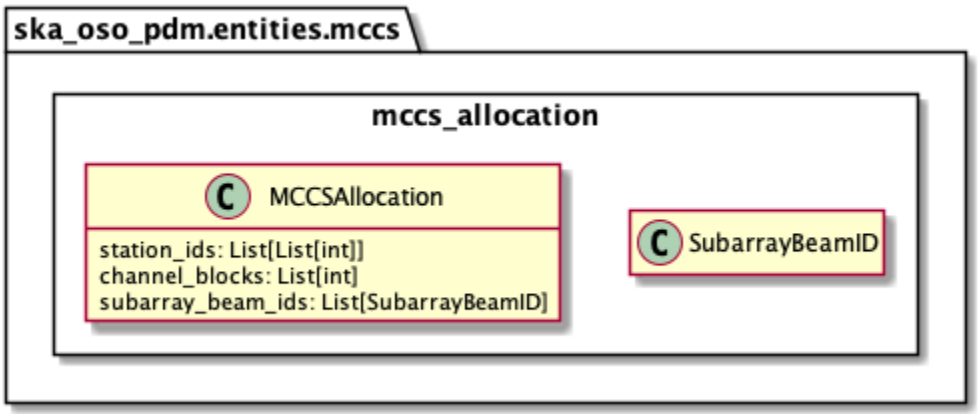


Fig. 1: Class diagram for the mccs_allocation module

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 `str`

CHAPTER

SIXTEEN

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.

