

---

# **Subarray Node Documentation**

***Release 1.0***

**NCRA India**

**Jun 30, 2023**



# GETTING STARTED

<b>1</b>	<b>Getting started</b>	<b>3</b>
<b>2</b>	<b>SubarrayNode code quality guidelines</b>	<b>5</b>
<b>3</b>	<b>API</b>	<b>7</b>
<b>4</b>	<b>Indices and tables</b>	<b>35</b>
	<b>Python Module Index</b>	<b>37</b>
	<b>Index</b>	<b>39</b>



This project is developing the SubarrayNode (Mid and Low) component of the Telescope Monitoring and Control (TMC) prototype, for the [Square Kilometre Array](#).



## GETTING STARTED

This page contains instructions for software developers who want to get started with usage and development of the SubarrayNode.

### 1.1 Background

Detailed information on how the SKA Software development community works is available at the [SKA software developer portal](#). There you will find guidelines, policies, standards and a range of other documentation.

### 1.2 Set up your development environment

This project is structured to use k8s for development and testing so that the build environment, test environment and test results are all completely reproducible and are independent of host environment. It uses `make` to provide a consistent UI (run `make help` for targets documentation).

#### 1.2.1 Install minikube

You will need to install *minikube* or equivalent k8s installation in order to set up your test environment. You can follow the instruction [here](#): `:: git clone git@gitlab.com:ska-telescope/sdi/deploy-minikube.git cd deploy-minikube make all eval $(minikube docker-env)`

*Please note that the command `eval \$(minikube docker-env)` will point your local docker client at the docker-in-docker for minikube. Use this only for building the docker image and another shell for other work.*

#### 1.2.2 How to Use

Clone this repo: `:: git clone https://gitlab.com/ska-telescope/ska-tmc-subarraynode.git cd ska-tmc-subarraynode`

Install dependencies: `:: apt update apt install -y curl git build-essential libboost-python-dev libtango-dev curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py | python3 - source $HOME/.poetry/env`

**Please note that:**

- the *libtango-dev* will install an old version of the TANGO-controls framework (9.2.5);
- the best way to get the framework is compiling it (instructions can be found [here](#));
- the above script has been tested with Ubuntu 20.04.

*During this step, 'libtango-dev' instalation can ask for the Tango Server IP:PORT. Just accept the default proposed value.*

Install python requirements for linting and unit testing: :: \$ poetry install

Activate the poetry environment: :: \$ source \$(poetry env info --path)/bin/activate

Alternate way to install and activate poetry

Follow the steps till installation of dependencies. Then run below command: :: \$ virtualenv cn\_venv \$ source cn\_venv/bin/activate \$ make requirements

Run python-test: :: \$ make python-test PyTango 9.3.3 (9, 3, 3) PyTango compiled with: Python : 3.8.5 Numpy : 0.0.0  
## output generated from a WSL windows machine Tango : 9.2.5 Boost : 1.71.0

PyTango runtime is: Python : 3.8.5 Numpy : None Tango : 9.2.5

PyTango running on: uname\_result(system='Linux', node='LAPTOP-5LBGJH83', release='4.19.128-microsoft-standard', version='#1 SMP Tue Jun 23 12:58:10 UTC 2020', machine='x86\_64', processor='x86\_64')

===== test session starts ===== platform linux  
- Python 3.8.5, pytest-5.4.3, py-1.10.0, pluggy-0.13.1 - /home/ [...]

----- JSON report ----- JSON report written to:  
build/reports/report.json (165946 bytes)

----- coverage: platform linux, python 3.8.5-final-0 ----- Coverage HTML written to dir build/htmlcov Cover-  
age XML written to file build/reports/code-coverage.xml

===== 48 passed, 5 deselected in 42.42s =====

Formatting the code: :: \$ make python-format [...] ----- Your  
code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

Python linting: :: \$ make python-lint [...] ----- Your code has  
been rated at 10.00/10 (previous run: 10.00/10, +0.00)



## SUBARRAYNODE CODE QUALITY GUIDELINES

### 2.1 Code formatting / style

#### 2.1.1 Black

SubarrayNode uses the `black` code formatter to format its code. Formatting can be checked using the command `make python-format`.

The CI pipeline does check that if code has been formatted using `black` or not.

#### 2.1.2 Linting

SubarrayNode uses below libraries/utilities for linting. Linting can be checked using command `make python-lint`.

- **isort** - It provides a command line utility, Python library and plugins for various editors to quickly sort all your imports.
- **black** - It is used to check if the code has been blacked.
- **flake8** - It is used to check code base against coding style (PEP8), programming errors (like “library imported but unused” and “Undefined name”),etc.
- **pylint** - It is looks for programming errors, helps enforcing a coding standard, sniffs for code smells and offers simple refactoring suggestions.

### 2.2 Test coverage

SubarrayNode uses `pytest` to test its code, with the `pytest-cov` plugin for measuring coverage.



## 3.1 ska\_tmc\_subarraynode package

### 3.1.1 Subpackages

**ska\_tmc\_subarraynode.commands package**

**Submodules**

**ska\_tmc\_subarraynode.commands.abstract\_command module**

Abstract command class for SubarrayNode, a subclass of TMCCCommand. This class provides methods to initialize adapters and executing commands on subarray devices.

```
class ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand(*args: Any,  
                                                                           **kwargs: Any)
```

Bases: TMCCCommand

This code defines the SubarrayNodeCommand class, which is a subclass of TMCCCommand. This class provides methods for initializing adapters and executing commands on subarray devices.

```
adapter_error_message_result(dev_name: str, exception) →  
                               Tuple[ska_tango_base.commands.ResultCode, str]
```

Returns ResultCode.FAILED with failure message as a tuple

```
do(argin=None)
```

```
get_adapter_by_device_name(device_name)
```

**The get\_adapter\_by\_device\_name method takes a device\_name as**

input and searches for an adapter object in the adapter\_factory object's adapters attribute that matches the input device\_name. If a matching adapter object is found, it is returned. If no matching adapter object is found, None is returned.

Args: device\_name (str): The name of the device to search for.

Returns: An adapter object if a matching device is found, otherwise None.

```
get_dish_adapter_by_device_name(device_name_list)
```

**The get\_dish\_adapter\_by\_device\_name method takes a list of**

device\_name\_list as input and searches for adapter objects in

the `adapter_factory` object's `adapters` attribute that match any of the device names in the input list. It returns a list of matching adapter objects.

Args: `device_name_list` (list): A list of device names to search for.

Returns: A list of adapter objects that match the device names in the input list. If no matching adapter object is found, an empty list is returned.

**`init_adapters()`**

**`init_adapters_low()`**

**`init_adapters_mid()`**

**`reject_command(message)`**

Method to return task status as `TaskStatus.REJECTED` along with the error message

**`update_command_in_progress(command_name)`**

**The `update_command_in_progress` method updates the `command_in_progress` and `command_in_progress_id` attributes of the `component_manager` object with the given `command_name`. It also logs information about the updated command.**

Args: `command_name` (str): The name of the command being updated.

Returns: None

### **`ska_tmc_subarraynode.commands.assign_resources_command` module**

`AssignResourcesCommand` class for `SubarrayNode`.

```
class ska_tmc_subarraynode.commands.assign_resources_command.AssignResources(*args: Any,  
                                     **kwargs:  
                                     Any)
```

Bases: `SubarrayNodeCommand`

A class for `SubarrayNode`'s `AssignResources()` command.

Assigns resources to the subarray. It accepts receptor id list as well as SDP resources string as a `DevString`. Upon successful execution, the '`receptorIDList`' attribute of the subarray is updated with the list of receptors and SDP resources string is pass to `SDPSubarrayLeafNode`, and returns list of assigned resources as well as passed SDP string as a `DevString`.

Note: Resource allocation for CSP and SDP resources is also implemented but currently CSP accepts only `receptorIDList` and SDP accepts resources allocated to it.

**`assign_csp_resources(receptor_ids)`**

This function accepts the receptor IDs list as input and invokes the assign resources command on the CSP Subarray Leaf Node.

#### **Parameters**

**argin** – List of strings Contains the list of strings that has the resources ids. Currently this list contains only receptor ids.

Example: ['SKA001', 'SKA002']

#### **Returns**

List of strings. Returns the list of CSP resources successfully assigned to the Subarray. Currently, the `CSPSubarrayLeafNode.AssignResources` function returns void. The function only loops back the input argument in case of successful resource allocation, or returns exception object in case of failure.

**assign\_low\_csp\_resources**(*argin*)

This function accepts the CSP Resources as input and assigns CSP resources to CSP Subarray through CSP Subarray Leaf Node.

**Parameters**

**argin** – List of strings Contains the list of strings that has the resources ids. Currently processing block ids are passed to this function.

**Returns**

A tuple containing ResultCode and string.

**assign\_sdp\_resources**(*argin*)

This function accepts the receptor ID list as input and assigns SDP resources to SDP Subarray through SDP Subarray Leaf Node.

**Parameters**

**argin** – List of strings Contains the list of strings that has the resources ids. Currently processing block ids are passed to this function.

**Returns**

List of strings.

Example: ['PB1', 'PB2']

Returns the list of successfully assigned resources. Currently the SDPSubarrayLeafNode.AssignResources function returns void. Thus, this function just loops back the input argument in case of success or returns exception object in case of failure.

