

---

# **TMC Prototype Documentation**

***Release 1.0***

**NCRA India**

**Jan 05, 2021**



---

## Contents:

---

|           |                                |           |
|-----------|--------------------------------|-----------|
| <b>1</b>  | <b>Central Node</b>            | <b>1</b>  |
| <b>2</b>  | <b>Central Node</b>            | <b>5</b>  |
| <b>3</b>  | <b>Subarray Node</b>           | <b>11</b> |
| <b>4</b>  | <b>Subarray Node</b>           | <b>21</b> |
| <b>5</b>  | <b>Dish Leaf Node</b>          | <b>27</b> |
| <b>6</b>  | <b>Dish Master</b>             | <b>35</b> |
| <b>7</b>  | <b>CSP Master Leaf Node</b>    | <b>37</b> |
| <b>8</b>  | <b>SDP Subarray Leaf Node</b>  | <b>41</b> |
| <b>9</b>  | <b>Subarray Node</b>           | <b>43</b> |
| <b>10</b> | <b>CSP Subarray Leaf Node</b>  | <b>55</b> |
| <b>11</b> | <b>SDP Master Leaf Node</b>    | <b>67</b> |
| <b>12</b> | <b>MCCS Master Leaf Node</b>   | <b>73</b> |
| <b>13</b> | <b>MCCS Subarray Leaf Node</b> | <b>79</b> |
| <b>14</b> | <b>Indices and tables</b>      | <b>85</b> |
|           | <b>Python Module Index</b>     | <b>87</b> |
|           | <b>Index</b>                   | <b>89</b> |



# CHAPTER 1

---

## Central Node

---

Central Node is a coordinator of the complete M&C system. Central Node implements the standard set of state and mode attributes defined by the SKA Control Model.

**class** tmcprototype.centralnode.src.centralnode.central\_node.**CentralNode** (\*args,  
\*\*kwargs)

Central Node is a coordinator of the complete M&C system.

**AssignResources** (argin)

AssignResources command invokes the AssignResources command on lower level devices.

**CentralAlarmHandler**

Used by autodoc\_mock\_imports.

**CspMasterLeafNodeFQDN**

Used by autodoc\_mock\_imports.

**DishLeafNodePrefix**

Used by autodoc\_mock\_imports.

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

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

**do** ()

Initializes the attributes and properties of the Central Node.

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

**Return type** (ReturnCode, str)

**Raises** DevFailed if error occurs while initializing the CentralNode device or if error occurs while creating device proxy for any of the devices like SubarrayNode, DishLeafNode, CSPMasterLeafNode or SDPMasterLeafNode.

**NumDishes**

Used by autodoc\_mock\_imports.

**ReleaseResources** (argin)

Release all the resources assigned to the given Subarray.

**SdpMasterLeafNodeFQDN**

Used by autodoc\_mock\_imports.

**StandByTelescope ()**

This command invokes SetStandbyLPMode() command on DishLeafNode, StandBy() command on CspMasterLeafNode and SdpMasterLeafNode and Off() command on SubarrayNode and sets CentralNode into OFF state.

**StartUpTelescope ()**

This command invokes SetOperateMode() command on DishLeadNode, On() command on CspMasterLeafNode, SdpMasterLeafNode and SubarrayNode and sets the Central Node into ON state.

**StowAntennas (arg)**

This command stows the specified receptors.

**TMArmHandler**

Used by autodoc\_mock\_imports.

**TMMidSubarrayNodes**

Used by autodoc\_mock\_imports.

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook ()**

Internal construct of TANGO.

**delete\_device ()**

Internal construct of TANGO.

**init\_command\_objects ()**

Initialises the command handlers for commands supported by this device.

**is\_AssignResources\_allowed ()**

Checks whether this command is allowed to be run in current device 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\_ReleaseResources\_allowed ()**

Checks whether this command is allowed to be run in current device 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\_StandByTelescope\_allowed ()**

Checks whether this command is allowed to be run in current device 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\_StartUpTelescope\_allowed ()**

Checks whether this command is allowed to be run in current device 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\_StowAntennas\_allowed()**

Checks whether this command is allowed to be run in current device 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\_activityMessage()**

Internal construct of TANGO. Returns activity message.

