

---

# **Subarray Node Documentation**

***Release 1.0***

**NCRA India**

**Mar 01, 2022**



# 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>37</b>
	<b>Python Module Index</b>	<b>39</b>
	<b>Index</b>	<b>41</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-5LBJH83', 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 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

```
class ska_tmc_subarraynode.commands.abstract_command.AbstractOnOff(*args: Any, **kwargs: Any)
    Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCCommand._name
    check_allowed_low()
        Checks whether this command is allowed It checks that the device is in a state to perform this command
        and that all the component needed for the operation are not unresponsive
        Returns True if this command is allowed
        Return type boolean
    check_allowed_mid()
        Checks whether this command is allowed It checks that the device is in a state to perform this command
        and that all the component needed for the operation are not unresponsive
        Returns True if this command is allowed
        Return type boolean
    init_adapters_low()
    init_adapters_mid()
class ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand(*args: Any,
                                                                            **kwargs: Any)
    Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCCommand._name
    check_allowed()
    do(argin=None)
    init_adapters()
    is_allowed(raise_if_disallowed=False)
    is_allowed_low(raise_if_disallowed)
```

`is_allowed_mid(raise_if_disallowed)`

### `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: `ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.AssignResourcesCommand._name`

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: ['0001', '0002']

**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_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(argin=None)`

`do_low(argin)`

Method to invoke AssignResources command on Subarraynode low.

**Parameters** `argin` – DevString in JSON form containing following fields: interface: Schema to allocate assign resources.

**mccs:** subarray\_beam\_ids: list of integers  
 station\_ids: list of integers  
 channel\_blocks: list of integers

Example:

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

**Returns** A tuple containing ResultCode and string.

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

Method to invoke AssignResources command on subarraynode mid.

**Parameters** *argin* – DevString.

Example:

```
{ "interface": "https://schema.skao.int/ska-tmc-assignresources/2.0", "subarray_id": 1,
  "dish": { "receptor_ids": ["0001"] }, "sdp": { "interface": "https://schema.skao.int/ska-sdp-assignres/0.3", "eb_id": "eb-mvp01-20200325-00001", "max_length": 100.0, "scan_types": [ { "scan_type_id": "science_A", "reference_frame": "ra": "02:42:40.771", "dec": "-00:00:47.84", "channels": [ { "count": 744, "start": 0, "stride": 2, "freq_min": 0.35e9, "freq_max": 0.368e9, "link_map": [[0,0],[200,1],[744,2],[944,3]] }, { "count": 744, "start": 2000, "stride": 1, "freq_min": 0.36e9, "freq_max": 0.368e9, "link_map": [[2000,4],[2200,5]] } ], { "scan_type_id": "calibration_B", "reference_frame": "ICRS", "ra": "12:29:06.699", "dec": "02:03:08.598", "channels": [ { "count": 744, "start": 0, "stride": 2, "freq_min": 0.35e9, "freq_max": 0.368e9, "link_map": [[0,0],[200,1],[744,2],[944,3]] }, { "count": 744, "start": 2000, "stride": 1, "freq_min": 0.36e9, "freq_max": 0.368e9, "link_map": [[2000,4],[2200,5]] } ] }, "processing_blocks": [ { "pb_id": "pb-mvp01-20200325-00001", "workflow": { "kind": "realtime", "name": "vis_receive", "version": "0.1.0", "parameters": {} }, { "pb_id": "pb-mvp01-20200325-00002", "workflow": { "kind": "realtime", "name": "test_realtime", "version": "0.1.0", "parameters": {} }, { "pb_id": "pb-mvp01-20200325-00003", "workflow": { "kind": "batch", "name": "ical", "version": "0.1.0", "parameters": {}, "dependencies": [ { "pb_id": "pb-mvp01-20200325-00001", "kind": [ "visibilities" ] } ], { "pb_id": "pb-mvp01-20200325-00004", "workflow": { "kind": "batch", "name": "dpreb", "version": "0.1.0", "parameters": {}, "dependencies": [ { "pb_id": "pb-mvp01-20200325-00003", "kind": [ "calibration" ] } ] } ] }
```

**Returns** A tuple containing a return code and string of Resources added to the Subarray. Example of string of Resources : ["0001", "0002"] 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 –

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

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

Method to check 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*)

Method to check whether this command is allowed 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

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

Adds the receptors in dish leaf node group. The healthState and pointingState attributes of 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: ['0001', '0002']

**Returns** List of Resources added to the Subarray. Example: ['0001', '0002']

## 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: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.ConfigureCommand.\_name

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*)

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

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

Method to invoke Configure command on the Mccs Subarray Leaf Node.

**Parameters** **argin** – DevString.

JSON string example is:

```
{“interface”:”https://schema.skao.int/ska-low-tmc-configure/2.0”,“transaction_id”:”txn-...-00001”,“mccs”:{“stations”:[{“station_id”:1},{“station_id”:2}],“subarray_beams”:[{“subarray_beam_id”:1,“station_ids”:[1,
```

**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 Aus-

```

    tralis","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)

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

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

Whether this command is allowed to run in the current state of the state model. :param *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*)

Method to check whether this command is allowed in the current state of the state model. :param *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

**class** `ska_tmc_subarraynode.commands.configure_command.ElementDeviceData`

Bases: `object`

**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":      "sci-
ence period 23"}, "common":{"config_id":"sbi-mvp01-20200325-00001-science_A",      "fre-
quency_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":{"sdp":{

```

**Returns** csp configuration schema

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

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

**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: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.EndCommand.\_name

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**(argin=None)

**do\_low**()

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**()

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

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

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

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

**stop\_dish\_tracking**()



**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: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.EndScanCommand._name
```

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**(*argin=None*)

**do\_low**()

Method to invoke Endscan command.

**Returns** None

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

**do\_mid**()

Method to invoke Endscan command.

**Returns** None

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

**end\_scan**()

**end\_scan\_low**()

**end\_scan\_mccs**()

set up mcs devices

**end\_scan\_mid**()

**endscan\_csp**()

set up csp devices

**endscan\_sdp**()

set up sdp devices

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

**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.obsreset\_command module**

ObsReset Command for SubarrayNode.

```
class ska_tmc_subarraynode.commands.obsreset_command.ObsReset(*args: Any, **kwargs: Any)
    Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.ObsResetCommand._name
```

A class for SubarrayNode's ObsReset() command.

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

**do**(argin=None)

**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**()

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

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

**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.

**obsrest\_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: `ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.ReleaseAllResourcesCommand._name`

A class for SKASubarray'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()**

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

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

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()**

Method to invoke ReleaseAllResources command.

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

**rtype:** (ResultCode, str)

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

Method to check whether this command is allowed 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

**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

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

Reset Command for SubarrayNode.

**class** `ska_tmc_subarraynode.commands.reset_command.Reset`(\*args: Any, \*\*kwargs: Any)

Bases: `ska_tango_base.SKABaseDevice`, `ska_tango_base.SKABaseDevice.ResetCommand`, `._name`

A class for SubarrayNode's Reset() command.

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

**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*)

**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: `ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.RestartCommand._name`

A class for SubarrayNode's Restart() command.

**clean\_up()**

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

**do()**

**do\_low()**

This command on Subarray Node Low invokes Restart command on MCCS 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)

**Raises Exception if error occurs while invoking command on MCCS Subarray Leaf Node.** –

**do\_mid()**

This method invokes Restart command on CSPSubarrayLeafNode, SDpSubarrayLeafNode and Dish-LeafNode.

**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()****get\_mccs\_subarray\_obstate()****get\_sdp\_subarray\_obstate()****get\_subarray\_obstate(dev\_name)****is\_allowed(raise\_if\_disallowed=True)****is\_allowed\_low(raise\_if\_disallowed)**

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

**restart\_csp()**

set up csp devices

**restart\_dishes()****restart\_sdp()**

set up sdp devices

**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: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.ScanCommand.\_name

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**(argin=None)

**do\_low**(argin)

Method to invoke Scan command.

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

JSON string example as follows:

```
{ "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 JIVE.