**do\_low**(*argin*)

Method to invoke AssignResources command on subarraynode low.

**Parameters**

**argin** – DevString.

Example:

```
{“interface”:      “https://schema.skao.int/ska-tmc-assignresources/2.1”,    “transaction_id”:“txn-...-00001”,“subarray_id”: 1,“mccs”:{“subarray_beam_ids”:

    [1,“station_ids”: [[1,2]],“channel_blocks”: [3]],“sdp”:{“interface”:

    “https://schema.skao.int/ska-sdp-assignres/0.4”,“execution_block”:      {“eb_id”:      “eb-mvp01-20200325-00001”,“max_length”:      100,“context”:{},“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 timing”},{“beam_id”:“pst2”,“timing_beam_id”:2,“function”:      “pulsar timing”},{“beam_id”:      “vlbi1”,“vlbi_beam_id”:1,“function”:      “vlbi”}],“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]]}], {“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”]}],“fields”:      [{“field_id”:      “field_a”,“phase_dir”:{“ra”:[123,0.1],“dec”:[123,0.1],“reference_time”:      “...”,“reference_frame”:      “ICRF3”}],“pointing_fqdn”:      “low-tmc/telstate/0/pointing”}}],    “processing_blocks”:[{“pb_id”:      “pb-mvp01-20200325-00001”,“sbi_ids”:[“sbi-mvp01-20200325-00001”],
```

```

“script”: {}, “parameters”: {}, “dependencies”: {}, { “pb_id”:      “pb-mvp01-20200325-00002”, “sbi_ids”:
[“sbi-mvp01-20200325-00002”], “script”: {}, “parameters”: {}, “dependencies”: {}, { “pb_id”:      “pb-
mvp01-20200325-00003”, “sbi_ids”: [“sbi-mvp01-20200325-00001”, “sbi-mvp01-20200325-
00002”], “script”: {}, “parameters”:      {}, “dependencies”: {} }, “resources”: { “csp_links”: [1,2,3,4],
“receptors”: [“FS4”, “FS8”], “receive_nodes”: 10 } } “csp”: { “interface”: “https://schema.skao.int/
ska-low-csp-assignresources/2.0”, “common”: { “subarray_id”: 1 }, “lowcbf”: { “resources”:      [ { “de-
vice”: “fsp_01”, “shared”: true, “fw_image”: “pst”, “fw_mode”: “unused” }, { “device”: “p4_01”,
“shared”: true, “fw_image”: “p4.bin”, “fw_mode”: “p4” } ] } } }

```

**Returns**

A tuple containing ResultCode and string.

**rtype:**

(ResultCode, str)

**Raises**

- **ValueError** if input argument json string contains invalid value –
- **Exception** if the command execution is not successful –

**do\_mid(*argin*)**

Method to invoke AssignResources command on subarraynode mid.

**Parameters**

**argin** – DevString.

Example:

```

{ “interface”:      “https://schema.skao.int/ska-tmc-assignresources/2.1”, “transaction_id”:      “txn-
...-00001”, “subarray_id”:      1, “dish”:      { “receptor_ids”:      [“SKA001”], “sdp”: { “interface”:
“https://schema.skao.int/ska-sdp-assignres/0.4”, “execution_block”:      { “eb_id”:      “eb-mvp01-
20200325-00001”, “max_length”:      100, “context”: {}, “beams”: [ { “beam_id”:      “vis0”, “func-
tion”: “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 timing” }, { “beam_id”: “pst2”, “timing_beam_id”: 2,
“function”:      “pulsar timing” }, { “beam_id”:      “vlbi1”, “vlbi_beam_id”: 1, “function”:      “vlbi” },
“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]] } ] }, { “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”] }, { “fields”:
[ { “field_id”:      “field_a”, “phase_dir”: { “ra”: [123,0.1], “dec”: [123,0.1], “reference_time”:      “...”,
“reference_frame”:      “ICRF3” }, “pointing_fqdn”:      “low-tmc/telstate/0/pointing” } ] }, “process-
ing_blocks”: [ { “pb_id”:      “pb-mvp01-20200325-00001”, “sbi_ids”: [“sbi-mvp01-20200325-00001”],
“script”: {}, “parameters”: {}, “dependencies”: {} }, { “pb_id”:      “pb-mvp01-20200325-00002”, “sbi_ids”:
[“sbi-mvp01-20200325-00002”], “script”: {}, “parameters”: {}, “dependencies”: {} }, { “pb_id”:      “pb-
mvp01-20200325-00003”, “sbi_ids”: [“sbi-mvp01-20200325-00001”, “sbi-mvp01-20200325-00002”],
“script”: {}, “parameters”:      {}, “dependencies”: {} }, “resources”: { “csp_links”: [1,2,3,4],      “recep-
tors”: [“FS4”, “FS8”], “receive_nodes”: 10 } } }

```

**Returns**

A tuple containing a return code and string of Resources added to the Subarray. Example of

string of Resources : ["SKA001","SKA002"] as argout if allocation successful.

**rtype:**

(ResultCode, str)

**Raises**

- **ValueError** if input argument json string contains invalid value –
- **Exception** if the command execution is not successful –

**invoke\_assign\_resources**(*argin=None, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)

This is a long running method for AssignResources command, it executes do hook, invokes AssignResources command on CspSubarrayleafnode and SdpSubarrayleafnode.

**Parameters**

- **logger** (*logging.Logger*) – logger
- **logger** – argin
- **argin** (*logging.Logger*) – JSON string consisting of the resources to be assigned
- **task\_abort\_event** (*Event, optional*) – Check for abort, defaults to None

**set\_low\_assigned\_resources**(*argin*)

Set assigned resources to low telescope

**set\_up\_dish\_data**(*receptor\_ids*)

Adds the receptors in dish leaf node group. The healthState and pointintgState attributes of all all the dishes are subscribed.

Note: Currently there are only receptors allocated so the group contains only receptor ids.

**Parameters**

**receptor\_ids** – List of receptor IDs to be allocated to subarray. Example: ['SKA001', 'SKA002']

**Returns**

List of Resources added to the Subarray. Example: ['SKA001', 'SKA001']

**update\_task\_status**(*result: ska\_tango\_base.commands.ResultCode, message: str = ""*) → None

Method to update task status with result code and exception message if any.

**validate\_low\_json**(*argin: <module 'json' from  
'/home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/\_\_init\_\_.py'>*)

Method to validate low input jsons for AssignResources command

**validate\_receptor\_ids**(*receptor\_list: List[str]*) → bool

Validate the receptor IDs to be in the correct format. The expected format is 'SKAnnn' where nnn is a 3 digit number in range 001 to 133.

**ska\_tmc\_subarraynode.commands.configure\_command module**

Configure Command class for SubarrayNode.

**class** ska\_tmc\_subarraynode.commands.configure\_command.**Configure**(\*args: Any, \*\*kwargs: Any)

Bases: *SubarrayNodeCommand*

A class for SubarrayNode's Configure() command.

Configures the resources assigned to the Subarray. The configuration data for SDP, CSP and Dish is extracted out of the input configuration string and relayed to the respective underlying devices (SDP Subarray Leaf Node, CSP Subarray Leaf Node and Dish Leaf Node).

**check\_only\_dish\_config**(scan\_configuration)

Method to check only dish configuration

**configure\_low\_csp**()

Method to configure low CSP

**do\_low**(argin)

This method executes the Configure workflow of the Subarray Node Low. It will invoke Configure command on the CSP Subarray Leaf Node, SDP Subarray Leaf Node

**Parameters**

**argin** – DevString.

JSON string example is:

```
{ "interface": "https://schema.skao.int/ska-low-tmc-configure/3.0", "transaction_id": "txn-
...-00001", "mccs": { "stations": [ { "station_id": 1, "station_id": 2 }, "subarray_beams":
[ { "subarray_beam_id": 1, "station_ids": [ 1, 2 ], "update_rate": 0.0, "channels": [ [ 0, 8, 1, 1 ],
[ 8, 8, 2, 1 ], [ 24, 16, 2, 1 ] ], "antenna_weights": [ 1.0, 1.0, 1.0 ], "phase_centre": [ 0.0, 0.0 ], "target": { "ref-
erence_frame": "HORIZON", "target_name": "DriftScan", "az": 180.0, "el": 45.0 } } ], "sdp": { "in-
terface": "https://schema.skao.int/ska-sdp-configure/0.4", "scan_type": "science_A", "csp":
{ "interface": "https://schema.skao.int/ska-csp-configure/2.0", "subarray": { "subarray_name":
"science
period23", "common": { "config_id": "sbi-mvp01-20200325-00001-
science_A", "lowcbf": { "stations": { "stns": [ [ 1, 0 ], [ 2, 0 ], [ 3, 0 ], [ 4, 0 ] ], "stn_beams":
[ { "beam_id": 1, "freq_ids": [ 64, 65, 66, 67, 68, 68, 70, 71 ], "boresight_dly_poly": "url" } ], "tim-
ing_beams": { "beams": [ { "pst_beam_id": 13, "stn_beam_id": 1, "offset_dly_poly": "url",
"stn_weights": [ 0.9, 1.0, 1.0, 0.9 ], "jones": "url", "dest_ip": [ "10.22.0.1:2345", "10.22.0.3:3456" ]
, "dest_chans": [ 128, 256 ], "rfi_enable": [ true, true, true ], "rfi_static_chans": [ 1, 206, 997 ],
"rfi_dynamic_chans": [ 242, 1342 ], "rfi_weighted": 0.87 } } ] } }, "tmc": { "scan_duration": 10.0 } }
```

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ReturnCode, str)

**do\_mid**(argin)

Method to invoke Configure command.