**read\_subarray1HealthState()**

Internal construct of TANGO. Returns Subarray1 health state.

**read\_subarray2HealthState()**

Internal construct of TANGO. Returns Subarray2 health state.

**read\_subarray3HealthState()**

Internal construct of TANGO. Returns Subarray3 health state.

**read\_telescopeHealthState()**

Internal construct of TANGO. Returns the Telescope health state.

**subarray1HealthState**

Used by autodoc\_mock\_imports.

**subarray2HealthState**

Used by autodoc\_mock\_imports.

**subarray3HealthState**

Used by autodoc\_mock\_imports.

**telescopeHealthState**

Used by autodoc\_mock\_imports.

**write\_activityMessage(value)**

Internal construct of TANGO. Sets the activity message.

`tmcprototype.centralnode.src.centralnode.central_node.main(args=None, **kwargs)`

Runs the CentralNode. :param args: Arguments internal to TANGO

**Parameters** **kwargs** – Arguments internal to TANGO

**Returns** CentralNode TANGO object.

`tmcprototype.centralnode.src.centralnode.central_node.AssignResources`

Used by autodoc\_mock\_imports.

`tmcprototype.centralnode.src.centralnode.central_node.DeviceData`

Used by autodoc\_mock\_imports.

`tmcprototype.centralnode.src.centralnode.central_node.ObsStateAggregator`

Used by autodoc\_mock\_imports.

`tmcprototype.centralnode.src.centralnode.central_node.ReleaseResources`

Used by autodoc\_mock\_imports.

`tmcprototype.centralnode.src.centralnode.central_node.StandByTelescope`

Used by autodoc\_mock\_imports.

`tmcprototype.centralnode.src.centralnode.central_node.StartupTelescope`

Used by autodoc\_mock\_imports.

tmcprototype.centralnode.src.centralnode.central\_node.**StowAntennas**  
Used by autodoc\_mock\_imports.



## CHAPTER 2

---

### Central Node

---

Central Node is a coordinator of the complete M&C system. Central Node implements the standard set of state and mode attributes defined by the SKA Control Model.

```
class tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode (*args,  
                                                                           **kwargs)
```

Central Node is a coordinator of the complete M&C system.

**AssignResources** (*argin*)

AssignResources command invokes the AssignResources command on lower level devices.

```
class AssignResourcesCommand (*args, **kwargs)
```

A class for CentralNode's AssignResources() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do** (*argin*)

Assigns resources to given subarray. It accepts the subarray id, station ids, station beam id and channels in JSON string format.

**Parameters** **argin** – The string in JSON format. The JSON contains following values:

**subarray\_id**: DevShort. Mandatory. Sub-Array to allocate resources to

**station\_ids**: DevArray. Mandatory list of stations contributing beams to the data set

**channels**: DevArray. Mandatory list of frequency channels used

**station\_beam\_ids**: DevArray. Mandatory logical ID of beam

**Example:**

```
{ "subarray_id": 1, "station_ids": [1,2], "channels": [1,2,3,4,5,6,7,8], "station_beam_ids": [1]  
}
```

Note: From Jive, enter above input string without any space.

**Returns** None

**Raises** DevFailed if error occurs while invoking command on any of the devices like SubarrayNode, MCCSMasterLeafNode

Note: Enter input without spaces as: {"subarray\_id":1,"station\_ids":[1,2],"channels":[1,2,3,4,5,6,7,8],"station\_beam\_ids":1}

### **CentralAlarmHandler**

Used by autodoc\_mock\_imports.

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

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

**do** ()

Initializes the attributes and properties of the Central Node Low.

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

**Return type** (ReturnCode, str)

**Raises** DevFailed if error occurs while initializing the CentralNode device or if error occurs while creating device proxy for any of the devices like SubarrayNodeLow or MccsMasterLeafNode.

### **MCCSMasterLeafNodeFQDN**

Used by autodoc\_mock\_imports.

**ReleaseResources** (argIn)

Release all the resources assigned to the given Subarray.

**class ReleaseResourcesCommand** (\*args, \*\*kwargs)

A class for CentralNode's ReleaseResources() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do** (argIn)

Release all the resources assigned to the given Subarray. It accepts the subarray id, release\_all flag in JSON string format. When the release\_all flag is True, ReleaseAllResources command is invoked on the respective SubarrayNode.