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

**rtype:** (ReturnCode, str)

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

**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

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

set up csp devices

**scan\_mccs**(*argin*)

set up mccs devices

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

set up sdp devices

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

**update\_mccs\_json**(*input\_scan*)

This Scan command input string is updated to send to MCCS SubarrayLeafNode.

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

```
class ska_tmc_subarraynode.commands.off_command.Off(*args: Any, **kwargs: Any)
    Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name
    A class for Subarraynode's Off() command.

    do_low(argin=None)
        Method to invoke off command on the MCCS Subarray Leaf Node. param: 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 CSP and SDP Subarray Leaf Nodes. param: None
        return: A tuple containing a return code and a string message indicating status.
        rtype: (ResultCode, str)

    get_csp_subarray_obstate()
    get_sdp_subarray_obstate()
    get_subarray_obstate(dev_name)
```

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

```
class ska_tmc_subarraynode.commands.on_command.On(*args: Any, **kwargs: Any)
    Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name
    A class for the SubarrayNode's On() command.

    do_low(argin=None)
        Method to invoke On command on MccsSubarrayLeafNode
        Raises Exception if the command execution is not successful –

    do_mid(argin=None)
        Method to invoke On command on CspSubarrayLeafNode and SdpSubarrayLeafNode.
        Raises DevFailed if the command execution is not successful –
```

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

```
class ska_tmc_subarraynode.commands.standby_command.Standby(*args: Any, **kwargs: Any)
    Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name
    A class for the SubarrayNodes's Standby() command.

    do_low(argin=None)
        Method to invoke Standby command on MCCSSubarrayLeafNode

    do_mid(argin=None)
        Method to invoke Standby command on CSPSubarrayLeafNode and SDPSubarrayLeafNode.
        Raises Exception if the command execution is not successful –

    get_command_object(command_name)
```



## Module contents

### ska\_tmc\_subarraynode.manager package

#### Submodules

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

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

Bases: ska\_tmc\_common.aggregators.ska\_tmc\_common.aggregators.Aggregator.\_name

**aggregate()**

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

Bases: ska\_tmc\_common.aggregators.ska\_tmc\_common.aggregators.Aggregator.\_name

**aggregate()**

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

Bases: ska\_tmc\_common.aggregators.ska\_tmc\_common.aggregators.Aggregator.\_name

**aggregate()**

Calculates aggregated observation state of Subarray.

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

Bases: ska\_tmc\_common.aggregators.ska\_tmc\_common.aggregators.Aggregator.\_name

**aggregate()**

Calculates aggregated observation state of Subarray.

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

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

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

Bases: ska\_tmc\_common.event\_receiver.ska\_tmc\_common.event\_receiver.EventReceiver.\_name

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?

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

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

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

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

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

```
unsubscribe_dish_leaf_state(dish_leaf_proxy, evt_id)
unsubscribe_dish_pointing(dish_proxy, evt_id)
unsubscribe_dish_state(dish_proxy, evt_id)
unsubscribe_dish_events()
unsubscribe_dish_leaf_events()
```

### `ska_tmc_subarraynode.manager.subarray_node_component_manager` module

This module provided a reference implementation of a `BaseComponentManager`.

It is provided for explanatory purposes, and to support testing of this package.

```
class ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager(*args:
                                                                                               Any,
                                                                                               **kwargs
                                                                                               Any)
```

Bases: `ska_tango_base.subarray.ska_tango_base.subarray.SubarrayComponentManager._name`

A component manager for The Subarray Node component.

It supports:

- Monitoring its component, e.g. detect that it has been turned off or on
- Fetching the latest SCM indicator values of the components periodically and trigger the subarray health state and observation state aggregation
- Receiving the change events from the subarray component and trigger the subarray health state and observation state aggregation

**adapter\_error\_message\_result**(*dev\_name*, *e*)

**add\_device**(*dev\_name*)

Add device to the monitoring loop

**Parameters** *dev\_name* (*str*) – device name

**add\_multiple\_devices**(*device\_list*)

Add multiple devices to the monitoring loop

**Parameters** *device\_list* – list of device names

**property assigned\_resources**

Return the resources assigned to the component.

**Returns** the resources assigned to the component

**Return type** list of *str*

**check\_command\_not\_allowed\_exception**(*op\_state*, *states\_not\_allowed*, *cmd\_name*)

**check\_device\_unresponsive\_exception**(*dev\_name*)

**check\_if\_no\_dishes\_available**()

**property checked\_devices**

Return the list of the checked monitored devices

**Returns** list of the checked monitored devices

**property command\_executed**

**property** `command_executor`

**property** `command_in_progress`

**device\_failed**(*device\_info*, *exception*)

Set a device to failed and call the relative callback if available

**Parameters**

- **device\_info** (*DeviceInfo*) – a device info
- **exception** – an exception

**Type** Exception

**property** `devices`

Return the list of the monitored devices

**Returns** list of the monitored devices

**generate\_command\_result**(*result\_code*, *message*)

**get\_assigned\_resources**()

**get\_csp\_subarray\_dev\_name**()

Return Csp Subarray device name

**get\_dev\_info**(*dev\_name*)

**get\_device**(*dev\_name*)

Return the device info our of the monitoring loop with name *dev\_name*

**Parameters** **dev\_name** (*str*) – name of the device

**Returns** a device info

**Return type** DeviceInfo

**get\_dish\_dev\_names**()

Return the names of the dishes assigned to Subarray

**get\_dish\_leaf\_prefix**()

Return the Dish Leaf Node prefix

**get\_mccs\_subarray\_dev\_name**()

Returns Mccs Subarray device name

**get\_sb\_id**()

**get\_scan\_duration**()

**get\_scan\_id**()

**get\_sdp\_subarray\_dev\_name**()

Return Sdp Subarray device name

**get\_tm\_leaf\_csp\_subarray\_dev\_name**()

Return Csp Subarray Leaf Node device name

**get\_tm\_leaf\_dish\_dev\_names**()

Return the names of the dish leaf nodes assigned to Subarray

**init\_adapters**(*adapter\_factory*)

**init\_adapters\_low**(*adapter\_factory*)

**init\_adapters\_mid**(*adapter\_factory*)

**property input\_parameter**

Return the input parameter

**Returns** input parameter

**Return type** *InputParameter*

**is\_scan\_timer\_running()****property last\_command\_executed****remove\_dish\_devices()****reset()****set\_assigned\_resources(resources=[])**

For SubarrayNode Mid, set assigned\_resources with the list of dishes in argin. :param dev\_name: name of the dish devices :type list: list[str]

**set\_dish\_dev\_names(dish\_dev\_names)**

Update the names of the dishes assigned to Subarray

**Parameters** **dev\_name** – name of the dish devices

**set\_sb\_id(sb\_id)****set\_scan\_duration(scan\_duration)****set\_scan\_id(scan\_id)****set\_tmc\_leaf\_dish\_dev\_names(tm\_leaf\_dish\_dev\_names)**