**Parameters**

**argin** – DevString.



JSON string that includes pointing parameters of Dish - Azimuth and Elevation Angle, CSP Configuration and SDP Configuration parameters. JSON string example is:

```
{
  "interface": "https://schema.skao.int/ska-tmc-configure/2.0",
  "transaction_id": "txn-00001",
  "pointing": {
    "target": {
      "reference_frame": "ICRS",
      "target_name": "Polaris Australis",
      "ra": "21:08:47.92",
      "dec": "-88:57:22.9"
    },
    "dish": {
      "receiver_band": "1"
    },
    "csp": {
      "interface": "https://schema.skao.int/ska-csp-configure/2.0",
      "subarray": {
        "subarray_name": "science period 23"
      },
      "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
      },
      "cbf": {
        "fsp": [
          {
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [[0, 2], [744, 0]],
            "channel_offset": 0,
            "output_link_map": [[0, 0], [200, 1]]
          },
          {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "channel_averaging_map": [[0, 2], [744, 0]],
            "channel_offset": 744,
            "output_link_map": [[0, 4], [200, 5]]
          }
        ],
        "zoom_window_tuning": 650000,
        "vlbi": {},
        "pss": {},
        "pst": {},
        "sdp": {
          "interface": "https://schema.skao.int/ska-sdp-configure/0.3",
          "scan_type": "science_A",
          "tmc": {
            "scan_duration": 10.0
          }
        }
      }
    }
  }
}
```

Note: While invoking this command from JIVE, provide above JSON string without any space.

### Returns

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

### rtype:

(ReturnCode, str)

**invoke\_configure**(*argin=None, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)

This is a long running method for Configure command, it executes do hook, invokes Configure command on CspSubarrayleafnode and SdpSubarrayleafnode.

### Parameters

- **argin** (*Json string, defaults to None*) – JSON string consisting of the resources to be configured
- **task\_abort\_event** (*Event, optional*) – Check for abort, defaults to None

### update\_task\_status()

Method for implementing for updating task status

**validate\_low\_json**(*argin: <module 'json' from /home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/\_\_init\_\_.py>*)

Method to validate keys for low csp and sdp devices from configure json.

**validate\_mid\_json**(*argin: <module 'json' from /home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/\_\_init\_\_.py>, req\_keys*)

Method to validate keys for mid csp and sdp devices from configure json.

**class** ska\_tmc\_subarraynode.commands.configure\_command.**ElementDeviceData**

Bases: object

A class representing data for an element device.

**static build\_up\_csp\_cmd\_data**(*scan\_config, delay\_model\_subscription, receive\_addresses\_map, component\_manager*)

Here the input data for CSP is build which is used in configuration of CSP. Below is the csp\_config\_schema variable value generated by using ska\_telmodel library.

```
{
  "interface": "https://schema.skao.int/ska-csp-configure/2.0",
  "subarray": {
    "subarray_name": "science period 23"
  },
  "common": {
    "config_id": "sbi-mvp01-20200325-00001-science_A",
    "frequency_band": "1",
    "subarray_id": 1
  },
  "cbf": {
    "fsp": [
      {
        "fsp_id": 1,
        "function_mode": "CORR",
```

```
“frequency_slice_id”:1,”integration_factor”:1,”zoom_factor”:0,”channel_averaging_map”:
[[0,2],[744,0]],”channel_offset”:0,”output_link_map”:[[0,0],[200,1]],”output_host”:
[[0,”192.168.0.1”],[400,”192.168.0.2”]],”output_mac”:[[0,”06-00-00-00-00-00”]],      “out-
put_port”:[[0,9000,1],[400,9000,1]],{“fsp_id”:2,”function_mode”:”CORR”,      “fre-
quency_slice_id”:2,”integration_factor”:1,”zoom_factor”:1,”channel_averaging_map”:
[[0,2],[744,0]],”channel_offset”:744,”output_link_map”:[[0,4],[200,5]],”zoom_window_tuning”:
650000,”output_host”:[[0,”192.168.0.3”],[400,”192.168.0.4”]],”output_mac”:      [[0,”06-00-00-00-00-
01”]],”output_port”:[[0,9000,1],[400,9000,1]]},”vlbi”:{},”pss”:{},”pst”:{}}
```

**Returns**

csp configuration schema

**static build\_up\_dsh\_cmd\_data**(*scan\_config*, *component\_manager*)

Method to build up dish command data

**static build\_up\_sdp\_cmd\_data**(*scan\_config*, *component\_manager*)

Method to build up sdp command data

**ska\_tmc\_subarraynode.commands.end\_command module**

A class for TMC SubarrayNode’s End() command

**class** `ska_tmc_subarraynode.commands.end_command.End`(\*args: Any, \*\*kwargs: Any)

Bases: [\*SubarrayNodeCommand\*](#)

A class for SubarrayNode’s End() command.

This command on Subarray Node invokes End command on CSP Subarray Leaf Node and SDP Subarray Leaf Node, and stops tracking of all the assigned dishes.

**do\_low**(*argin=None*)

Method to invoke End command on MCCS Subarray Leaf Node.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**do\_mid**(*argin=None*)

Method to invoke End command on CSP Subarray Leaf Node, SDP Subarray Leaf Node and Dish Leaf Nodes.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**end\_csp**()

End command on CSP Subarray Leaf Node

**end\_sdp**()

End command on SDP Subarray Leaf Node

**invoke\_end**(*logger*, *task\_callback*: *Optional[Callable] = None*, *task\_abort\_event*: *Optional[Event] = None*)

This is a long running method for End command, it executes do hook, invokes End Command SdpSubarrayleafnode.

#### Parameters

- **logger** (*logging.Logger*) – logger
- **task\_abort\_event** (*Event*, *optional*) – Check for abort, defaults to None

**stop\_dish\_tracking()**

Method to stop dish tracking

**update\_task\_status()**

updating the task status

### ska\_tmc\_subarraynode.commands.end\_scan\_command module

A class for TMC SubarrayNode's EndScan() command.

**class** ska\_tmc\_subarraynode.commands.end\_scan\_command.**EndScan**(\*args: Any, \*\*kwargs: Any)

Bases: *SubarrayNodeCommand*

A class for SubarrayNode's EndScan() command.

Ends the scan. It is invoked on subarray after completion of the scan duration. It can also be invoked by an external client while a scan is in progress, Which stops the scan immediately irrespective of the provided scan duration.

**do\_low**(*argin=None*)

This method executes the End Scan workflow of the Subarray Node Low. It will invoke End Scan command on the CSP Subarray Leaf Node, SDP Subarray Leaf Node.

#### Returns

None

#### Raises

**DevFailed** if the command execution is not successful. –

**do\_mid**(*argin=None*)

Method to invoke Endscan command.

#### Returns

None

#### Raises

**The command execution is not successful.** –

**end\_scan()**

Method for setting device for Subarray Mid

**end\_scan\_low()**

setting up device for low sdp and csp

**end\_scan\_mid()**

Method for setting up device for mid SDP and CSP

**endscan\_csp()**

set up csp devices

**endscan\_sdp()**

set up sdp devices

**invoke\_end\_scan**(*logger*, *task\_callback*: *Optional[Callable]* = *None*, *task\_abort\_event*: *Optional[Event]* = *None*)

This is a long running method for Scan command, it executes do hook, invokes EndScan command on CspSubarrayleafnode and SdpSubarrayleafnode.

**Parameters**

- **logger** (*logging.Logger*) – logger
- **task\_abort\_event** (*Event*, *optional*) – Check for abort, defaults to None

**update\_task\_status()**

Method for implementing for updating task status

### **ska\_tmc\_subarraynode.commands.obsreset\_command module**

ObsReset Command for SubarrayNode.

**class** `ska_tmc_subarraynode.commands.obsreset_command.ObsReset`(\*args: Any, \*\*kwargs: Any)

Bases: `ObsResetCommand`

A class for SubarrayNode's ObsReset() command.

This command invokes ObsReset command on CspSubarrayLeafNode, SdpSubarrayLeafNode and DishLeafNode.

**completed()**

Placeholder method that indicates a task has completed.

**do**(*argin*=*None*)

do method of obsreset command

**do\_low()**

Method to invoke ObsReset command on MCCS Subarray Leaf Node.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(`ResultCode`, `str`)

**do\_mid()**

Method to invoke ObsReset command on CSP Subarray Leaf Node, SDP Subarray Leaf Node and Dish Leaf Nodes.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(`ResultCode`, `str`)

**get\_csp\_subarray\_obsstate()**

Returns csp subarray obsstate

**get\_sdp\_subarray\_obsstate()**

Returns sdp subarray obsstate

**is\_allowed(raise\_if\_disallowed=True)**

check if component manager is of instance of mid or low

**is\_allowed\_low(raise\_if\_disallowed)**

Whether this command is allowed to run in the current state of the state model.