**Parameters** **argIn** – The string in JSON format. The JSON contains following values:

**subarray\_id:** DevShort. Mandatory.

**release\_all:** Boolean(True or False). Mandatory. True when all the resources to be released from Subarray.

**Example:**

```
{ "subarray_id": 1, "release_all": true,
}
```

**Note: From Jive, enter input as:** {"subarray\_id":1,"release\_all":true} without any space.

**raises** ValueError if input argument json string contains invalid value KeyError if input argument json string contains invalid key DevFailed if the command execution or command invocation on SubarrayNode is not successful

### **StandByTelescope** ()

This command invokes Off() command on SubarrayNode, MCCSMasterLeafNode and sets CentralNode into OFF state.

**class StandByTelescopeCommand** (\*args, \*\*kwargs)

A class for Low CentralNode's StandByTelescope() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do ()**

Sets the CentralNodeLow into OFF state. Invokes the respective command on lower level nodes and devices.

param argin: None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while invoking command on any of the devices like SubarrayNode or MccsMasterLeafNode.

**StartupTelescope ()**

This command invokes On() command on SubarrayNode, MCCSMasterLeafNode and sets the Central Node into ON state.

**class StartupTelescopeCommand (\*args, \*\*kwargs)**

A class for Low CentralNode's StartupCommand() command.

**check\_allowed ()**

Checks whether this command is allowed to be run in current device 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

**do ()**

Setting the startup state to TRUE enables the telescope to accept subarray commands as per the subarray model. Set the CentralNode into ON state.

**Parameters** **argin** – None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while invoking command on any of the devices like SubarrayNode or MccsMasterLeafNode.

**TMArmHandler**

Used by autodoc\_mock\_imports.

**TMLowSubarrayNodes**

Used by autodoc\_mock\_imports.

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook ()**

Internal construct of TANGO.

**delete\_device ()**

Internal construct of TANGO.

**health\_state\_cb** (*evt*)

Receives the subscribed Subarray health state and MCCS Master Leaf Node health state, aggregates them to calculate the telescope health state.

**Parameters** **evt** – A event on Subarray healthState and MCCSMasterLeafNode healthstate.

**Type** Event object It has the following members:

- date (event timestamp)
- reception\_date (event reception timestamp)
- type (event type)
- dev\_name (device name)
- name (attribute name)
- value (event value)

**Returns** None

**Raises** KeyError if error occurs while setting telescope healthState

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**is\_AssignResources\_allowed** ()

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**is\_ReleaseResources\_allowed** ()

Checks whether this command is allowed to be run in current device 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\_StandByTelescope\_allowed** ()

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**is\_StartUpTelescope\_allowed** ()

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**read\_activityMessage** ()

Internal construct of TANGO. Returns activity message.

**read\_subarray1HealthState** ()

Internal construct of TANGO. Returns Subarray1 health state.

**read\_telescopeHealthState** ()

Internal construct of TANGO. Returns the Telescope health state.

**subarray1HealthState**

Used by autodoc\_mock\_imports.

**telescopeHealthState**

Used by autodoc\_mock\_imports.

**write\_activityMessage** (*value*)

Internal construct of TANGO. Sets the activity message.

`tmcprototype.centralnodelow.src.centralnodelow.central_node_low.main` (*args=None*,  
\*\**kwargs*)

Runs the CentralNode. :param args: Arguments internal to TANGO

**Parameters** **kwargs** – Arguments internal to TANGO

**Returns** CentralNode TANGO object.



---

### Subarray Node

---

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** tmcprototype.subarraynode.src.subarraynode.subarray\_node.**SubarrayNode** (\*args, \*\*kwargs)

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

**CspSubarrayFQDN**

Used by autodoc\_mock\_imports.

**CspSubarrayLNFQDN**

Used by autodoc\_mock\_imports.

**DishLeafNodePrefix**

Used by autodoc\_mock\_imports.

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

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.

**Return type** (ReturnCode, str)

**Raises** DevFailed if the error while subscribing the tango attribute

**SdpSubarrayFQDN**

Used by autodoc\_mock\_imports.

**SdpSubarrayLNFQDN**

Used by autodoc\_mock\_imports.

**Track** (argin)

Invokes Track command on the Dishes assigned to the Subarray.