Update the names of the dish leaf nodes assigned to Subarray

**Parameters** **dev\_name** – name of the dish devices

**stop()****stop\_scan\_timer()****unsubscribe\_dish\_events()****update\_assigned\_resources\_low()****update\_device\_health\_state(dev\_name, health\_state)**

Update a monitored device health state aggregate the health states available

**Parameters**

- **dev\_name** (*str*) – name of the device
- **health\_state** (*HealthState*) – health state of the device

**update\_device\_info(device\_info)**

Update a device with correct monitoring information and call the relative callback if available

**Parameters** **device\_info** (*DeviceInfo*) – a device info

**update\_device\_obs\_state(dev\_name, obs\_state)**

Update a monitored device obs state, and call the relative callbacks if available

**Parameters**

- **dev\_name** (*str*) – name of the device
- **obs\_state** (*ObsState*) – obs state of the device

**update\_device\_pointing\_state(dev\_name, pointing\_state)**

Update a monitored device pointing state aggregate the Subarray obs states and Dish pointing states

**Parameters**

- **dev\_name** (*str*) – name of the device
- **pointing\_state** ([PointingState](#)) – pointing state of the device

**update\_device\_state**(*dev\_name, state*)

Update a monitored device state, aggregate the states available and call the relative callbacks if available

**Parameters**

- **dev\_name** (*str*) – name of the device
- **state** (*DevState*) – state of the device

**update\_event\_failure**(*dev\_name*)

**update\_input\_parameter**()

**update\_receive\_addresses**(*dev\_name, receive\_addresses*)

Update receiveAddresses for a monitored device

**Parameters**

- **dev\_name** (*str*) – name of the device
- **receive\_addresses** (*str*) – receiveAddresses

**ska\_tmc\_subarraynode.manager.monitoring\_loop module**

```
class ska_tmc_subarraynode.manager.monitoring_loop.SubarrayNodeMonitoringLoop(*args: Any,
                                                                              **kwargs:
                                                                              Any)
```

Bases: `ska_tmc_common.monitoring_loop.ska_tmc_common.monitoring_loop.MonitoringLoop._name`

The SubarrayNodeMonitoringLoop class has the responsibility to monitor the sub devices managed by the sub-array node.

It is an infinite loop which ping, get the state, the obsState, the healthState and device information of the monitored SKA devices

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

**create\_device\_info**(*dev\_info, proxy*)

**device\_task**(*dev\_info*)

**Module contents****ska\_tmc\_subarraynode.model package****Submodules****ska\_tmc\_subarraynode.model.component module**

```
class ska_tmc_subarraynode.model.component.DishDeviceInfo(*args: Any, **kwargs: Any)
```

Bases: `ska_tmc_common.device_info.ska_tmc_common.device_info.DeviceInfo._name`

**from\_dev\_info**(*dishDeviceInfo*)

`to_dict()`

`to_json()`

**class** `ska_tmc_subarraynode.model.component.SdpSubarrayDeviceInfo(*args: Any, **kwargs: Any)`  
Bases: `ska_tmc_common.device_info.ska_tmc_common.device_info.SubArrayDeviceInfo._name`

**from\_dev\_info**(*sdpSubarrayDeviceInfo*)

`to_dict()`

`to_json()`

**class** `ska_tmc_subarraynode.model.component.SubarrayComponent(*args: Any, **kwargs: Any)`  
Bases: `ska_tmc_common.tmc_component_manager.ska_tmc_common.tmc_component_manager.TmcComponent._name`

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**(*dev\_name*)

Return the monitored device info by name.

**Parameters** *dev\_name* – name of the device

**Returns** the monitored device info

**Return type** DeviceInfo

**remove\_device**(*dev\_name*)

Remove a device from the list

**Parameters** *dev\_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*)

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

**property subarray\_health\_state**

Return the aggregated subarray health state

**Returns** the subarray health state

**Return type** HealthState

**to\_dict**()

**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**

**class** ska\_tmc\_subarraynode.model.enum.**PointingState**(*value*)

Bases: enum.IntEnum

An enumeration.

**NONE** = 0

**READY** = 1

**SCAN** = 4

**SLEW** = 2

**TRACK** = 3

**UNKNOWN** = 5

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

**class** ska\_tmc\_subarraynode.model.input.**InputParameter**(*changed\_callback*)

Bases: object

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

**class** ska\_tmc\_subarraynode.model.input.**InputParameterLow**(*changed\_callback*)

Bases: [ska\\_tmc\\_subarraynode.model.input.InputParameter](#)

**property mccs\_subarray\_dev\_name**

Return the MCCS Subarray device name

**Returns** the MCCS Subarray device name

**Return type** str

**property** `tm_leaf_mccs_subarray_dev_name`

Return the TM Leaf MCCS Subarray device name

**Returns** the TM Leaf MCCS Subarray device name

**Return type** str

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

**class** `ska_tmc_subarraynode.model.input.InputParameterMid(changed_callback)`

Bases: `ska_tmc_subarraynode.model.input.InputParameter`

**property** `csp_subarray_dev_name`

Return the CSP Subarray device name

**Returns** the CSP Subarray device name

**Return type** str

**property** `dish_dev_names`

Return the dish device names

**Returns** the TM dish device names

**Return type** list

**property** `sdp_subarray_dev_name`

Returns the SDP Subarray device name

**Returns** the SDP Subarray device name

**Return type** str

**property** `tm_leaf_csp_subarray_dev_name`

Return the CSP Subarray Leaf Node device names

**Returns** the CSP Subarray Leaf Node device names

**Return type** str

**property** `tm_leaf_dish_dev_names`

Return the TM dish device names

**Returns** the TM dish device names

**Return type** list

**property** `tm_leaf_sdp_subarray_dev_name`

Return the SDP Subarray Leaf Node device names

**Returns** the SDP Subarray Leaf Node device names

**Return type** str

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



## Module contents

### 3.1.2 Submodules

### 3.1.3 ska\_tmc\_subarraynode.\_subarraynode\_node module

Subarray Node Provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray.

**class** ska\_tmc\_subarraynode.subarray\_node.**AbstractSubarrayNode**(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.ska\_tango\_base.SKASubarray.\_name

Provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray.

#### Device Attributes

**scanID:** ID of ongoing SCAN

**sbID:** ID of ongoing Scheduling Block

**class** InitCommand(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.InitCommand.\_name

A class for the TMC SubarrayNode's init\_device() method.

**do()**

Initializes the attributes and properties of the Subarray Node.

**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 error while subscribing the tango attribute –**

**Off()**

Invokes Off command on SubarrayNode

**On()**

Invokes On command on SubarrayNode

**Standby()**

Invokes Standby command on SubarrayNode

**always\_executed\_hook()**

Internal construct of TANGO.

**create\_component\_manager()**

**delete\_device()**

**enqueue\_command**(command\_name, args=None)

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**is\_Abort\_allowed()**

Check if command *Abort* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_AssignResources\_allowed()**

Check if command *AssignResources* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_Configure\_allowed()**

Check if command *Configure* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_EndScan\_allowed()**

Check if command *EndScan* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_End\_allowed()**

Check if command *End* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_ObsReset\_allowed()**

Check if command *ObsReset* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_Off\_allowed()**

Checks whether the command is allowed to be run in the current state

**Returns** True if this command is allowed to be run in current device state

**Return type** boolean

**Raises** DevFailed if this command is not allowed to be run in current device state

**is\_On\_allowed()**

Checks whether the command is allowed to be run in the current state

**Returns** True if this command is allowed to be run in current device state

**Return type** boolean

**Raises** DevFailed if this command is not allowed to be run in current device state

**is\_ReleaseAllResources\_allowed()**

Check if command *ReleaseAllResources* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_Reset\_allowed()**

Whether the `Reset()` command is allowed to be run in the current state.