**Parameters**

**raise\_if\_disallowed** – whether to raise an error or simply return False if the command is disallowed

**Returns**

whether this command is allowed to run

**Return type**

boolean

**is\_allowed\_mid(raise\_if\_disallowed)**

Whether this command is allowed to run in the current state of the state model.

**Parameters**

**raise\_if\_disallowed** – whether to raise an error or simply return False if the command is disallowed

**Returns**

whether this command is allowed to run

**Return type**

boolean

**obsreset\_csp()**

Invoke ObsReset command on CSP Subarray Leaf Node.

**obsreset\_sdp()**

Invoke ObsReset command on SDP Subarray Leaf Node.

**obsreset\_dish()**

Invoke ObsReset command on Dish Leaf Node.

## **ska\_tmc\_subarraynode.commands.release\_all\_resources\_command module**

ReleaseAllResources Command for SubarrayNode

```
class ska_tmc_subarraynode.commands.release_all_resources_command.ReleaseAllResources(*args:  
    Any,  
    **kwargs:  
    Any)
```

Bases: *SubarrayNodeCommand*

A class for TMC SubarrayNode's ReleaseAllResources() command.

It checks whether all resources are already released. If yes then it returns code FAILED. If not it Releases all the resources from the subarray i.e. Releases resources from TMC Subarray Node, CSP Subarray and SDP

Subarray. Upon successful execution, all the resources of a given subarray get released and empty array is returned. Selective release is not yet supported.

### **clean\_up\_low\_resources()**

Clears the assignedResources attribute.

Cleans dictionaries of the resources across the subarraynode.

Note: Currently there are only receptors allocated so only the receptor ids details are stored.

#### **Parameters**

**argin** – None

#### **Returns**

None

### **clean\_up\_mid\_resources()**

Clears the AssignedResources attribute.

Cleans dictionaries of the resources across the subarraynode.

Note: Currently there are only receptors allocated so only the receptor ids details are stored.

#### **Parameters**

**argin** – None

#### **Returns**

None

### **do\_low(*argin=None*)**

Method to invoke ReleaseAllResources command.

#### **Returns**

A tuple containing a return code STARTED on successful release all resources and message.

#### **rtype:**

(ResultCode, str)

### **do\_mid(*argin=None*)**

Method to invoke ReleaseAllResources command.

#### **Returns**

A tuple containing a return code and "" as a string on successful release all resources.

#### **rtype:**

(ResultCode, str)

### **invoke\_release\_resources(*logger, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)**

This is a long running method for ReleaseResources command, it executes do hook, invokes ReleaseResources command on CspSubarrayleafnode and SdpSubarrayleafnode.

#### **Parameters**

- **logger** (*logging.Logger*) – logger
- **task\_abort\_event** (*Event, optional*) – Check for abort, defaults to None

**release\_csp\_resources()**

This function invokes releaseAllResources command on CSP Subarray via CSP Subarray Leaf Node.

**Parameters**

**argin** – DevVoid

**Returns**

DevVoid

**release\_sdp\_resources()**

This function invokes releaseAllResources command on SDP Subarray via SDP Subarray Leaf Node.

**Parameters**

**argin** – DevVoid

**Returns**

DevVoid

**update\_task\_status()**

Method for implementing for updating task status

**ska\_tmc\_subarraynode.commands.reset\_command module**

Reset Command for SubarrayNode.

**class** ska\_tmc\_subarraynode.commands.reset\_command.**Reset**(\*args: Any, \*\*kwargs: Any)

Bases: ResetCommand

A class for SubarrayNode's Reset() command.

**do**(argin=None)

do method for reset command.

**do\_low**(argin=None)

” Method to invoke Reset command on TMC Low Devices.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**Raises**

**DevFailed.** –

**do\_mid**(argin=None)

” Method to invoke Reset command on SubarrayNode.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**Raises****Exception.** –**is\_allowed**(*raise\_if\_disallowed=True*)

check if component manager is of instance of mid or low

**is\_allowed\_low**(*raise\_if\_disallowed*)

Whether this command is allowed to run in the current state of the state model.

**Parameters****raise\_if\_disallowed** – whether to raise an error or simply return False if the command is disallowed**Returns**

whether this command is allowed to run

**Return type**

boolean

**is\_allowed\_mid**(*raise\_if\_disallowed*)

Whether this command is allowed to run in the current state of the state model.

**Parameters****raise\_if\_disallowed** – whether to raise an error or simply return False if the command is disallowed**Returns**

whether this command is allowed to run

**Return type**

boolean

**ska\_tmc\_subarraynode.commands.restart\_command module**

Restart Command for SubarrayNode.

**class** `ska_tmc_subarraynode.commands.restart_command.Restart`(\*args: Any, \*\*kwargs: Any)Bases: `SubarrayNodeCommand`

A class for SubarrayNode's Restart() command.

**clean\_up\_configuration()**

Restart command is a like a hard reset. So, the transition should ensure the removal of all resources from the subarray.

Removes group of dishes from tango group client.

Unsubscribes events for dish health state and dish pointing state.

Cleans dictionaries of the resources across the subarraynode.

Note: Currently there are only receptors allocated so the group contains only receptor ids.

**Parameters****argin** – None**Returns**

None



**clean\_up\_dishes()**

Removes group of dishes from tango group client

**do\_low(*argin=None*)**

This command on Subarray Node Low invokes Restart command on CSP Subarray Leaf Node and SDP Subarray Leaf Node and restarts the ongoing activity.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**do\_mid(*argin=None*)**

This method invokes Restart command on CSPSubarrayLeafNode, SdpSubarrayLeafNode and DishLeafNode.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ResultCode, str)

**Raises**

- **Exception** if error occurs while invoking command on CSPSubarrayLeafNode, SDpSubarrayLeafNode or –
- **DishLeafNode.** –

**get\_csp\_subarray\_obstate()**

Return obstate of csp subarray obstate

**get\_mccs\_subarray\_obstate()**

Return obstate of mcs subarray obstate

**get\_sdp\_subarray\_obstate()**

Return obstate of sdp subarray obstate

**get\_subarray\_obstate(*dev\_name*)**

Return obstate of subarray obstate

**invoke\_restart(*logger, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)**

This is a long running method for Restart command, it executes do hook, invokes Restart Command

**Parameters**

- **logger** (*logging.Logger*) – logger
- **task\_abort\_event** (*Event, optional*) – Check for abort, defaults to None

**restart\_csp()**

set up csp devices

**restart\_dishes()**

Restarts all dish leaf nodes in the component manager.

**restart\_sdp()**

set up sdp devices

**update\_task\_status()**

Method for implementing for updating task status

**ska\_tmc\_subarraynode.commands.scan\_command module**

A class for TMC SubarrayNode's Scan() command

**class** ska\_tmc\_subarraynode.commands.scan\_command.Scan(\*args: Any, \*\*kwargs: Any)

Bases: [SubarrayNodeCommand](#)

A class for SubarrayNode's Scan() command.

The command accepts Scan id as an input and executes a scan on the subarray. Scan command is invoked on respective CSP and SDP subarray node for the provided interval of time. It checks whether the scan is already in progress. If yes it throws error showing duplication of command.

**do\_low(argin)**

Method to invoke Scan command.

**Parameters**

**argin** – DevString. JSON string containing id.

JSON string example as follows: TODO : Update once json schema is confirmed {“interface”:“<https://schema.skao.int/ska-low-tmc-scan/2.0>”,“transaction\_id”:“txn-...-00001”,“scan\_id”:1} Note: Above JSON string can be used as an input argument while invoking this command from JIVE.

**Returns**

A tuple containing a return code and a string message indicating status. The message is for information purpose only.

**rtype:**

(ReturnCode, str)

**Raises**

**DevFailed** if the command execution is not successful –

**do\_mid(argin)**

Method to invoke Scan command.

**Parameters**

**argin** – DevString. JSON string containing id.

**Example**

```
{ “interface”: “https://schema.skao.intg/ska-tmc-scan/2.0”, “transaction_id”: “txn-...-00001”,  
  “scan_id”: 1 }
```

Note: Above JSON string can be used as an input argument while invoking this command from WEBJIVE.

return: A tuple containing a return code and a string message indicating status. The message is for information purpose only.

rtype: (ReturnCode, str)

**invoke\_scan**(*argin=None, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)

This is a long running method for Scan command, it executes do hook, invokes Scan command on CspSubarrayleafnode and SdpSubarrayleafnode.

#### Parameters

- **argin** (*Json string, defaults to None*) – JSON string consisting of the resources to be Scan
- **task\_abort\_event** (*Event, optional*) – Check for abort, defaults to None

**scan\_csp**(*argin*)

set up csp devices

**scan\_csp\_low**(*argin*)

set up csp devices

**scan\_sdp**(*argin*)

set up sdp devices

**start\_scan\_timer**(*scan\_duration*)

Method for starting scan timer

**update\_task\_status**()

Method for implementing for updating task status

### ska\_tmc\_subarraynode.commands.off\_command module

Off Command for SubarrayNode

**class** ska\_tmc\_subarraynode.commands.off\_command.**Off**(\*args: Any, \*\*kwargs: Any)

Bases: [SubarrayNodeCommand](#)

A class for Subarraynode's Off() command.

