developer.skatelescope.org Documentation

Release 0.5.0

Marco Bartolini

May 10, 2024

CONTENTS

SDP Data Product Metadata is a Python package to record SKA-specific metadata alongside the data products. It creates metadata files containing:

- Execution block ID
- The context of the execution block provided by the Observation Execution Tool (OET)
- The configuration of the processing block used to generate the data products
- A list of data product files with description, status, size and a cyclic redundancy check (CRC) for error detection
- Associated IVOA ObsCore attributes used for querying astronomical observations

CHAPTER

ONE

CONTENTS OF METADATA FILE

Below shows the contents of the metadata and brief description about them:

- `interface`: Giving the schema. Currently there is no schema for this (just a placeholder). It will be added to the Telescope Model.
- `execution_block`: Identifies the observation
- `context`: This contains free-form data provided by OET. The data is meant to be passed verbatim through from OET/TMC as part of AssignResources (SDP) or Configure (other sub-systems). Currently this is just a placeholder.
- `config`: This is the configuration used for executing the processing script
- `files`: This contains a list of data products (typically MS or HDF files) which have been generated.
  - `crc`: A cyclic redundancy check (CRC) generated from the contents of the file and used to detect errors in the data
  - `description`: Useful details about the file
  - `path`: Path where the data product file is located
  - `size`: Size of the file (in KB)
  - `status`: Indicate current file status
    - `working`: Processes still running, files might be missing or incomplete
    - `done`: Processes finished, files should be complete
    - `failure`: Not finished successfully, files might be incomplete or corrupt
- `obscore`: This contains attributes as specified by the IVOA recommendation for Observation Data Models. It defines core components that are necessary to perform data discovery when querying for astronomical observations. Details of the attributes can be found at IVOA ObsCore.

More details can be found in ADR-55

Note - If the metadata filename needs to be updated, you can do that by publishing it on *METADATA_FILENAME* environment variable.

CHAPTER

TWO

API

2.1 MetaData

class ska_sdp_dataproduct_metadata.metadata.**MetaData**(*path=None*)

Class for generating the metadata file

Parameters
path – location of the metadata file to read

exception ValidationError(*message*, *errors*)

An exception indicating an error during validation of metadata against the schema.

get_data()

Return the data dictionary within the MetaData object

load_processing_block(*pb_id=None*, *mount_path=None*)

Configure a MetaData object based on the data in a processing block

Parameters
pb_id – processing block ID

metadata_schema = <_io.TextIOWrapper
name='/home/docs/checkouts/readthedocs.org/user_builds/sdp-data-product-metadata/
checkouts/latest/src/ska_sdp_dataproduct_metadata/schema/metadata.json' mode='r'
encoding='utf-8'>

new_file(*dp_path=None*, *description=None*, *crc=None*)

Creates a new file into the metadata and add current file status.

Parameters

- **dp_path** – path of the data product Not to be confused with path of the metadata file
- **description** – Description of the file
- **crc** – CRC (Cyclic Redundancy Check) checksum for the file. NB: CRC is supplied, not calculated

Returns
instance of the File class

property output_path

Output metadata path

read(*file*)

Read input metadata file and load in yaml.

Parameters
file – input metadata file

Returns
Returns the yaml loaded metadata file

runtime_abspath(*path*)

The absolute path of *path* relative to the standard prefix. This value is valid at runtime; i.e., it maps to the filesystem in use.

Parameters
path – A path relative to the standard prefix.

set_config(*script*)

Set configuration of generating software.

Parameters
script – Processing script details

set_execution_block_id(*execution_block_id*)

Set the execution_block_id for this MetaData object NB: If this MetaData object describes a dataproduct that was not generated from an execution_block, then it is possible to use any SKA Unique Identifier (https://gitlab.com/ska-telescope/ska-ser-skuid)

Parameters
execution_block_id – an execution_block_id

validate() → list

Validate the current contents of the metadata against the schema.

Returns
A list of errors.