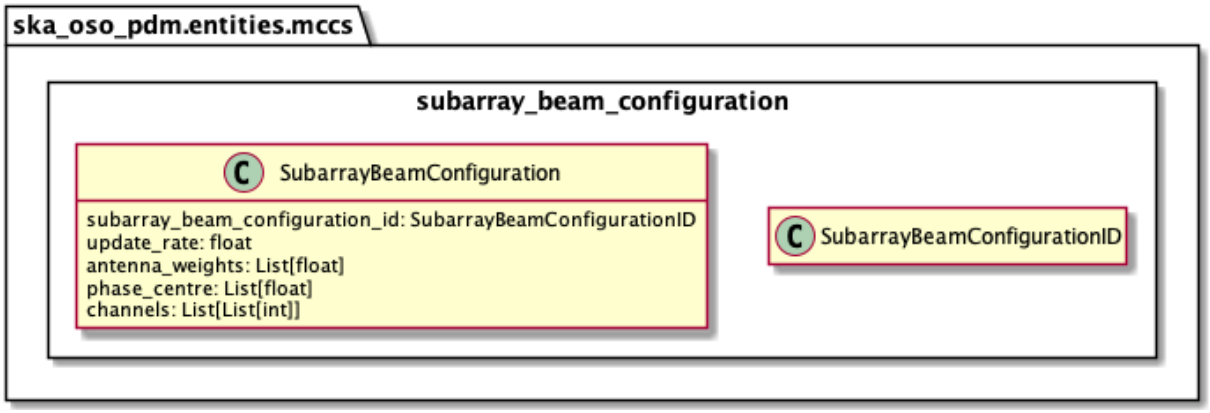


Fig. 1: Class diagram for the subarray_beam_configuration module

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]
    ]
},
...
```

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 `str`

CHAPTER

SEVENTEEN

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.

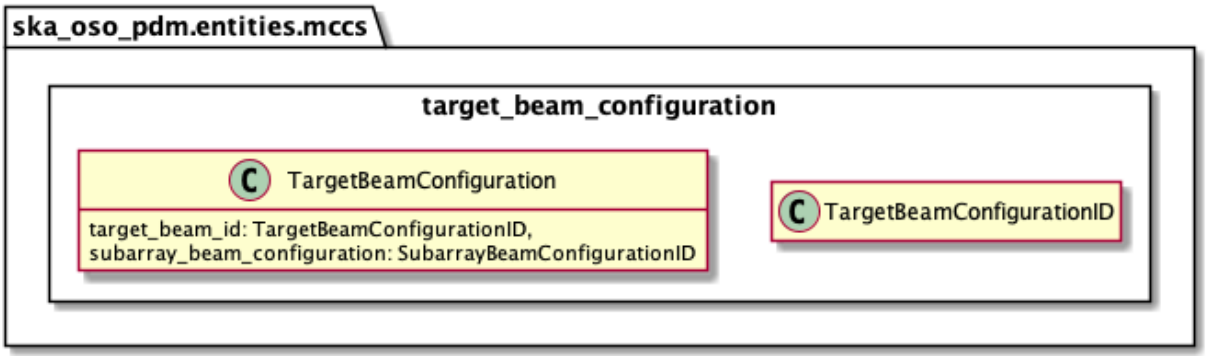


Fig. 1: Class diagram for the target_beam_configuration module

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 `str`

CHAPTER

EIGHTEEN

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

NINETEEN

SKA_OSO_PDM.SCHEMAS.SHARED

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

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

Field that serialises a list to a dict, with the specified attribute acting as key.

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

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

CHAPTER

TWENTY

SKA_OSO_PDM.SCHEMAS.COMMON

CHAPTER

TWENTYONE

SKA_OSO_PDM.SCHEMAS.COMMON.TARGET

The schemas.common.target defines Marshmallow schema that map the target pointing section of an SKA scheduling block to/from JSON.

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

Marshmallow schema for handling polymorphic coordinates classes.

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

Marshmallow schema for converting a CrossScanParameters to/from JSON

make_crossscanparameters(*data*, **_)

Convert parsed JSON back into a CrossScanParameters object

Parameters

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

Returns
CrossScanParameters instance populated to match JSON

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

Convert an EquatorialCoordinates to/from JSON.

make_equatorialcoordinates(*data*, **_)

Convert parsed JSON back into an EquatorialCoordinates object.

Parameters

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

Returns
EquatorialCoordinates instance populated to match JSON

reference_frame

alias of `EquatorialCoordinatesReferenceFrame`

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

Marshmallow schema for converting a FivePointParameters to/from JSON

make_fivepointparameters(*data*, **_)

Convert parsed JSON back into a FivePointParameters object

Parameters

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

Returns
FivePointParameters instance populated to match JSON

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

Marshmallow schema to convert a HorizontalCoordinates to/from JSON.

make_horizontalcoordinates(*data*, **_)

Convert parsed JSON back into a HorizontalCoordinates object.

Parameters

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

Returns
HorizontalCoordinates instance populated to match JSON

reference_frame

alias of HorizontalCoordinatesReferenceFrame

class JsonEquatorialCoordinate(*kind*, *ra*, *dec*, *reference_frame*, *unit*)

property dec

Alias for field number 2

property kind

Alias for field number 0

property ra

Alias for field number 1

property reference_frame

Alias for field number 3

property unit

Alias for field number 4

class JsonHorizontalCoordinate(*kind*, *az*, *el*, *reference_frame*, *unit*)

property az

Alias for field number 1

property el

Alias for field number 2

property kind

Alias for field number 0

property reference_frame

Alias for field number 3

property unit

Alias for field number 4

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

Marshmallow schema for handling polymorphic pointing pattern parameter classes.

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

Marshallow schema for converts a PointingPattern to/from JSON.

make_pointingpattern(*data*, **_)

Convert parsed JSON back into a PointingPattern object.

Parameters

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

Returns

PointingPattern instance populated to match JSON

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

Marshmallow schema for converting a RasterParameters to/from JSON

make_rasterparameters(*data*, **_)

Convert parsed JSON back into a RasterParameters object

Parameters

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

Returns

RasterParameters instance populated to match JSON

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

Marshmallow schema for converting a SinglePointParameters to/from JSON

make_singlepointparameters(*data*, **_)

Convert parsed JSON back into a SinglePointParameters object

Parameters

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

Returns

SinglePointParameters instance populated to match JSON

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

Schema for marshalling SolarSystemObject coordinates to/from JSON

create_instance(*data*, **_)

Convert parsed JSON back into a SolarSystemObject

Parameters

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

- **_** – kwargs passed by Marshmallow

Returns
instance populated to match JSON

name
alias of SolarSystemObjectName

class StarRasterParametersSchema(**args: Any*, ***kwargs: Any*)
Marshmallow schema for converting a StarRasterParameters to/from JSON

make_starrasterparameters(*data*, **_)
Convert parsed JSON back into a StarRasterParameters object

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

Returns
StarRasterParameters instance populated to match JSON

class TargetSchema(**args: Any*, ***kwargs: Any*)
Marshmallow class to convert Target to/from JSON.

make_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

pointing_pattern
alias of *PointingPatternSchema*

reference_coordinate
alias of *CoordinatesSchema*

CHAPTER

TWENTYTWO

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 FileSystemScriptSchema(**args: Any*, ***kwargs: Any*)

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: Any*, ***kwargs: Any*)

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 InlineScriptSchema(**args: Any*, ***kwargs: Any*)

Schema for an InlineScript, used in the activities section of an SKA scheduling block

make_embeddedscript(*data*, **_)

Convert parsed JSON back into a EmbeddedScript object.

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

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: Any*, ***kwargs: Any*)

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

CHAPTER

TWENTYTHREE

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: Any*, ***kwargs: Any*)

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: Any*, ***kwargs: Any*)

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

dish_allocations

alias of *DishAllocationSchema*

dish_configurations

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 scan definition

mccs_allocation