**do\_low**(*argin=None*)

Method to invoke off command on the MCCS Subarray Leaf Node.

#### Parameters

**argin** – Input json for Command, defaults to None

:type None

return: A tuple containing a return code and a string message indicating status.

rtype: (ResultCode, str)

**do\_mid**(*argin=None*)

Method to invoke Off command on SDP Subarray Leaf Nodes. :param argin: Input json for Command, defaults to None :type None

return: A tuple containing a return code and a string message indicating status.

rtype: (ResultCode, str)

**get\_csp\_subarray\_obstate**()

Return obstate of csp subarray obstate

**get\_sdp\_subarray\_obstate()**

Return obstate of sdp subarray obstate

**get\_subarray\_obstate(*dev\_name*)**

Return obstate of subarray obstate

**subarray\_off(*logger, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)**

“This is a long running method for Off command, it executes do hook, invokes Off command on SdpSubarrayleafnode. :param *logger*: logger :type *logger*: logging.Logger :param *task\_callback*: Update task state, defaults to None :type *task\_callback*: Callable, optional :param *task\_abort\_event*: Check for abort, defaults to None :type *task\_abort\_event*: Event, optional

**update\_task\_status()**

Method for implementing for updating task status

### **ska\_tmc\_subarraynode.commands.on\_command module**

On Command for SubarrayNode

**class** `ska_tmc_subarraynode.commands.on_command.On(*args: Any, **kwargs: Any)`

Bases: [\*SubarrayNodeCommand\*](#)

A class for the SubarrayNode’s On() command.

**do\_low(*argin=None*)**

Method to invoke On command on MccsSubarrayLeafNode, Low CspSubarrayLeafNode and Low SdpSubarrayLeafNode.

#### **Parameters**

**argin** – Input json for Command, defaults to None

:type None

return: A tuple containing a return code and a string message indicating status. rtype: (ResultCode, str)

#### **Raises**

**Exception if the command execution is not successful –**

**do\_mid(*argin=None*)**

Method to invoke On command on Mid CspSubarrayLeafNode and SdpSubarrayLeafNode.

#### **Parameters**

**argin** – Input json for Command, defaults to None

:type None

return: A tuple containing a return code and a string message indicating status. rtype: (ResultCode, str)

#### **Raises**

**DevFailed if the command execution is not successful –**

**on\_leaf\_nodes(*logger, task\_callback: Optional[Callable] = None, task\_abort\_event: Optional[Event] = None*)**

This is a long running method for On command, it executes do hook, invokes On command on CspSubarrayleafnode and SdpSubarrayleafnode.

#### **Parameters**

- **logger** (*logging.Logger*) – logger

- **task\_callback** (*Callable*, *optional*) – Update task state, defaults to None
- **task\_abort\_event** (*Event*, *optional*) – Check for abort, defaults to None

**update\_task\_status()**

Method for implementing for updating task status

## **ska\_tmc\_subarraynode.commands.standby\_command module**

### **Module contents**

## **ska\_tmc\_subarraynode.manager package**

### **Submodules**

## **ska\_tmc\_subarraynode.manager.aggregators module**

**This Module is for Aggregating the HealthState and  
ObsStates of Devices**

**class** ska\_tmc\_subarraynode.manager.aggregators.**HealthStateAggregatorLow**(\*args: Any, \*\*kwargs: Any)

Bases: Aggregator

The HealthStateAggregatorLow class is an implementation of an aggregator that is responsible for aggregating the health states of various devices in the low telescope

**aggregate()**

This method aggregating HealthState in Low telescope

**class** ska\_tmc\_subarraynode.manager.aggregators.**HealthStateAggregatorMid**(\*args: Any, \*\*kwargs: Any)

Bases: Aggregator

The HealthStateAggregatorMid class is an implementation of an aggregator that is responsible for aggregating the health states of various devices in the mid telescope

**aggregate()**

This method aggregating HealthState in Mid telescope

**class** ska\_tmc\_subarraynode.manager.aggregators.**ObsStateAggregatorLow**(\*args: Any, \*\*kwargs: Any)

Bases: Aggregator

The ObsStateAggregatorLow class is a subclass of the Aggregator class and is responsible for aggregating and managing the observation states of the subarrays

**aggregate()**

Calculates aggregated observation state of Subarray.

**subarray\_node\_obstate\_not\_aggregated()**

the subarray's obsState is not yet aggregated due to missing events.

```
class ska_tmc_subarraynode.manager.aggregators.ObsStateAggregatorMid(*args: Any, **kwargs: Any)
```

Bases: Aggregator

The ObsStateAggregatorMid class is a subclass of the Aggregator class and is responsible for aggregating and managing the observation states of the subarrays

**aggregate()**

Calculates aggregated observation state of Subarray.

**subarray\_node\_obstate\_not\_aggregated()**

the subarray's obsState is not yet aggregated due to missing events.

```
class ska_tmc_subarraynode.manager.aggregators.SubarrayAvailabilityAggregatorLow(*args: Any, **kwargs: Any)
```

Bases: Aggregator

class to aggregate tmc low subarray device availability depending on tmc low leaf nodes

**aggregate()**

Aggregates the subarray availability

```
class ska_tmc_subarraynode.manager.aggregators.SubarrayAvailabilityAggregatorMid(*args: Any, **kwargs: Any)
```

Bases: Aggregator

class to aggregate tmc mid subarray device availability depending on tmc mid leaf nodes

**aggregate()**

Aggregates the subarray availability

### ska\_tmc\_subarraynode.manager.event\_receiver module

This Module is used to Receive events from devices

```
class ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver(*args: Any, **kwargs: Any)
```

Bases: EventReceiver

The SubarrayNodeEventReceiver class has the responsibility to receive events from the sub devices managed by the Subarray node.

The ComponentManager uses the handle events methods for the attribute of interest. For each of them a callback is defined.

TBD: what about scalability? what if we have 1000 devices?

**check\_event\_error(evt)**

Method for checking event error.

**handle\_assigned\_resources\_event(evt)**

Method for handling assigned resources events.

**handle\_csp\_leaf\_obs\_state\_event(*evt*)**

Handle the CSP Leaf Node obsState event.

**Parameters**

**evt** (*tango.ChangeEvent*) – Event data

**handle\_lrcr\_event(*event*)**

Method for handling longRunningCommandResult events.

**handle\_pointing\_state\_event(*evt*)**

Method for handling pointing state event.

**handle\_receive\_addresses\_event(*evt*)**

Method for handling and receiving addresses events.

**handle\_sdp\_leaf\_obs\_state\_event(*evt*)**

Handle the SDP Leaf Node obsState event.

**Parameters**

**evt** (*tango.ChangeEvent*) – Event data

**handle\_subsystem\_availability\_event(*event\_data*)**

Method for handling isSubarrayAvailable attribute events.

**subscribe\_events(*dev\_info*)**

Method for subscribing events

**unsubscribe\_dish\_health(*dish\_proxy, evt\_id*)**

Method for unsubscribing dish health

**unsubscribe\_dish\_leaf\_health(*dish\_leaf\_proxy, evt\_id*)**

Method for unsubscribing dish leaf health

**unsubscribe\_dish\_leaf\_state(*dish\_leaf\_proxy, evt\_id*)**

Method for unsubscribing dish leaf health.

**unsubscribe\_dish\_pointing(*dish\_proxy, evt\_id*)**

Method for unsubscribing dish pointing

**unsubscribe\_dish\_state(*dish\_proxy, evt\_id*)**

Method for unsubscribing dish state

**unsubscribe\_dish\_events()**

Method for unsubscribing dish events

**unsubscribe\_dish\_leaf\_events()**

Method for unsubscribing dish leaf events.

**ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager module****Module contents****ska\_tmc\_subarraynode.model package****Submodules**

### ska\_tmc\_subarraynode.model.component module

This module is used for maintaining and monitoring the components of subarray device.

**class** ska\_tmc\_subarraynode.model.component.SubarrayComponent(\*args: Any, \*\*kwargs: Any)

Bases: TmcComponent

A component class for Subarray Node

It supports:

- Maintaining a connection to its component
- Monitoring its component

#### property assigned\_resources

Return the resources assigned to the component.

##### Returns

the resources assigned to the component

##### Return type

list of str

#### property devices

Return the monitored devices.

##### Returns

the monitored devices

##### Return type

DeviceInfo[]

#### get\_device(device\_name)

Return the monitored device info by name.

##### Parameters

**device\_name** – name of the device

##### Returns

the monitored device info

##### Return type

DeviceInfo

#### invoke\_device\_callback(dev\_info)

This method invoke device callback

#### remove\_device(device\_name)

Remove a device from the list

##### Parameters

**device\_name** – name of the device

#### property sb\_id

Return the Sb\_id

##### Returns

the Sb\_id

##### Return type

str



**property scan\_duration**

Return the duration of scan

**Returns**

the scan duration

**Return type**

int

**property scan\_id**

Return the Scan id

**Returns**

the Scan id

**Return type**

str

**set\_obs\_callbacks(*\_update\_assigned\_resources\_callback=None*)**

This method sets obs callback

**set\_op\_callbacks(*\_update\_device\_callback=None, \_update\_subarray\_health\_state\_callback=None, \_update\_subarray\_availability\_status\_callback=None*)**

this method update device callback and subarray health state

**property subarray\_availability**

Return the aggregated status for subarray node availability

**Returns**

the subarray availability

**Return type**

bool

**property subarray\_health\_state**

Return the aggregated subarray health state

**Returns**

the subarray health state

**Return type**

HealthState

**property subarray\_id**

Return the subarray\_id

**Returns**

the subarray\_id

**Return type**

str

**to\_dict()**

Return result HealthState in dictionary

**update\_device(*dev\_info*)**

Update (or add if missing) Device Information into the list of the component.