**Returns** whether the `Reset()` command is allowed to be run in the current state

**Return type** boolean

**is\_Restart\_allowed()**

Check if command *Restart* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_Scan\_allowed()**

Check if command *Scan* is allowed in the current device state.

**Returns** True if the command is allowed

**Return type** boolean

**is\_Standby\_allowed()**

Checks whether the command is allowed to be run in the current state

**Returns** True if this command is allowed to be run in current device state

**Return type** boolean

**Raises** DevFailed if this command is not allowed to be run in current device state

**read\_commandExecuted()**

Return the commandExecuted attribute.

**read\_commandInProgress()****read\_internalModel()****read\_lastCommandExecuted()**

Return the lastCommandExecuted attribute as list of string.

**read\_lastDeviceInfoChanged()****read\_sbID()**

Internal construct of TANGO. Returns the scheduling block ID.

**read\_scanID()**

Internal construct of TANGO. Returns the Scan ID.

EXAMPLE: 123 Where 123 is a Scan ID from configuration json string.

**read\_subarrayHealthState()****read\_transformedInternalModel()****update\_assigned\_resources\_callback(*assigned\_resources*)****update\_command\_in\_progress\_callback(*command\_in\_progress*)****update\_device\_callback(*dev\_info*)****update\_subarray\_health\_state\_callback(*subarray\_health\_state*)**

### 3.1.4 ska\_tmc\_subarraynode.\_subarraynode\_node\_low module

Subarray Node Low provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray Low.

**class** ska\_tmc\_subarraynode.subarray\_node\_low.**SubarrayNodeLow**(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.ska\_tango\_base.SKASubarray.\_name

Provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray.

#### Device Properties

**MccsSubarrayLNFQDN:** This property contains the FQDN of the MCCS Subarray Leaf Node associated with the Subarray Node.

**MccsSubarrayFQDN:** This property contains the FQDN of the MCCS Subarray associated with the Subarray Node.

#### Device Attributes

**class** InitCommand(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.InitCommand.\_name

A class for the TMC SubarrayNodeMid's init\_device() method.

**do()**

Initializes the attributes and properties of the Subarray Node Mid.

**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 error while subscribing the tango attribute –**

**create\_component\_manager()**

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**read\_mccsSubarrayDevName()**

Return the mccsSubarrayDevName attribute.

**read\_tmcLeafMccsSubarrayDevName()**

Return the tmcLeafMccsSubarrayDevName attribute.

**write\_mccsSubarrayDevName(value)**

Set the mccsSubarrayDevName attribute.

**write\_tmcLeafMccsSubarrayDevName(value)**

Set the tmcLeafMccsSubarrayDevName attribute.

ska\_tmc\_subarraynode.subarray\_node\_low.**main**(args=None, \*\*kwargs)

Runs the SubarrayNodeLow. :param args: Arguments internal to TANGO :param kwargs: Arguments internal to TANGO :return: SubarrayNodeLow TANGO object.

### 3.1.5 ska\_tmc\_subarraynode.\_subarraynode\_node\_mid module

Subarray Node Mid provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray Mid.

**class** ska\_tmc\_subarraynode.subarray\_node\_mid.**SubarrayNodeMid**(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.ska\_tango\_base.SKASubarray.\_name

Provides the monitoring and control interface required by users as well as other TM Components (such as OET, Central Node) for a Subarray.

#### Device Properties

**SdpSubarrayLNFQDN:** This property contains the FQDN of the SDP Subarray Leaf Node associated with the Subarray Node.

**CspSubarrayLNFQDN:** This property contains the FQDN of the CSP Subarray Leaf Node associated with the Subarray Node.

**DishLeafNodePrefix:** Device name prefix for the Dish Leaf Node.

**CspSubarrayFQDN:** FQDN of the CSP Subarray Tango Device Server.

**SdpSubarrayFQDN:** FQDN of the CSP Subarray Tango Device Server.

#### Device Attributes

**class** InitCommand(\*args: Any, \*\*kwargs: Any)

Bases: ska\_tango\_base.SKASubarray.ska\_tango\_base.SKASubarray.InitCommand.\_name

A class for the TMC SubarrayNodeMid's init\_device() method.

**do()**

Initializes the attributes and properties of the Subarray Node Mid.

**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 error while subscribing the tango attribute –

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**read\_cspSubarrayDevName()**

Return the cspsubarraydevname attribute.

**read\_dishDevNames()**

Return the dishdevnames attribute.

**read\_sdpSubarrayDevName()**

Return the sdpsubarraydevname attribute.

**read\_tmcLeafCspSubarrayDevName()**

Return the tmcLeafCspSubarrayDevName attribute.

**read\_tmcLeafDishDevNames()**

Return the tmcLeafDishDevNames attribute.

**read\_tmcLeafSdpSubarrayDevName()**

Return the tmcLeafSdpSubarrayDevName attribute.

**write\_cspSubarrayDevName(value)**

Set the cspsubarraydevname attribute.

**write\_dishDevNames**(*value*)

Set the dishdevnames attribute.

**write\_sdpSubarrayDevName**(*value*)

Set the sdpSubarrayDevName attribute.

**write\_tmcLeafCspSubarrayDevName**(*value*)

Set the tmcLeafCspSubarrayDevName attribute.

**write\_tmcLeafDishDevNames**(*value*)

Set the tmcLeafDishDevNames attribute.

**write\_tmcLeafSdpSubarrayDevName**(*value*)

Set the tmcLeafSdpSubarrayDevName attribute.

**ska\_tmc\_subarraynode.subarray\_node\_mid.main**(*args=None, \*\*kwargs*)

Runs the SubarrayNode. :param args: Arguments internal to TANGO :param kwargs: Arguments internal to TANGO :return: SubarrayNode TANGO object.