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook()**

Internal construct of TANGO.

**delete\_device()**

Internal construct of TANGO.

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**is\_Track\_allowed()**

Checks whether this command is allowed to be run in current device 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\_activityMessage()**

Internal construct of TANGO. Returns activityMessage. Example: "Subarray node is initialized successfully" //result occurred after initialization of device.

**read\_receptorIDList()**

Internal construct of TANGO. Returns the receptor IDs allocated to the Subarray.

**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.

**receptorIDList**

Used by autodoc\_mock\_imports.

**sbID**

Used by autodoc\_mock\_imports.

**scanID**

Used by autodoc\_mock\_imports.

**write\_activityMessage(value)**

Internal construct of TANGO. Sets the activityMessage.

```
tmcprototype.subarraynode.src.subarraynode.subarray_node.main(args=None,  
                                                                **kwargs)
```

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

```
tmcprototype.subarraynode.src.subarraynode.subarray_node.AssignResources
```

Used by autodoc\_mock\_imports.

```
tmcprototype.subarraynode.src.subarraynode.subarray_node.ReleaseAllResources
```

Used by autodoc\_mock\_imports.

```
tmcprototype.subarraynode.src.subarraynode.subarray_node.Configure
```

Used by autodoc\_mock\_imports.

```
tmcprototype.subarraynode.src.subarraynode.subarray_node.Scan
```

Used by autodoc\_mock\_imports.



`tmcprototype.subarraynode.src.subarraynode.subarray_node.EndScan`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.End`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.On`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.Off`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.Track`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.Abort`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.Restart`  
Used by autodoc\_mock\_imports.

`tmcprototype.subarraynode.src.subarraynode.subarray_node.ObsReset`  
Used by autodoc\_mock\_imports.

On Command class for SubarrayNode

```
class tmcprototype.subarraynode.src.subarraynode.on_command.On (*args,
                                                                **kwargs)
```

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

**do()**

This command invokes On Command on CSPSubarray and SDPSubarray through respective leaf nodes.  
This comamnd changes Subaray device state from OFF to ON.

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

**Return type** (ResultCode, str)

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

**set\_csp\_client** (*device\_data*)  
set up csp devices

**set\_sdp\_client** (*device\_data*)  
set up sdp devices

**turn\_on\_leaf\_node** (*tango\_client*)

Off Command class for SubarrayNode

```
class tmcprototype.subarraynode.src.subarraynode.off_command.Off (*args,
                                                                **kwargs)
```

A class for the SubarrayNodes's Off() command.

**do()**

This command invokes Off Command on CSPSubarray and SDPSubarray through respective leaf nodes.  
This comamnd changes Subaray device state from ON to OFF.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

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

AssignResourcesCommand class for SubarrayNode.

```
class tmcprototype.subarraynode.src.subarraynode.assign_resources_command.AssignResources (
```

A class for SubarrayNode's AssignResources() command.

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

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*)

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.