**Parameters**

**dev\_info** – a DeviceInfo object

**update\_device\_exception**(*dev\_info*, *exception*)

Update (or add if missing) Device Information into the list of the component.

**Parameters**

**dev\_info** – a DeviceInfo object

### **ska\_tmc\_subarraynode.model.enum module**

Enum Constant Values for Subarray Node

**class** `ska_tmc_subarraynode.model.enum.PointingState`(*value*)

Bases: `IntEnum`

An enumeration class representing the different pointing states of a device.

**NONE** = 0

**READY** = 1

**SCAN** = 4

**SLEW** = 2

**TRACK** = 3

**UNKNOWN** = 5

### **ska\_tmc\_subarraynode.model.input module**

This module provide Provides input parameters to both the telescopes

**class** `ska_tmc_subarraynode.model.input.InputParameter`(*changed\_callback*)

Bases: `object`

Method to Check CSP Subarray Device

**property** `csp_subarray_dev_name`

Return the CSP Subarray device name

**Returns**

the CSP Subarray device name

**Return type**

str

**property** `sdp_subarray_dev_name`

Returns the SDP Subarray device name

**Returns**

the SDP Subarray device name

**Return type**

str

**property** `tmc_csp_sln_device_name`

Return the CSP Subarray Leaf Node device names

**Returns**

the CSP Subarray Leaf Node device names

**Return type**

str

**property tmc\_sdp\_sln\_device\_name**

Return the SDP Subarray Leaf Node device names

**Returns**

the SDP Subarray Leaf Node device names

**Return type**

str

**update**(*component\_manager*)

update the devices in liveliness probe

**class** ska\_tmc\_subarraynode.model.input.**InputParameterLow**(*changed\_callback*)Bases: [\*InputParameter\*](#)

Method to Check CSP subarray Low Device

**class** ska\_tmc\_subarraynode.model.input.**InputParameterMid**(*changed\_callback*)Bases: [\*InputParameter\*](#)

Method to Check CSP Subarray Mid Device

**property dish\_dev\_names**

Return the dish device names

**Returns**

the TM dish device names

**Return type**

list

**property tmc\_dish\_ln\_device\_names**

Return the TM dish device names

**Returns**

the TM dish device names

**Return type**

list

**update**(*component\_manager*)

Update input parameter for Input Parameter

## Module contents

### 3.1.2 Submodules

#### 3.1.3 `ska_tmc_subarraynode._subarraynode_node` module

#### 3.1.4 `ska_tmc_subarraynode._subarraynode_node_low` module

#### 3.1.5 `ska_tmc_subarraynode._subarraynode_node_mid` module

#### 3.1.6 `ska_tmc_subarraynode.exceptions` module

This module has custom exception for repository `ska_tmc_subarraynode`

**exception** `ska_tmc_subarraynode.exceptions.CommandNotAllowed`

Bases: `Exception`

Raised when a command is not allowed.

**exception** `ska_tmc_subarraynode.exceptions.DeviceUnresponsive`

Bases: `Exception`

Raised when a device is not responsive.

**exception** `ska_tmc_subarraynode.exceptions.InvalidObsStateError`

Bases: `ValueError`

Raised when subarray is not in required obsState.

#### 3.1.7 `ska_tmc_subarraynode.release` module

Release information for Python Package

#### 3.1.8 `ska_tmc_subarraynode.transaction_id` module

This module is for identifying and changing the transaction ids

`ska_tmc_subarraynode.transaction_id.identify_with_id(name: str, arg_name: str)`

This method decorator that identifies a transaction with a unique ID and adds it to the wrapped function's object.

`ska_tmc_subarraynode.transaction_id.inject_id(obj, data: Dict) → Dict`

This method injecting a transaction id

`ska_tmc_subarraynode.transaction_id.inject_with_id(arg_position: int, arg_name: str)`

For this method A decorator that injects an ID field into a dictionary-like argument of a function.

`ska_tmc_subarraynode.transaction_id.update_with_id(obj, parameters: Any) → Union[Dict, str]`

Updates the given dictionary-like *obj* with an ID field and the values in *parameters*.

### 3.1.9 Module contents



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### S

- `ska_tmc_subarraynode`, 33
- `ska_tmc_subarraynode.commands`, 25
  - `abstract_command`, 7
  - `assign_resources_command`, 8
  - `configure_command`, 12
  - `end_command`, 14
  - `end_scan_command`, 15
  - `obsreset_command`, 16
  - `off_command`, 23
  - `on_command`, 24
  - `release_all_resources_command`, 17
  - `reset_command`, 19
  - `restart_command`, 20
  - `scan_command`, 22
- `ska_tmc_subarraynode.exceptions`, 32
- `ska_tmc_subarraynode.manager`, 27
  - `aggregators`, 25
  - `event_receiver`, 26
- `ska_tmc_subarraynode.model`, 32
  - `component`, 28
  - `enum`, 30
  - `input`, 30
- `ska_tmc_subarraynode.release`, 32
- `ska_tmc_subarraynode.transaction_id`, 32



## INDEX

## A

```
adapter_error_message_result()
```

(*ska\_tmc\_subarraynode.commands.abstract\_command.SubarrayNodeCommand* method), 7

```
aggregate() (ska_tmc_subarraynode.manager.aggregator.checkOnlyAggregateConfig()
method), 25 (ska_tmc_subarraynode
```

```
aggregate() (ska_tmc_subarraynode.manager.aggregators.HealthSimMethod, gaioMid
method), 25 clean_up_configuration
```

`aggregate()` (`ska_tmc_subarraynode.manager.aggregators.ObsState(AggregatorsSubarraynode.commands.restart_command.Restart`  
`method)`), 25 (`AggregatorsSubarraynode.commands.restart_command.Restart`  
`method)`), 20

`aggregate()` (`ska_tmc_subarraynode.manager.aggregator.CudaSupAgg.aggregate()` `ska_tmc_subarraynode.commands.restart_command`  
`method`), 26 `method`), 20

```
aggregate() (ska_tmc_subarraynode.manager.aggregator.CleanLowAndHighAgg) or Low
method), 26 (ska_tmc_subarraynode.com
```

```
aggregate() (ska_tmc_subarraynode.manager.aggregators.SubarraynodeAggregatorMid  
method), 26 clean up mid resources()
```

```

    assign_csp_resources()
    (ska tmc subarraynode.commands.assign_resources command)

```

```
method), 8 CommandNotAllowed, 32  
assign low csp resources() completed() (ska_tmc_subarraynode.commands.obsreset_command.Obsr
```

```

    (ska_tmc_subarraynode.commands.assign_resources_command).assign_resources
    method).9 Configure (class in ska_tmc_subarraynode)

```

```

    memobj), 2)
    assign_sdp_resources()
    (ska_tmc.subarraynode.commands.assign_resources)

```

```

        (ska_tmc_subarraynode.commands.assign_resources_command.AssignResourcesCommand).execute(
            method), 9
        (ska_tmc_subarraynode.commands.configure_command.ConfigureCommand).execute(
            method), 10
    assigned_resources(ska_tmc_subarraynode.model.component.SubarrayComponent)

```

```

assigned_resources (ska_tmc_subarray_node.model.component.subarray_node.component
property), 28
AssignResources (class in ska_tmc_subarray_node.model.component.subarray_node.model)

```

```

assign_resources (class in ska_tmc_subarraynode.model.input.inputparameter)
ska_tmc_subarraynode.commands.assign_resources_command.py, 30
8

```

**B**

```
build_up_csp_cmd_data()
```

```
(ska_tmc_subarraynode.commands.configure_command(device_id, data,
static method), 13
```

```
build_up_dsh_cmd_data()
    (ska_tmc_subarraynode.commands.configure_commandElementDevice
```

```
static method, 14
build up cmd data()
do() (ska_time_subarraynode.commands.abstract_command.SubarrayNode
method), 7
do() (ska_time_subarraynode.commands.obsreset_command.ObsReset
```

```

        do() (ska_tmc_subarraynode.configure_command(ElementData,
        (ska_tmc_subarraynode.configure_command, static method), 14)
        do() (ska_tmc_subarraynode.configure_command(ElementData,

```

## C

```
check_event_error()
```

`umand.SubarrayNodeEventManager().manager.event_receiver.SubarrayNodeEventManager().method`),<sup>26</sup>

```
rcCheckOnlyAdisgConfig()
(ska_tmc_subarraynode.commands.configure_command.Configure
```

```
rs.HealthStatus = HealthStatusNegotiatorMid
clean_up_configuration()
```