alias of *MCCSAllocationSchema*

metadata

alias of *MetaDataSchema*

sdp_configuration

alias of *SDPConfigurationSchema*

subarray_beam_configurations

alias of *SubarrayBeamConfigurationSchema*

target_beam_configurations

alias of *TargetBeamConfigurationSchema*

targets

alias of *TargetSchema*

telescope

alias of *TelescopeType*

CHAPTER

TWENTYFOUR

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: Any*, ***kwargs: Any*)

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

TWENTYFIVE

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

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

fsp_configs

alias of *FSPConfigurationSchema*

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

Marshmallow schema for the ska_oso_pdm.CSPConfiguration class

cbf_config

alias of *CBFConfigurationSchema*

common_config

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

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

subarray_config

alias of *SubarrayConfigurationSchema*

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

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

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

`function_mode`

alias of *FSPFunctionMode*

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

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

CHAPTER

TWENTYSIX

SKA_OSO_PDM.SCHEMAS.SDP

The schemas.sdp.sdp_configuration module defines Marshmallow schemas .

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

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

execution_block

alias of *ExecutionBlockSchema*

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 BeamMapping

processing_blocks

alias of *ProcessingBlockSchema*

resources

alias of *ResourcesSchema*

The schemas.sdp.sdp_configuration module defines Marshmallow schemas .

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

Marshmallow class for the Execution Block class

beams

alias of *BeamSchema*

channels

alias of *ChannelsSchema*

create_execution_block(*data*, **_)

Convert parsed JSON back into a SDPConfiguration object.

Parameters

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

Returns
Execution Block 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 EB configuration

polarisations

alias of *PolarisationSchema*

scan_types

alias of *ScanTypeSchema*

The schemas.sdp.processing_block module defines Marshmallow schemas .

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

Marshmallow schema for the PbDependency 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: Any*, ***kwargs: Any*)

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

dependencies

alias of *PbDependencySchema*

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 *ScriptSchema*

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

Represents the type of workflow being configured on the SDP

create_sdp_script(*data*, **_)

Convert parsed JSON back into a Script object.

Parameters

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

Returns
SDP Script object populated from data

kind

alias of *ScriptKind*

The schemas.sdp.resources defines Marshmallow schemas to convert an SDP Resources instance to/from JSON.

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

Represents the type of Resources being configured on the SDP

create_sdp_res(*data*, **_)

Convert parsed JSON back into a Resources object.

Parameters

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

Returns
SDP Resources 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 Resources configuration

The schemas.sdp.beam module defines Marshmallow schemas that convert an SDP Beam instance to/from JSON.

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

Marshmallow schema for the Beams class.

create_beams(data, **_)

Convert parsed JSON back into a Beam object.

Parameters

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

Returns

Beam instance 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 Beam configuration

function

alias of *BeamFunction*

The schemas.sdp.channels module defines Marshmallow schemas .

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

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

ScanType object populated from data

spectral_windows

alias of *SpectralWindowSchema*

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

Marshmallow schema for the SpectralWindow class.

create_spectral_window(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

The schemas.sdp.scan_type module defines Marshmallow schemas that convert BeamMapping and ScanTypes SDP classes to/from JSON.

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

Marshmallow schema to convert a BeamMapping instance to/from JSON.

create_channel(*data*, ***_*)

Convert parsed JSON back into a BeamMapping object.

Parameters

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

Returns
BeamMapping 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 BeamMapping

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

Marshmallow schema for the ScanType class.

beams

alias of *BeamMappingSchema*

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

The schemas.sdp.polarisation module defines Marshmallow schemas that convert an SDP Polarisation instance to/from JSON.

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

Marshmallow schema for the Polarisation class.

create_polarisations(*data*, **_)

Convert parsed JSON back into a Polarisation object.

Parameters

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

Returns
Polarisation instance 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 Polarisation configuration

CHAPTER

TWENTYSEVEN

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: 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

CHAPTER

TWENTYEIGHT

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: Any*, ***kwargs: Any*)

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

receiver_band

alias of *ReceiverBand*

CHAPTER

TWENTYNINE

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: Any*, ***kwargs: Any*)

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

THIRTY

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: Any*, ***kwargs: Any*)

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

THIRTYONE

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: Any*, ***kwargs: Any*)

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

THIRTYTWO

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.

32.1 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

THIRTYTHREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

M

N

P

R

S

T

U