**Parameters** *argin* – DevString.

Example:

```
{ "dish": { "receptorIDList": ["0002", "0001"], "sdp": { "id": "sbi-mvp01-20200325-00001", "max_length": 100.0, "scan_types": [ { "id": "science_A", "coordinate_system": "ICRS", "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]] } ], { "id": "calibration_B", "coordinate_system": "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]] }, { "start": 2000, "stride": 1, "freq_min": 0.36e9, "freq_max": 0.368e9, "link_map": [[2000,4],[2200,5]] } ] }, "processing_blocks": [ { "id": "pb-mvp01-20200325-00001", "workflow": { "type": "realtime", "id": "vis_receive", "version": "0.1.0", "parameters": {} }, { "id": "pb-mvp01-20200325-00002", "workflow": { "type": "realtime", "id": "test_realtime",
```

```

“version”:”0.1.0”),”parameters”:{}},{“id”:”pb-mvp01-20200325-00003”,”workflow”:
{“type”:”batch”,”id”:”ical”,”version”:”0.1.0”),”parameters”:{},”dependencies”:[
{“pb_id”:”pb-mvp01-20200325-00001”,”type”:[“visibilities”]}]},{“id”:
20200325-00004”,”workflow”:{“type”:”batch”,”id”:”dpreb”,”version”:”0.1.0”),
“parameters”:{},”dependencies”:[{“pb_id”:”pb-mvp01-20200325-00003”,”type”: [“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.

**Return type** (ResultCode, str)

**Raises** ValueError if input argument json string contains invalid value DevFailed if the command execution is not successful

**set\_up\_dish\_data** (*argin*)

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** *argin* – DevVarStringArray. List of receptor IDs to be allocated to subarray.  
Example: [‘0001’, ‘0002’]

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

ReleaseAllResources Command for SubarrayNode

**class** tmcprototype.subarraynode.src.subarraynode.release\_all\_resources\_command.**ReleaseAllResources**

A class for SKASubarray’s ReleaseAllResources() command.

**do** ()

It checks whether all resources are already released. If yes then it throws error while executing command. If not it Releases all the resources from the subarray i.e. Releases resources from TMC Subarray Node, CSP Subarray and SDP Subarray. If the command execution fails, array of receptors(device names) which are failed to be released from the subarray, is returned to Central Node. Upon successful execution, all the resources of a given subarray get released and empty array is returned. Selective release is not yet supported.

**Returns** A tuple containing a return code and “[ ]” as a string on successful release all resources.

Example: “[ ]” as string on successful release all resources.

**Return type** (ResultCode, str)

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

**release\_csp\_resources** (*csp\_subarray\_in\_fqdn*)

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

**Parameters** *argin* – DevVoid

**Returns** DevVoid

**release\_sdp\_resources** (*sdp\_subarray\_in\_fqdn*)

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

**Parameters** `argin` – DevVoid

**Returns** DevVoid

Configure Command class for SubarrayNode.

```
class tmcprototype.subarraynode.src.subarraynode.configure_command.Configure (*args,  
                                                                    **kwargs)
```

A class for SubarrayNode's Configure() command.

```
check_only_dish_config (scan_configuration)
```

```
do (argin)
```

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).

**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: {"pointing":{"target":{"system":"ICRS","name":"Polaris Australis","RA":"21:08:47.92","dec":"-88:57:22.9"}}, "dish":{"receiverBand":"1"},"csp":{"id":"sbi-mvp01-20200325-00001-science\_A","frequencyBand":"1","fsp":[{"fspID":1,"functionMode":"CORR","frequencySliceID":1,"integrationTime":1400,"corrBandwidth":0,"channelAveragingMap":[[0, 2], [744, 0]], "fspChannelOffset": 0, "outputLinkMap": [[0, 0], [200, 1]], "outputHost": [[0, '192.168.0.1'], [400, '192.168.0.2']], "outputMac": [[0, '06-00-00-00-00-00']], "outputPort": [[0, 9000, 1], [400, 9000, 1]]}, {"fspID":2,"functionMode":"CORR","frequencySliceID":2,"integrationTime":1400,"corrBandwidth":0,"channelAveragingMap":[[0, 2], [744, 0]], "fspChannelOffset": 744, "outputLinkMap": [[0, 4], [200, 5]], "outputHost": [[0, '192.168.0.3'], [400, '192.168.0.4']], "outputMac": [[0, '06-00-00-00-00-01']], "outputPort": [[0, 9000, 1], [400, 9000, 1]]}}}, "sdp":{"scan\_type":"science\_A"},"tmc":{"scanDuration":10.0}} CSP block in json string is as per earlier implementation and not aligned to SP-872 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.

**Return type** (ReturnCode, str)

**Raises** JSONDecodeError if input argument json string contains invalid value

```
class tmcprototype.subarraynode.src.subarraynode.configure_command.ElementDeviceData
```

```
static build_up_csp_cmd_data (scan_config, attr_name_map, receive_addresses_map)
```

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. { 'id': 'sbi-mvp01-20200325-00001-science\_A', 'frequencyBand': '1', 'fsp': [{ 'fspID': 1, 'functionMode': 'CORR', 'frequencySliceID': 1, 'integrationTime': 1400, 'corrBandwidth': 0, 'channelAveragingMap': [[0, 2], [744, 0]], 'fspChannelOffset': 0, 'outputLinkMap': [[0, 0], [200, 1]], 'outputHost': [[0, '192.168.0.1'], [400, '192.168.0.2']], 'outputMac': [[0, '06-00-00-00-00-00']], 'outputPort': [[0, 9000, 1], [400, 9000, 1]]}, { 'fspID': 2, 'functionMode': 'CORR', 'frequencySliceID': 2, 'integrationTime': 1400, 'corrBandwidth': 0, 'channelAveragingMap': [[0, 2], [744, 0]], 'fspChannelOffset': 744, 'outputLinkMap': [[0, 4], [200, 5]], 'outputHost': [[0, '192.168.0.3'], [400, '192.168.0.4']], 'outputMac': [[0, '06-00-00-00-00-01']], 'outputPort': [[0, 9000, 1], [400, 9000, 1]]}] }

**Returns** csp configuration schema