`rs.ObsState(AggregateLawraynode.commands.restart_command.Restart  
method), 20`

```
rc = 0;
rc = rc || tmc_subarraynode.commands.restart_command(
    method), 20
```

```
rc_t tmc_subarraynode::argcs()orLow
(ska_tmc_subarraynode::commands.release all resources comm
```

```
rs.SubarrayMethod(), AggregatorMid  
clean up mid resources()
```

```

        (ska_tmc_subarraynode.commands.release_all_resources_command)
        resources_command = ska_tmc_subarraynode.commands.release_all_resources_command
    except AttributeError:
        resources_command = None
    return resources_command

```

```

CommandNotAllowed, 32
completed() (ska_tmc_subarraynode.commands.obsreset_command.ObsResetCommand, 32)

```

```

    completed() (ska_tmc_sugar
resources_communications) 16 Resources
Configure(class in ska_tmc_sugar

```

## 12 `configureCommand` class in `sklearn.metrics._metrics`

(ska\_tmc\_subarraynode.commands.configure\_command.ConfigureCommand)

```

csp_subarray_dev_name
    (ska tmc subarro

```

## D

`devices(ska_tmc_subarraynode.model.component.SubarrayComponent  
property)`,<sup>28</sup>

device unresponsive, data

property), 31

```
do() (ska_tmc_subarraynode.commands.abstract_command.Subarray
      method), 7
do() (ska_tmc_subarraynode.commands.obsreset_command.ObsReset
      method), 7
```

```
do() (ska_tmc_subarraynode.commands.obsreset_command.ObsresetCommand.ElementData
method), 19
do() (ska_tmc_subarraynode.commands.reset_command.ResetCommand.ElementData
method), 19
```

do\_low() (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResources method), 9

do\_low() (ska\_tmc\_subarraynode.commands.configure\_command.Configure method), 12

do\_low() (ska\_tmc\_subarraynode.commands.end\_command.End method), 14

do\_low() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 15

do\_low() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset method), 16

do\_low() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 23

do\_low() (ska\_tmc\_subarraynode.commands.on\_command.On method), 24

do\_low() (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResources method), 18

do\_low() (ska\_tmc\_subarraynode.commands.reset\_command.Reset method), 19

do\_low() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

do\_low() (ska\_tmc\_subarraynode.commands.scan\_command.Scan method), 22

do\_mid() (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResources method), 10

do\_mid() (ska\_tmc\_subarraynode.commands.configure\_command.Configure method), 12

do\_mid() (ska\_tmc\_subarraynode.commands.end\_command.End method), 14

do\_mid() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 15

do\_mid() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset method), 16

do\_mid() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 23

do\_mid() (ska\_tmc\_subarraynode.commands.on\_command.On method), 24

do\_mid() (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResources method), 18

do\_mid() (ska\_tmc\_subarraynode.commands.reset\_command.Reset method), 19

do\_mid() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

do\_mid() (ska\_tmc\_subarraynode.commands.scan\_command.Scan method), 22

**E**

ElementDeviceData (class in ska\_tmc\_subarraynode.commands.configure\_command), 13

End (class in ska\_tmc\_subarraynode.commands.end\_command), 14

end\_csp() (ska\_tmc\_subarraynode.commands.end\_command.End method), 14

end\_scan() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 15

end\_scan\_mid() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 15

end\_sdp() (ska\_tmc\_subarraynode.commands.end\_command.End method), 14

EndScanCsp() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScanCsp method), 15

EndScanSdp() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScanSdp method), 15

**G**

get\_adapter\_by\_device\_name() (ska\_tmc\_subarraynode.commands.abstract\_command.Subarray method), 7

get\_csp\_subarray\_obsstate() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset method), 16

get\_csp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 23

get\_csp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

get\_device() (ska\_tmc\_subarraynode.model.component.SubarrayComponent method), 28

get\_dish\_adapter\_by\_device\_name() (ska\_tmc\_subarraynode.commands.abstract\_command.Subarray method), 7

get\_mccs\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

get\_sdp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 23

get\_sdp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

get\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 24

get\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 21

**H**

handle\_assigned\_resources\_event()

`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 26`  
`handle_csp_leaf_obs_state_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 26`  
`handle_lrcr_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 27`  
`handle_pointing_state_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 27`  
`handle_receive_addresses_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 27`  
`handle_sdp_leaf_obs_state_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 27`  
`handle_subsystem_availability_event()`  
`(ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEventReceiver`  
`method), 27`  
`HealthStateAggregatorLow` (class in `ska_tmc_subarraynode.manager.aggregators`),  
25  
`HealthStateAggregatorMid` (class in `ska_tmc_subarraynode.manager.aggregators`),  
25

## I

`identify_with_id()` (in module `ska_tmc_subarraynode.transaction_id`), 32  
`init_adapters()` (`ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand`  
`method), 8`  
`init_adapters_low()`  
`(ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand`  
`method), 8`  
`init_adapters_mid()`  
`(ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand`  
`method), 8`  
`inject_id()` (in module `ska_tmc_subarraynode.transaction_id`), 32  
`inject_with_id()` (in module `ska_tmc_subarraynode.transaction_id`), 32  
`InputParameter` (class in `ska_tmc_subarraynode.model.input`), 30  
`InputParameterLow` (class in `ska_tmc_subarraynode.model.input`), 31  
`InputParameterMid` (class in `ska_tmc_subarraynode.model.input`), 31  
`InvalidObsStateError`, 32  
`invoke_assign_resources()`  
`(ska_tmc_subarraynode.commands.assign_resources_command.AssignResources`  
`method), 11`  
`invoke_configure()` (`ska_tmc_subarraynode.commands.configure_command.Configure`  
`method), 13`

## M

`ska_tmc_subarraynode`, 33  
`ska_tmc_subarraynode.commands`, 25  
`ska_tmc_subarraynode.commands.abstract_command`,  
7  
`ska_tmc_subarraynode.commands.assign_resources_command`,  
8  
`ska_tmc_subarraynode.commands.configure_command`,  
7  
`ska_tmc_subarraynode.commands.end_command`,  
14  
`ska_tmc_subarraynode.commands.end_scan_command`,  
15  
`ska_tmc_subarraynode.commands.obsreset_command`,  
16  
`ska_tmc_subarraynode.commands.off_command`,  
23  
`ska_tmc_subarraynode.commands.on_command`,  
24  
`ska_tmc_subarraynode.commands.release_all_resources_co`  
17  
`ska_tmc_subarraynode.commands.reset_command`,  
19  
`ska_tmc_subarraynode.commands.restart_command`,  
20

[ska\\_tmc\\_subarraynode.commands.scan\\_command](#)[release\\_sdp\\_resources\(\)](#)  
[22](#) ([ska\\_tmc\\_subarraynode.commands.release\\_all\\_resources\\_command](#)  
[ska\\_tmc\\_subarraynode.exceptions](#), [32](#) [method](#)), [19](#)  
[ska\\_tmc\\_subarraynode.manager](#), [27](#) [ReleaseAllResources](#) (class in  
[ska\\_tmc\\_subarraynode.manager.aggregators](#), [ska\\_tmc\\_subarraynode.commands.release\\_all\\_resources\\_command](#)  
[25](#) [17](#)  
[ska\\_tmc\\_subarraynode.manager.event\\_receiver](#)[remove\\_device\(\)](#) ([ska\\_tmc\\_subarraynode.model.component.SubarrayComponent](#)  
[26](#) [method](#)), [28](#)  
[ska\\_tmc\\_subarraynode.model](#), [32](#) [Reset](#) (class in [ska\\_tmc\\_subarraynode.commands.reset\\_command](#)),  
[ska\\_tmc\\_subarraynode.model.component](#), [28](#) [19](#)  
[ska\\_tmc\\_subarraynode.model.enum](#), [30](#) [Restart](#) (class in [ska\\_tmc\\_subarraynode.commands.restart\\_command](#)),  
[ska\\_tmc\\_subarraynode.model.input](#), [30](#) [20](#)  
[ska\\_tmc\\_subarraynode.release](#), [32](#) [restart\\_csp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.restart\\_command.RestartCSP](#)  
[ska\\_tmc\\_subarraynode.transaction\\_id](#), [32](#) [method](#)), [21](#)  
[restart\\_dishes\(\)](#) ([ska\\_tmc\\_subarraynode.commands.restart\\_command.RestartDishes](#)  
[method](#)), [21](#)  
[restart\\_sdp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.restart\\_command.RestartSDP](#)  
[method](#)), [22](#)

## N

[NONE](#) ([ska\\_tmc\\_subarraynode.model.enum.PointingState](#)  
[attribute](#)), [30](#)