validator = {'additionalProperties': True, 'properties': {'config': {'additionalProperties': True, 'properties': {'cmdline': {'type': ['string', 'null']}, 'commit': {'type': ['string', 'null']}, 'image': {'type': ['string', 'null']}, 'processing_block': {'type': ['string', 'null']}, 'processing_script': {'type': ['string', 'null']}, 'version': {'type': ['string', 'null']}}, 'required': [], 'type': 'object'}, 'context': {'additionalProperties': True, 'properties': {'intent': {'type': 'string'}, 'notes': {'type': 'string'}, 'observer': {'type': 'string'}}, 'required': [], 'type': 'object'}, 'execution_block': {'type': 'string'}, 'files': {'items': {'additionalProperties': True, 'properties': {'crc': {'type': ['string', 'null']}, 'description': {'type': 'string'}, 'path': {'type': 'string'}, 'size': {'type': 'integer'}, 'status': {'enum': ['done', 'failure', 'working'], 'type': 'string'}}, 'required': [], 'type': 'object'}, 'type': 'array'}, 'interface': {'format': 'uri', 'type': 'string'}, 'obscore': {'additionalProperties': False, 'properties': {'access_estsize': {'type': 'integer'}, 'access_format': {'type': 'string'}, 'access_url': {'format': 'uri', 'qt-uri-protocols': ['https'], 'type': 'string'}, 'bib_reference': {'type': 'string'}, 'calib_level': {'enum': [0, 1, 2, 3, 4], 'type': 'integer'}, 'data_rights': {'type': 'string'}, 'dataproduct_subtype': {'type': 'string'}, 'dataproduct_type': {'type': 'string'}, 'em_calib_status': {'type': 'string'}, 'em_max': {'type': 'number'}, 'em_min': {'type': 'number'}, 'em_res_power': {'type': 'number'}, 'em_res_power_max': {'type': 'number'}, 'em_res_power_min': {'type': 'number'}, 'em_resolution': {'type': 'number'}, 'em_stat_error': {'type': 'number'}, 'em_ucd': {'type': 'string'}, 'em_unit': {'type': 'string'}, 'em_xel': {'type': 'integer'}, 'facility_name': {'type': 'string'}, 'instrument_name': {'type': 'string'}, 'o_calib_status': {'type': 'string'}, 'o_stat_error': {'type': 'number'}, 'o_ucd': {'type': 'string'}, 'o_unit': {'type': 'string'}, 'obs_collection': {'type': 'string'}, 'obs_creation_date': {'type': 'string'}, 'obs_creator_did': {'type': 'string'}, 'obs_creator_name': {'type': 'string'}, 'obs_id': {'type': 'string'}, 'obs_publisher_did': {'type': 'string'}, 'obs_release_date': {'type': 'string'}, 'obs_title': {'type': 'string'}, 'pol_states': {'type': 'string'}, 'pol_xel': {'type': 'integer'}, 'proposal_id': {'type': 'string'}, 'publisher_id': {'type': 'string'}, 's_calib_status': {'type': 'string'}, 's_dec': {'type': 'number'}, 's_fov': {'type': 'number'}, 's_pixel_scale': {'type': 'number'}, 's_ra': {'type': 'number'}, 's_region': {'type': 'string'}, 's_resolution': {'type': 'number'}, 's_resolution_max': {'type': 'number'}, 's_resolution_min': {'type': 'number'}, 's_stat_error': {'type': 'number'}, 's_ucd': {'type': 'string'}, 's_unit': {'type': 'string'}, 's_xel1': {'type': 'integer'}, 's_xel2': {'type': 'integer'}, 't_calib_status': {'type': 'string'}, 't_exptime': {'type': 'number'}, 't_max': {'type': 'number'}, 't_min': {'type': 'number'}, 't_refpos': {'type': 'string'}, 't_resolution': {'type': 'number'}, 't_stat_error': {'type': 'number'}, 't_xel': {'type': 'integer'}, 'target_class': {'type': 'string'}, 'target_name': {'type': 'string'}}, 'required': [], 'type': 'object'}}, 'required': ['execution_block'], 'type': 'object'}

write()

Write the metadata to a yaml file.

2.2 File

class ska_sdp_dataproduct_metadata.metadata.**File**(*metadata*, *path*)

Class to represent the file in the metadata.

property full_path

Get the full path object.

update_status(*status*)

Update the current file status.

Param
status: status to be updated to

2.3 ObsCore

class ska_sdp_dataproduct_metadata.obscore.**ObsCore**

SKA-specific possible values for ObsCore attributes

class **AccessFormat**(*value*)

The format (mime-type) of the data product if downloaded as a file

BINARY = 'application/octet-stream'

FITS = 'image/fits'

HDF5 = 'application/x-hdf5'

JPEG = 'image/jpeg'

PNG = 'image/png'

TAR_GZ = 'application/x-tar-gzip'

UNKNOWN = 'application/unknown'

class **CalibrationLevel**(*value*)

The amount of calibration processing that has been applied to create the data product Refer to the IVOA standard for a full description of the categories

LEVEL_0 = 0

LEVEL_1 = 1

LEVEL_2 = 2

LEVEL_3 = 3

LEVEL_4 = 4

class **DataProductType**(*value*)

A simple string value describing the primary nature of the data product

MS = 'MS'

POINTING = 'POINTING-OFFSETS'

UNKNOWN = 'Unknown'

class ObservationCollection(*value*)

A string identifying the data collection to which the data product belongs

SIMULATION = 'Simulation'

UNKNOWN = 'Unknown'

SKA = 'SKA-Observatory'

SKA_LOW = 'SKA-LOW'

SKA_MID = 'SKA-MID'

class UCD(*value*)

A list of Unified Content Descriptors (Preite Martinez, et al. 2007) describing the nature of the observable within the data product https://www.ivoa.net/documents/latest/UCDlist.html

COUNT = 'phot.count'

FLUX_DENSITY = 'phot.flux.density'

FOURIER = 'stat.fourier'

UNKNOWN = 'Unknown'

CHAPTER

THREE

INDICES AND TABLES

- genindex

INDEX

R

S

T

U

V

W