### 3.1.6 ska\_tmc\_subarraynode.exceptions module

**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

**ska\_tmc\_subarraynode.transaction\_id.identify\_with\_id**(*name: str, arg\_name: str*)

**ska\_tmc\_subarraynode.transaction\_id.inject\_id**(*obj, data: Dict*) → Dict

**ska\_tmc\_subarraynode.transaction\_id.inject\_with\_id**(*arg\_position: int, arg\_name: str*)

**ska\_tmc\_subarraynode.transaction\_id.update\_with\_id**(*obj, parameters: Any*) → Union[Dict, str]

### 3.1.9 Module contents





## INDICES AND TABLES

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



## PYTHON MODULE INDEX

**S**

- [ska\\_tmc\\_subarraynode](#), 35
- [ska\\_tmc\\_subarraynode.commands](#), 21
- [ska\\_tmc\\_subarraynode.commands.abstract\\_command](#), 7
- [ska\\_tmc\\_subarraynode.commands.assign\\_resources\\_command](#), 8
- [ska\\_tmc\\_subarraynode.commands.configure\\_command](#), 10
- [ska\\_tmc\\_subarraynode.commands.end\\_command](#), 12
- [ska\\_tmc\\_subarraynode.commands.end\\_scan\\_command](#), 13
- [ska\\_tmc\\_subarraynode.commands.obsreset\\_command](#), 14
- [ska\\_tmc\\_subarraynode.commands.off\\_command](#), 20
- [ska\\_tmc\\_subarraynode.commands.on\\_command](#), 20
- [ska\\_tmc\\_subarraynode.commands.release\\_all\\_resources\\_command](#), 15
- [ska\\_tmc\\_subarraynode.commands.reset\\_command](#), 16
- [ska\\_tmc\\_subarraynode.commands.restart\\_command](#), 17
- [ska\\_tmc\\_subarraynode.commands.scan\\_command](#), 18
- [ska\\_tmc\\_subarraynode.commands.standby\\_command](#), 20
- [ska\\_tmc\\_subarraynode.exceptions](#), 34
- [ska\\_tmc\\_subarraynode.manager](#), 25
- [ska\\_tmc\\_subarraynode.manager.aggregators](#), 21
- [ska\\_tmc\\_subarraynode.manager.event\\_receiver](#), 21
- [ska\\_tmc\\_subarraynode.manager.monitoring\\_loop](#), 25
- [ska\\_tmc\\_subarraynode.manager.subarray\\_node\\_component\\_manager](#), 22
- [ska\\_tmc\\_subarraynode.model](#), 29
- [ska\\_tmc\\_subarraynode.model.component](#), 25
- [ska\\_tmc\\_subarraynode.model.enum](#), 27
- [ska\\_tmc\\_subarraynode.model.input](#), 27
- [ska\\_tmc\\_subarraynode.release](#), 34
- [ska\\_tmc\\_subarraynode.subarray\\_node](#), 29
- [ska\\_tmc\\_subarraynode.subarray\\_node\\_low](#), 32
- [ska\\_tmc\\_subarraynode.subarray\\_node\\_mid](#), 33
- [ska\\_tmc\\_subarraynode.transaction\\_id](#), 34



## INDEX

### A

**AbstractOnOff** (class in `build_up_csp_cmd_data()`  
     (`ska_tmc_subarraynode.commands.abstract_command`),  
     7  
     static method), 11

**AbstractSubarrayNode** (class in `build_up_dsh_cmd_data()`  
     (`ska_tmc_subarraynode.subarray_node`),  
     29  
     static method), 11

**AbstractSubarrayNode.InitCommand** (class in `build_up_sdp_cmd_data()`  
     (`ska_tmc_subarraynode.subarray_node`), 29  
     static method), 11

**adapter\_error\_message\_result()**  
     (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**add\_device()** (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**add\_multiple\_devices()**  
     (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**aggregate()** (`ska_tmc_subarraynode.manager.aggregators.HealthStateAggregatorLow`  
     method), 21

**aggregate()** (`ska_tmc_subarraynode.manager.aggregators.HealthStateAggregatorMid`  
     method), 21

**aggregate()** (`ska_tmc_subarraynode.manager.aggregators.ObsStateAggregatorLow`  
     method), 21

**aggregate()** (`ska_tmc_subarraynode.manager.aggregators.ObsStateAggregatorMid`  
     method), 21

**always\_executed\_hook()**  
     (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode`  
     method), 29

**assign\_csp\_resources()**  
     (`ska_tmc_subarraynode.commands.assign_resources_command.AssignResources`  
     method), 8

**assign\_sdp\_resources()**  
     (`ska_tmc_subarraynode.commands.assign_resources_command.AssignResources`  
     method), 8

**assigned\_resources** (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22

**assigned\_resources** (`ska_tmc_subarraynode.model.component.SubarrayComponent`  
     property), 26

**AssignResources** (class in `command_executed` (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22  
     8  
     command\_executor (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22

### B

**build\_up\_csp\_cmd\_data()**  
     (`ska_tmc_subarraynode.commands.configure_command.ElementaryCommand`  
     static method), 11

**build\_up\_dsh\_cmd\_data()**  
     (`ska_tmc_subarraynode.commands.configure_command.ElementaryCommand`  
     static method), 11

**build\_up\_sdp\_cmd\_data()**  
     (`ska_tmc_subarraynode.commands.configure_command.ElementaryCommand`  
     static method), 11

**check\_allowed()** (`ska_tmc_subarraynode.commands.abstract_command.AbstractCommand`  
     method), 7

**check\_allowed\_low()**  
     (`ska_tmc_subarraynode.commands.abstract_command.AbstractCommand`  
     method), 7

**check\_allowed\_mid()**  
     (`ska_tmc_subarraynode.commands.abstract_command.AbstractCommand`  
     method), 7

**check\_command\_not\_allowed\_exception()**  
     (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**check\_device\_unresponsive\_exception()**  
     (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**check\_if\_no\_dishes\_available()**  
     (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     method), 22

**check\_only\_dish\_config()**  
     (`ska_tmc_subarraynode.commands.configure_command.ConfigureDishConfig`  
     method), 10