## O

[ObsReset](#) (class in [ska\\_tmc\\_subarraynode.commands.obsreset\\_command](#)),  
[16](#) [property](#)), [28](#)  
[obsreset\\_csp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.obsreset\\_command.ObsResetCSP](#)  
[method](#)), [17](#) [Scan](#) (class in [ska\\_tmc\\_subarraynode.commands.scan\\_command](#)),  
[obsreset\\_sdp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.obsreset\\_command.ObsResetSDP](#)  
[method](#)), [17](#) [SCAN](#) ([ska\\_tmc\\_subarraynode.model.enum.PointingState](#)  
[attribute](#)), [30](#)  
[obsreset\\_dish\(\)](#) ([ska\\_tmc\\_subarraynode.commands.obsreset\\_command.ObsResetDish](#)  
[method](#)), [17](#) [scan\\_csp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.scan\\_command.ScanCSP](#)  
[method](#)), [23](#)  
[ObsStateAggregatorLow](#) (class in [ska\\_tmc\\_subarraynode.manager.aggregators](#)),  
[25](#) [scan\\_csp\\_low\(\)](#) ([ska\\_tmc\\_subarraynode.commands.scan\\_command.ScanCSPLow](#)  
[method](#)), [23](#)  
[ObsStateAggregatorMid](#) (class in [ska\\_tmc\\_subarraynode.manager.aggregators](#)),  
[25](#) [scan\\_duration](#) ([ska\\_tmc\\_subarraynode.model.component.SubarrayComponent](#)  
[property](#)), [28](#)  
[Off](#) (class in [ska\\_tmc\\_subarraynode.commands.off\\_command](#)),  
[23](#) [scan\\_id](#) ([ska\\_tmc\\_subarraynode.model.component.SubarrayComponent](#)  
[property](#)), [29](#)  
[On](#) (class in [ska\\_tmc\\_subarraynode.commands.on\\_command](#)),  
[24](#) [scan\\_sdp\(\)](#) ([ska\\_tmc\\_subarraynode.commands.scan\\_command.ScanSDP](#)  
[method](#)), [23](#)  
[on\\_leaf\\_nodes\(\)](#) ([ska\\_tmc\\_subarraynode.commands.on\\_command.OnLeafNodes](#)  
[method](#)), [24](#) [sdp\\_subarray\\_dev\\_name](#)  
[property](#)), [30](#)  
[set\\_low\\_assigned\\_resources\(\)](#)  
[ska\\_tmc\\_subarraynode.commands.assign\\_resources\\_command.SetLowAssignedResources](#)  
[method](#)), [11](#)

## P

[PointingState](#) (class in [ska\\_tmc\\_subarraynode.model.enum](#)), [30](#)

## R

[READY](#) ([ska\\_tmc\\_subarraynode.model.enum.PointingState](#)  
[attribute](#)), [30](#)  
[reject\\_command\(\)](#) ([ska\\_tmc\\_subarraynode.commands.abstract\\_command.SubarrayNodeCommand](#)  
[method](#)), [8](#) [ska\\_tmc\\_subarraynode](#)  
[module](#), [33](#)  
[release\\_csp\\_resources\(\)](#) ([ska\\_tmc\\_subarraynode.commands](#)  
[ska\\_tmc\\_subarraynode.commands.release\\_all\\_resources\\_command.ReleaseAllResources](#)  
[method](#)), [18](#) [ska\\_tmc\\_subarraynode.commands.abstract\\_command](#)

module, 7	( <i>ska_tmc_subarraynode.model.component.SubarrayComponent</i>
<i>ska_tmc_subarraynode.commands.assign_resources_command</i>	property), 29
module, 8	<i>subarray_id</i> ( <i>ska_tmc_subarraynode.model.component.SubarrayComponent</i>
<i>ska_tmc_subarraynode.commands.configure_command</i>	property), 29
module, 12	<i>subarray_node_obstate_not_aggregated</i> ()
<i>ska_tmc_subarraynode.commands.end_command</i>	( <i>ska_tmc_subarraynode.manager.aggregators.ObsStateAggregator</i>
module, 14	method), 25
<i>ska_tmc_subarraynode.commands.end_scan_command</i>	<i>subarray_node_obstate_not_aggregated</i> ()
module, 15	( <i>ska_tmc_subarraynode.manager.aggregators.ObsStateAggregator</i>
<i>ska_tmc_subarraynode.commands.obsreset_command</i>	method), 26
module, 16	<i>subarray_off</i> () ( <i>ska_tmc_subarraynode.commands.off_command.Off</i>
<i>ska_tmc_subarraynode.commands.off_command</i>	method), 24
module, 23	<i>SubarrayAvailabilityAggregatorLow</i> (class in
<i>ska_tmc_subarraynode.commands.on_command</i>	<i>ska_tmc_subarraynode.manager.aggregators</i> ),
module, 24	26
<i>ska_tmc_subarraynode.commands.release_all_resources_command</i>	<i>SubarrayAvailabilityAggregatorMid</i> (class in
module, 17	<i>ska_tmc_subarraynode.manager.aggregators</i> ),
<i>ska_tmc_subarraynode.commands.reset_command</i>	26
module, 19	<i>SubarrayComponent</i> (class in
<i>ska_tmc_subarraynode.commands.restart_command</i>	<i>ska_tmc_subarraynode.model.component</i> ),
module, 20	28
<i>ska_tmc_subarraynode.commands.scan_command</i>	<i>SubarrayNodeCommand</i> (class in
module, 22	<i>ska_tmc_subarraynode.commands.abstract_command</i> ),
<i>ska_tmc_subarraynode.exceptions</i>	7
module, 32	<i>SubarrayNodeEventReceiver</i> (class in
<i>ska_tmc_subarraynode.manager</i>	<i>ska_tmc_subarraynode.manager.event_receiver</i> ),
module, 27	26
<i>ska_tmc_subarraynode.manager.aggregators</i>	<i>subscribe_events</i> () ( <i>ska_tmc_subarraynode.manager.event_receiver.Sub</i>
module, 25	method), 27
<i>ska_tmc_subarraynode.manager.event_receiver</i>	
module, 26	<b>T</b>
<i>ska_tmc_subarraynode.model</i>	<i>tmc_csp_sln_device_name</i>
module, 32	( <i>ska_tmc_subarraynode.model.input.InputParameter</i>
<i>ska_tmc_subarraynode.model.component</i>	property), 30
module, 28	<i>tmc_dish_ln_device_names</i>
<i>ska_tmc_subarraynode.model.enum</i>	( <i>ska_tmc_subarraynode.model.input.InputParameterMid</i>
module, 30	property), 31
<i>ska_tmc_subarraynode.model.input</i>	<i>tmc_sdp_sln_device_name</i>
module, 30	( <i>ska_tmc_subarraynode.model.input.InputParameter</i>
<i>ska_tmc_subarraynode.release</i>	property), 31
module, 32	<i>to_dict</i> () ( <i>ska_tmc_subarraynode.model.component.SubarrayComponent</i>
<i>ska_tmc_subarraynode.transaction_id</i>	method), 29
module, 32	<i>TRACK</i> ( <i>ska_tmc_subarraynode.model.enum.PointingState</i>
<i>SLEW</i> ( <i>ska_tmc_subarraynode.model.enum.PointingState</i>	attribute), 30
attribute), 30	<b>U</b>
<i>start_scan_timer</i> () ( <i>ska_tmc_subarraynode.commands.start_scan_command.Scan</i>	<i>UNKNOWN</i> ( <i>ska_tmc_subarraynode.model.enum.PointingState</i>
method), 23	attribute), 30
<i>stop_dish_tracking</i> ()	<i>unsubscribe_dish_health</i> ()
( <i>ska_tmc_subarraynode.commands.end_command.End</i>	( <i>ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEvent</i>
method), 15	method), 27
<i>subarray_availability</i>	<i>unsubscribe_dish_leaf_health</i> ()
( <i>ska_tmc_subarraynode.model.component.SubarrayComponent</i>	( <i>ska_tmc_subarraynode.manager.event_receiver.SubarrayNodeEvent</i>
property), 29	method), 27
<i>subarray_health_state</i>	



[unsubscribe\\_dish\\_leaf\\_state\(\)](#)  
 (ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver  
 method), 27

[unsubscribe\\_dish\\_pointing\(\)](#)  
 (ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver  
 method), 27

[unsubscribe\\_dish\\_state\(\)](#)  
 (ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver  
 method), 27

[unsubscribe\\_dish\\_events\(\)](#)  
 (ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver  
 method), 27

[unsubscribe\\_dish\\_leaf\\_events\(\)](#)  
 (ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver  
 method), 27

[update\(\)](#) (ska\_tmc\_subarraynode.model.input.InputParameter  
 method), 31

[update\(\)](#) (ska\_tmc\_subarraynode.model.input.InputParameterMid  
 method), 31

[update\\_command\\_in\\_progress\(\)](#)  
 (ska\_tmc\_subarraynode.commands.abstract\_command.SubarrayNodeCommand  
 method), 8

[update\\_device\(\)](#) (ska\_tmc\_subarraynode.model.component.SubarrayComponent  
 method), 29

[update\\_device\\_exception\(\)](#)  
 (ska\_tmc\_subarraynode.model.component.SubarrayComponent  
 method), 29

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResources  
 method), 11

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.configure\_command.Configure  
 method), 13

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.end\_command.End  
 method), 15

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan  
 method), 16

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.off\_command.Off  
 method), 24

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.on\_command.On  
 method), 25

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResources  
 method), 19

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.restart\_command.Restart  
 method), 22

[update\\_task\\_status\(\)](#)  
 (ska\_tmc\_subarraynode.commands.scan\_command.Scan  
 method), 23

[update\\_with\\_id\(\)](#) (in module  
 ska\_event\_receiver.SubarrayNodeEventReceiver.transaction\_id), 32

## V