```
static build_up_dsh_cmd_data (scan_config)
```

```
static build_up_sdp_cmd_data (scan_config)
```

Scan Command class for SubarrayNode

```
class tmcprototype.subarraynode.src.subarraynode.scan_command.Scan (*args,  
                                                                    **kwargs)
```

A class for SubarrayNode's Scan() command.

**do** (*argin*)

This command accepts id as input. And it Schedule scan on subarray from where 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.

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

JSON string example as follows:

```
{“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.

**Return type** (ReturnCode, str)

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

**scan\_csp** (*device\_data*, *argin*)

set up csp devices

**scan\_sdp** (*device\_data*, *argin*)

set up sdp devices

**class** tmcprototype.subarraynode.src.subarraynode.scan\_command.**ScanThread**

A class to start the Scan threading.

**call\_end\_scan\_command** (*device\_data*)

**scan\_thread** (*device\_data*)

EndScan Command class for SubarrayNode.

**class** tmcprototype.subarraynode.src.subarraynode.end\_scan\_command.**EndScan** (*\*args*,  
*\*\*kwargs*)

A class for SubarrayNode’s EndScan() command.

**do** ()

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.

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

The message is for information purpose only.

**Return type** (ReturnCode, str)

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

**endscan\_csp** (*device\_data*)

set up csp devices

**endscan\_sdp** (*device\_data*)

set up sdp devices

End Command class for SubarrayNode.

**class** tmcprototype.subarraynode.src.subarraynode.end\_command.**End** (*\*args*,  
*\*\*kwargs*)

A class for SubarrayNode’s End() command.

**do ()**

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

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

The message is for information purpose only.

**Return type** (ResultCode, str)

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

**end\_csp** (*device\_data*)

set up csp devices

**end\_sdp** (*device\_data*)

set up sdp devices

**stop\_dish\_tracking** (*device\_data*)

Abort Command for SubarrayNode.

```
class tmcprototype.subarraynode.src.subarraynode.abort_command.Abort (*args,  
                                                                    **kwargs)
```

A class for SubarrayNode's Abort() command.

**abort\_csp** (*device\_data*)

set up csp devices

**abort\_dishes** (*device\_data*)

**abort\_sdp** (*device\_data*)

set up sdp devices

**do ()**

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

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

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs in invoking command on any of the devices like CSPSubarrayLeafNode, SDPSubarrayLeafNode or DishLeafNode

Restart Command for SubarrayNode.

```
class tmcprototype.subarraynode.src.subarraynode.restart_command.Restart (*args,  
                                                                    **kwargs)
```

A class for SubarrayNode's Restart() command.

**do ()**

This command invokes Restart command on CSPSubarrayLeafNode, SDPSubarrayLeafNode and DishLeafNode.

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

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while invoking command on CSPSubarrayLeafNode, SDPSubarrayLeafNode or DishLeafNode.

**restart\_dsh\_grp** (*device\_data*)

**restart\_leaf\_nodes** (*leaf\_node\_fqdn, info\_string*)  
set up sdp devices

ObsReset Command for SubarrayNode.

**class** tmcprototype.subarraynode.src.subarraynode.obsreset\_command.**ObsReset** (*\*args, \*\*kwargs*)

A class for SubarrayNode's ObsReset() command.

**do** ()

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

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

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while invoking command on CspSubarrayLeafNode, SdpSubarrayLeafNode or DishLeafNode.

**obsreset\_csp** (*device\_data*)  
set up csp devices

**obsreset\_dsh\_grp** (*device\_data*)

**obsreset\_sdp** (*device\_data*)  
set up sdp devices

Track Command class for SubarrayNode.