**checked\_devices** (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22

**clean\_up()** (`ska_tmc_subarraynode.commands.release_all_resources_command.ReleaseAllResources`  
     method), 15

**clean\_up()** (`ska_tmc_subarraynode.commands.restart_command.RestartCommand`  
     method), 17

**command\_executed** (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22

**command\_executor** (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
     property), 22

`command_in_progress` `do()` (`ska_tmc_subarraynode.commands.scan_command.Scan`  
`(ska_tmc_subarraynode.manager.subarray_node_component_manager`  
`property)`, 23 `do()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode.InitC`  
`CommandNotAllowed`, 34 `method)`, 29  
`Configure` (`class` `in` `ska_tmc_subarraynode.commands.configure_command.Configure`  
10 `method)`, 32  
`create_component_manager()` `do()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid.InitC`  
`(ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode`  
`method)`, 29 `do_low()` (`ska_tmc_subarraynode.commands.assign_resources_command.`  
`create_component_manager()` `method)`, 8  
`(ska_tmc_subarraynode.subarray_node_low.SubarrayNodeLow`  
`method)`, 32 `do_low()` (`ska_tmc_subarraynode.commands.configure_command.Configure`  
`method)`, 10  
`create_device_info()` `do_low()` (`ska_tmc_subarraynode.commands.end_command.End`  
`(ska_tmc_subarraynode.manager.monitoring_loop.SubarrayNodeMonitoringLoop`  
`method)`, 25 `do_low()` (`ska_tmc_subarraynode.commands.end_scan_command.EndScan`  
`csp_subarray_dev_name` `method)`, 13  
`(ska_tmc_subarraynode.model.input.InputParameterMid`  
`property)`, 28 `do_low()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsRese`  
`method)`, 14  
**D** `do_low()` (`ska_tmc_subarraynode.commands.off_command.Off`  
`method)`, 20  
`delete_device()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode`  
`method)`, 29 `do_low()` (`ska_tmc_subarraynode.commands.on_command.On`  
`method)`, 20  
`device_failed()` (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method)`, 23 `do_low()` (`ska_tmc_subarraynode.commands.release_all_resources_comm`  
`method)`, 15  
`device_task()` (`ska_tmc_subarraynode.manager.monitoring_loop.SubarrayNodeMonitoringLoop`  
`method)`, 25 `do_low()` (`ska_tmc_subarraynode.commands.reset_command.Reset`  
`method)`, 16  
`devices` (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`property)`, 23 `do_low()` (`ska_tmc_subarraynode.commands.restart_command.Restart`  
`method)`, 17  
`devices` (`ska_tmc_subarraynode.model.component.SubarrayComponent`  
`property)`, 26 `do_low()` (`ska_tmc_subarraynode.commands.scan_command.Scan`  
`method)`, 18  
`DeviceUnresponsive`, 34 `do_low()` (`ska_tmc_subarraynode.commands.standby_command.Standby`  
`method)`, 20  
`dish_dev_names` (`ska_tmc_subarraynode.model.input.InputParameterMid`  
`property)`, 28 `do_mid()` (`ska_tmc_subarraynode.commands.assign_resources_command.`  
`DishDeviceInfo` (`class` `in` `method)`, 9  
`ska_tmc_subarraynode.model.component)`, 25 `do_mid()` (`ska_tmc_subarraynode.commands.configure_command.Configure`  
`method)`, 10  
`do()` (`ska_tmc_subarraynode.commands.abstract_command.SubarrayNodeCommand`  
`method)`, 7 `do_mid()` (`ska_tmc_subarraynode.commands.end_command.End`  
`method)`, 12  
`do()` (`ska_tmc_subarraynode.commands.assign_resources_command.AssignResources`  
`method)`, 8 `do_mid()` (`ska_tmc_subarraynode.commands.end_scan_command.EndScan`  
`method)`, 13  
`do()` (`ska_tmc_subarraynode.commands.configure_command.Configure`  
`method)`, 10 `do_mid()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsRese`  
`method)`, 14  
`do()` (`ska_tmc_subarraynode.commands.end_command.End`  
`method)`, 12 `do_mid()` (`ska_tmc_subarraynode.commands.off_command.Off`  
`method)`, 20  
`do()` (`ska_tmc_subarraynode.commands.end_scan_command.EndScan`  
`method)`, 13 `do_mid()` (`ska_tmc_subarraynode.commands.on_command.On`  
`method)`, 20  
`do()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsReset`  
`method)`, 14 `do_mid()` (`ska_tmc_subarraynode.commands.release_all_resources_comm`  
`method)`, 15  
`do()` (`ska_tmc_subarraynode.commands.release_all_resources_command.ReleaseAllResources`  
`method)`, 15 `do_mid()` (`ska_tmc_subarraynode.commands.reset_command.Reset`  
`method)`, 16  
`do()` (`ska_tmc_subarraynode.commands.reset_command.Reset`  
`method)`, 16 `do_mid()` (`ska_tmc_subarraynode.commands.restart_command.Restart`  
`method)`, 17  
`do()` (`ska_tmc_subarraynode.commands.restart_command.Restart`  
`method)`, 17 `do_mid()` (`ska_tmc_subarraynode.commands.scan_command.Scan`  
`method)`, 19

do\_mid() (ska\_tmc\_subarraynode.commands.standby\_command.Standby method), 20

**E**

ElementDeviceData (class in ska\_tmc\_subarraynode.commands.configure\_command), 11

End (class in ska\_tmc\_subarraynode.commands.end\_command), 12

end\_csp() (ska\_tmc\_subarraynode.commands.end\_command.End method), 12

end\_scan() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

end\_scan\_low() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

end\_scan\_mccs() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

end\_scan\_mid() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

end\_sdp() (ska\_tmc\_subarraynode.commands.end\_command.End method), 12

EndScan (class in ska\_tmc\_subarraynode.commands.end\_scan\_command), 13

endscan\_csp() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

endscan\_sdp() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScan method), 13

enqueue\_command() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode method), 29

**F**

from\_dev\_info() (ska\_tmc\_subarraynode.model.component.DishDeviceInfo method), 25

from\_dev\_info() (ska\_tmc\_subarraynode.model.component.SdpSubarrayDeviceInfo method), 26

**G**

generate\_command\_result() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_assigned\_resources() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_command\_object() (ska\_tmc\_subarraynode.commands.standby\_command.Standby method), 20

get\_csp\_subarray\_dev\_name() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_csp\_subarray\_obsstate() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset method), 14

get\_csp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 20

get\_csp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 18

get\_dev\_info() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_device() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_device() (ska\_tmc\_subarraynode.model.component.SubarrayComponent method), 26

get\_dish\_dev\_names() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_dish\_leaf\_prefix() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_mccs\_subarray\_dev\_name() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_mccs\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 18

get\_sb\_id() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_scan\_duration() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_scan\_id() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_sdp\_subarray\_dev\_name() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_sdp\_subarray\_obsstate() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset method), 14

get\_sdp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 20

get\_sdp\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 18

get\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.off\_command.Off method), 20

get\_subarray\_obstate() (ska\_tmc\_subarraynode.commands.restart\_command.Restart method), 18

get\_tm\_leaf\_csp\_subarray\_dev\_name() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

get\_tm\_leaf\_dish\_dev\_names() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager method), 23

method), 23

## H

handle\_pointing\_state\_event()  
(ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventManager  
method), 21

handle\_receive\_addresses\_event()  
(ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventManager  
method), 21

HealthStateAggregatorLow (class in  
ska\_tmc\_subarraynode.manager.aggregators),  
21

HealthStateAggregatorMid (class in  
ska\_tmc\_subarraynode.manager.aggregators),  
21

## I

identify\_with\_id() (in module  
ska\_tmc\_subarraynode.transaction\_id), 34

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

init\_adapters() (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager  
method), 23

init\_adapters\_low()  
(ska\_tmc\_subarraynode.commands.abstract\_command.AbstractCommand  
method), 7

init\_adapters\_low()  
(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager  
method), 23

init\_adapters\_mid()  
(ska\_tmc\_subarraynode.commands.abstract\_command.AbstractCommand  
method), 7

init\_adapters\_mid()  
(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager  
method), 23

init\_command\_objects()  
(ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
method), 29

init\_command\_objects()  
(ska\_tmc\_subarraynode.subarray\_node\_low.SubarrayNodeLow  
method), 32

init\_command\_objects()  
(ska\_tmc\_subarraynode.subarray\_node\_mid.SubarrayNodeMid  
method), 33

inject\_id() (in module  
ska\_tmc\_subarraynode.transaction\_id), 34

inject\_with\_id() (in module  
ska\_tmc\_subarraynode.transaction\_id), 34

input\_parameter (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager  
property), 23

InputParameter (class in  
ska\_tmc\_subarraynode.model.input), 27

InputParameterLow (class in  
ska\_tmc\_subarraynode.model.input), 27

InputParameterMid (class in  
ska\_tmc\_subarraynode.model.input), 28

InvalidObsStateError, 34

is\_Abort\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
method), 29

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

is\_allowed() (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResourcesCommand  
method), 9

is\_allowed() (ska\_tmc\_subarraynode.commands.configure\_command.ConfigureCommand  
method), 11

is\_allowed() (ska\_tmc\_subarraynode.commands.end\_command.EndCommand  
method), 12

is\_allowed() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScanCommand  
method), 13

is\_allowed() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsResetCommand  
method), 14

is\_allowed() (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResourcesCommand  
method), 15

is\_allowed() (ska\_tmc\_subarraynode.commands.reset\_command.ResetCommand  
method), 16

is\_allowed() (ska\_tmc\_subarraynode.commands.restart\_command.RestartCommand  
method), 18

is\_allowed() (ska\_tmc\_subarraynode.commands.scan\_command.ScanCommand  
method), 19

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.abstract\_command.AbstractCommand  
method), 7

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResourcesCommand  
method), 9

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.configure\_command.ConfigureCommand  
method), 11

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.end\_command.EndCommand  
method), 12

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScanCommand  
method), 13

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsResetCommand  
method), 14

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResourcesCommand  
method), 15

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.reset\_command.ResetCommand  
method), 16

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.restart\_command.RestartCommand  
method), 18

is\_allowed\_low() (ska\_tmc\_subarraynode.commands.scan\_command.ScanCommand  
method), 19

is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.abstract\_command.SubarrayNodeCommand  
method), 7

is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.assign\_resources\_command.AssignResourcesCommand  
method), 9

is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.configure\_command.ConfigureCommand  
method), 11

is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.end\_command.EndCommand  
method), 12

is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.end\_scan\_command.EndScanCommand  
method), 13



method), 13  
 is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.obsreset\_command.ObsReset  
 method), 14  
 is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.release\_all\_resources\_command.ReleaseAllResources  
 method), 16  
 is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.reset\_command.Reset  
 method), 17  
 is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.restart\_command.Restart  
 method), 18  
 is\_allowed\_mid() (ska\_tmc\_subarraynode.commands.scan\_command.Scan  
 method), 19  
 is\_AssignResources\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 29  
 is\_Configure\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_End\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_EndScan\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_ObsReset\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_Off\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_On\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_ReleaseAllResources\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_Reset\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_Restart\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 30  
 is\_Scan\_allowed() (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 31  
 is\_scan\_timer\_running()  
 (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.monitoring\_loop,  
 method), 24  
 is\_Standby\_allowed()  
 (ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode  
 method), 31

**L**

last\_command\_executed  
 (ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.subarray\_node\_component\_manager  
 property), 24

**M**

main() (in module ska\_tmc\_subarraynode.subarray\_node\_low), 32  
 main() (in module ska\_tmc\_subarraynode.subarray\_node\_mid), 33  
 main() (in module ska\_tmc\_subarraynode.subarray\_node\_high), 34  
 mccs\_subarray\_dev\_name  
 (ska\_tmc\_subarraynode.subarray\_node\_low.ParameterLow  
 property), 27  
 module  
 ska\_tmc\_subarraynode, 35  
 ska\_tmc\_subarraynode.commands, 21  
 ska\_tmc\_subarraynode.commands.abstract\_command,  
 7  
 ska\_tmc\_subarraynode.commands.assign\_resources\_command,  
 8  
 ska\_tmc\_subarraynode.commands.configure\_command,  
 10  
 ska\_tmc\_subarraynode.commands.end\_command,  
 10  
 ska\_tmc\_subarraynode.commands.end\_scan\_command,  
 11  
 ska\_tmc\_subarraynode.commands.obsreset\_command,  
 14  
 ska\_tmc\_subarraynode.commands.off\_command,  
 20  
 ska\_tmc\_subarraynode.commands.on\_command,  
 20  
 ska\_tmc\_subarraynode.commands.release\_all\_resources\_command,  
 25  
 ska\_tmc\_subarraynode.commands.reset\_command,  
 25  
 ska\_tmc\_subarraynode.commands.restart\_command,  
 17  
 ska\_tmc\_subarraynode.commands.scan\_command,  
 18  
 ska\_tmc\_subarraynode.commands.standby\_command,  
 20  
 ska\_tmc\_subarraynode.exceptions, 34  
 ska\_tmc\_subarraynode.manager, 25  
 ska\_tmc\_subarraynode.manager.aggregators,  
 25  
 ska\_tmc\_subarraynode.manager.event\_receiver,  
 21  
 ska\_tmc\_subarraynode.manager.monitoring\_loop,  
 25  
 ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager,  
 25  
 ska\_tmc\_subarraynode.model, 29  
 ska\_tmc\_subarraynode.model.component, 25  
 ska\_tmc\_subarraynode.model.enum, 27  
 ska\_tmc\_subarraynode.model.input, 27  
 ska\_tmc\_subarraynode.release, 34  
 ska\_tmc\_subarraynode.subarray\_node, 29  
 ska\_tmc\_subarraynode.subarray\_node\_low,  
 32  
 ska\_tmc\_subarraynode.subarray\_node\_mid,  
 33

`ska_tmc_subarraynode.transaction_id`, 34

**N**

`NONE` (`ska_tmc_subarraynode.model.enum.PointingState` attribute), 27

**O**

`ObsReset` (class in `ska_tmc_subarraynode.commands.obsreset_command`), 14

`obsreset_csp()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsReset` method), 14

`obsreset_sdp()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsReset` method), 14

`obsreset_dish()` (`ska_tmc_subarraynode.commands.obsreset_command.ObsReset` method), 14

`ObsStateAggregatorLow` (class in `ska_tmc_subarraynode.manager.aggregators`), 21

`ObsStateAggregatorMid` (class in `ska_tmc_subarraynode.manager.aggregators`), 21

`Off` (class in `ska_tmc_subarraynode.commands.off_command`), 20

`Off()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 29

`On` (class in `ska_tmc_subarraynode.commands.on_command`), 20

`On()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 29

**P**

`PointingState` (class in `ska_tmc_subarraynode.model.enum`), 27

**R**

`read_commandExecuted()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_commandInProgress()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_cspSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_dishDevNames()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_internalModel()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_lastCommandExecuted()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_lastDeviceInfoChanged()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_mccsSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_low.SubarrayNodeLow` method), 32

`read_sbID()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_scanID()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_sdpSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_subarrayHealthState()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`read_tmcLeafCspSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_tmcLeafDishDevNames()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_tmcLeafMccsSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_low.SubarrayNodeLow` method), 32

`read_tmcLeafSdpSubarrayDevName()` (`ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid` method), 33

`read_transformedInternalModel()` (`ska_tmc_subarraynode.subarray_node.AbstractSubarrayNode` method), 31

`READY` (`ska_tmc_subarraynode.model.enum.PointingState` attribute), 27

`release_csp_resources()` (`ska_tmc_subarraynode.commands.release_all_resources_command` method), 16

`release_sdp_resources()` (`ska_tmc_subarraynode.commands.release_all_resources_command` method), 16

`ReleaseAllResources` (class in `ska_tmc_subarraynode.commands.release_all_resources_command`), 15

`remove_device()` (`ska_tmc_subarraynode.model.component.SubarrayComponent` method), 26

`remove_dish_devices()` (`ska_tmc_subarraynode.manager.subarray_node_component_manager` method), 24

`Reset` (class in `ska_tmc_subarraynode.commands.reset_command`), 16

`reset()` (`ska_tmc_subarraynode.manager.subarray_node_component_manager` method), 24

`Restart` (class in `ska_tmc_subarraynode.commands.restart_command`), 16

`restart_csp()` (`ska_tmc_subarraynode.commands.restart_command.RestartCsp` method), 16

`method`), 18  
`restart_dishes()` (`ska_tmc_subarraynode.commands.restart_command`, 7 Restart  
`method`), 18  
`restart_sdp()` (`ska_tmc_subarraynode.commands.restart_command`, 7 Restart  
`method`), 18  
**S**  
`sb_id` (`ska_tmc_subarraynode.model.component.SubarrayNodeComponent`  
`property`), 26  
`Scan` (class in `ska_tmc_subarraynode.commands.scan_command`, 12  
18  
`SCAN` (`ska_tmc_subarraynode.model.enum.PointingState`  
`attribute`), 27  
`scan_csp()` (`ska_tmc_subarraynode.commands.scan_command`, 12  
`method`), 19  
`scan_duration` (`ska_tmc_subarraynode.model.component.SubarrayNodeComponent`  
`property`), 26  
`scan_id` (`ska_tmc_subarraynode.model.component.SubarrayNodeComponent`  
`property`), 26  
`scan_mccs()` (`ska_tmc_subarraynode.commands.scan_command`, 12  
`method`), 19  
`scan_sdp()` (`ska_tmc_subarraynode.commands.scan_command`, 12  
`method`), 19  
`sdp_subarray_dev_name`  
(`ska_tmc_subarraynode.model.input.InputParameters`, 17  
`property`), 28  
`SdpSubarrayDeviceInfo` (class in `ska_tmc_subarraynode.model.component`),  
26  
`set_assigned_resources()`  
(`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_dish_dev_names()`  
(`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_obs_callbacks()`  
(`ska_tmc_subarraynode.model.component.SubarrayNodeComponent`  
`method`), 27  
`set_op_callbacks()` (`ska_tmc_subarraynode.model.component.SubarrayNodeComponent`  
`method`), 27  
`set_sb_id()` (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_scan_duration()`  
(`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_scan_id()` (`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_tmc_leaf_dish_dev_names()`  
(`ska_tmc_subarraynode.manager.subarray_node_component_manager.SubarrayNodeComponentManager`  
`method`), 24  
`set_up_dish_data()` (`ska_tmc_subarraynode.commands.assign_resources_command`, 13 Assign Resources  
`method`), 10  
`ska_tmc_subarraynode`  
module, 35

ska\_tmc\_subarraynode.subarray\_node\_mid  
     module, 33  
 ska\_tmc\_subarraynode.transaction\_id  
     module, 34  
 SLEW (*ska\_tmc\_subarraynode.model.enum.PointingState*  
     attribute), 27  
 Standby (class in *ska\_tmc\_subarraynode.commands.standby\_command*),  
     20  
 Standby() (*ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode*  
     method), 29  
 start\_scan\_timer() (*ska\_tmc\_subarraynode.commands.scan\_command.Scan*  
     method), 19  
 stop() (*ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
     method), 24  
 stop\_dish\_tracking()  
     (*ska\_tmc\_subarraynode.commands.end\_command.End*  
     method), 12  
 stop\_scan\_timer() (*ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
     method), 24  
 subarray\_health\_state  
     (*ska\_tmc\_subarraynode.model.component.SubarrayComponent*  
     property), 27  
 subarray\_node\_obstate\_not\_aggregated()  
     (*ska\_tmc\_subarraynode.manager.aggregators.ObstateAggregatorMid*  
     method), 21  
 SubarrayComponent (class in  
     *ska\_tmc\_subarraynode.model.component*),  
     26  
 SubarrayNodeCommand (class in  
     *ska\_tmc\_subarraynode.commands.abstract\_command*),  
     7  
 SubarrayNodeComponentManager (class in  
     *ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager*),  
     22  
 SubarrayNodeEventReceiver (class in  
     *ska\_tmc\_subarraynode.manager.event\_receiver*),  
     21  
 SubarrayNodeLow (class in  
     *ska\_tmc\_subarraynode.subarray\_node\_low*),  
     32  
 SubarrayNodeLow.InitCommand (class in  
     *ska\_tmc\_subarraynode.subarray\_node\_low*),  
     32  
 SubarrayNodeMid (class in  
     *ska\_tmc\_subarraynode.subarray\_node\_mid*),  
     33  
 SubarrayNodeMid.InitCommand (class in  
     *ska\_tmc\_subarraynode.subarray\_node\_mid*),  
     33  
 SubarrayNodeMonitoringLoop (class in  
     *ska\_tmc\_subarraynode.manager.monitoring\_loop*),  
     25  
 subscribe\_events() (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 21

## T

tm\_leaf\_csp\_subarray\_dev\_name  
     (*ska\_tmc\_subarraynode.model.input.InputParameterMid*  
     property), 28  
 tm\_leaf\_dish\_dev\_names  
     (*ska\_tmc\_subarraynode.model.input.InputParameterMid*  
     property), 28  
 tm\_leaf\_mccs\_subarray\_dev\_name  
     (*ska\_tmc\_subarraynode.model.input.InputParameterLow*  
     property), 28  
 tm\_leaf\_sup\_subarray\_dev\_name  
     (*ska\_tmc\_subarraynode.model.input.InputParameterMid*  
     property), 28  
 to\_dict() (*ska\_tmc\_subarraynode.model.component.DishDeviceInfo*  
     method), 25  
 to\_dict() (*ska\_tmc\_subarraynode.model.component.SdpSubarrayDevice*  
     method), 26  
 to\_dict() (*ska\_tmc\_subarraynode.model.component.SubarrayComponent*  
     method), 27  
 to\_json() (*ska\_tmc\_subarraynode.model.component.DishDeviceInfo*  
     method), 26  
 to\_json() (*ska\_tmc\_subarraynode.model.component.SdpSubarrayDevice*  
     method), 26  
 TRACK (*ska\_tmc\_subarraynode.model.enum.PointingState*  
     attribute), 27

## U

UNKNOWN (*ska\_tmc\_subarraynode.model.enum.PointingState*  
     attribute), 27  
 unsubscribe\_dish\_health()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 21  
 unsubscribe\_dish\_leaf\_health()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 21  
 unsubscribe\_dish\_leaf\_state()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 21  
 unsubscribe\_dish\_pointing()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 22  
 unsubscribe\_dish\_state()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 22  
 unsubscribe\_dish\_events()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 22  
 unsubscribe\_dish\_events()  
     (*ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
     method), 24  
 unsubscribe\_dish\_leaf\_events()  
     (*ska\_tmc\_subarraynode.manager.event\_receiver.SubarrayNodeEventReceiver*  
     method), 22

`update()` (*ska\_tmc\_subarraynode.model.input.InputParameter*  
*method*), 27

`update()` (*ska\_tmc\_subarraynode.model.input.InputParameterLow*  
*method*), 28

`update()` (*ska\_tmc\_subarraynode.model.input.InputParameterMid*  
*method*), 28

`update_assigned_resources_callback()`  
*(ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode*  
*method)*, 31

`update_assigned_resources_low()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 24

`update_command_in_progress_callback()`  
*(ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode*  
*method)*, 31

`update_device()` (*ska\_tmc\_subarraynode.model.component.SubarrayComponent*  
*method*), 27

`update_device_callback()`  
*(ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode*  
*method)*, 31

`update_device_exception()`  
*(ska\_tmc\_subarraynode.model.component.SubarrayComponent*  
*method)*, 27

`update_device_health_state()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 24

`update_device_info()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 24

`update_device_obs_state()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 24

`update_device_pointing_state()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 24

`update_device_state()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 25

`update_event_failure()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 25

`update_input_parameter()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 25

`update_mccs_json()` (*ska\_tmc\_subarraynode.commands.scan\_command.Scan*  
*method*), 19

`update_receive_addresses()`  
*(ska\_tmc\_subarraynode.manager.subarray\_node\_component\_manager.SubarrayNodeComponentManager*  
*method)*, 25

`update_subarray_health_state_callback()`  
*(ska\_tmc\_subarraynode.subarray\_node.AbstractSubarrayNode*  
*method)*, 31

`update_with_id()` (*in* *module*  
*ska\_tmc\_subarraynode.transaction\_id*), 34