**class** tmcprototype.subarraynode.src.subarraynode.track\_command.**Track** (*\*args, \*\*kwargs*)

A class for SubarrayNode's Track command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do** (*argin*)

Invokes Track command on the Dishes assigned to the Subarray.

**Parameters** *argin* – DevString

Example: radec121:08:47.921-88:57:22.9 as *argin* *Argin* to be provided is the Ra and Dec values where first value is tag that is radec, second value is Ra in Hr:Min:Sec, and third value is Dec in Deg:Min:Sec.

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

The message is for information purpose only.

**Return type** (ResultCode, str)





## CHAPTER 4

---

### Subarray Node

---

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.

```
class tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode (*args,  
                                                                                       **kwargs)
```

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

```
class InitCommand (*args, **kwargs)
```

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.

**Return type** (ReturnCode, str)

**Raises** DevFailed if the error while subscribing the tango attribute

**MccsSubarrayFQDN**

Used by autodoc\_mock\_imports.

**MccsSubarrayLNFQDN**

Used by autodoc\_mock\_imports.

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook** ()

Internal construct of TANGO.

**calculate\_observation\_state** ()

Calculates aggregated observation state of Subarray.

**command\_class\_object** ()

Sets up the command objects :return: None

**delete\_device** ()

Internal construct of TANGO.

**get\_deviceproxy** (*device\_fqdn*)

Returns device proxy for given FQDN.

**health\_state\_cb** (*event*)

Receives the subscribed health states, aggregates them to calculate the overall subarray health state.

**Parameters** **evt** – A event on MCCS Subarray healthState.

**Type** Event object It has the following members:

- date (event timestamp)
- reception\_date (event reception timestamp)
- type (event type)
- dev\_name (device name)
- name (attribute name)
- value (event value)

**Returns** None

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**observation\_state\_cb** (*evt*)

Receives the subscribed MCCS Subarray obsState.

**Parameters** **evt** – A event on MCCS Subarray ObsState.

**Type** Event object It has the following members:

- date (event timestamp)
- reception\_date (event reception timestamp)
- type (event type)
- dev\_name (device name)
- name (attribute name)
- value (event value)

**Returns** None

**Raises** KeyError if error occurs while setting SubarrayNode's ObsState.

**read\_activityMessage** ()

Internal construct of TANGO. Returns activityMessage. Example: "Subarray node is initialized successfully" //result occurred after initialization of device.

**read\_scanID** ()

Internal construct of TANGO. Returns the Scan ID.

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

**scanID**

Used by autodoc\_mock\_imports.

**write\_activityMessage** (*value*)

Internal construct of TANGO. Sets the activityMessage.

```
tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.main (args=None,  

**kwargs)
```

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

AssignResourcesCommand class for SubarrayNodeLow.

```
class tmcprototype.subarraynodelow.src.subarraynodelow.assign_resources_command.AssignResou
```

A class for SubarrayNodeLow's AssignResources() command.

```
do (argin)
```

Assigns the resources to the subarray. It accepts station ids, channels, station beam ids

**Parameters** **argin** – DevString in JSON form containing following fields: station\_ids: list of integers

channels: list of integers

station\_beam\_ids: list of integers

Example:

```
{“station_ids”: [1, 2], “channels”: [1, 2, 3, 4, 5, 6, 7, 8], “station_beam_ids”: [1]}
```

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

ConfigureCommand class for SubarrayNodeLow.

```
class tmcprototype.subarraynodelow.src.subarraynodelow.configure_command.ConfigureCommand (
```

A class for SubarrayNodeLow's Configure() command.

```
do (argin)
```

Configures the resources assigned to the Mccs Subarray.

**Parameters** **argin** – DevString.

JSON string example is:

```
{“mccs”: {“stations”: [{“station_id”:1,}, {“station_id”:2,}], “station_beam_pointings”: [{“sta-  

tion_beam_id”:1, “target”: {“system”: “HORIZON”, “name”: “DriftScan”, “Az”:180.0, “El”:45.0},  

“update_rate”:0.0, “channels”: [1,2,3,4,5,6,7,8]}], “tmc”: {“scanDuration”:10.0} }
```

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

**Return type** (ReturnCode, str)

**Raises** JSONDecodeError if input argument json string contains invalid value DevFailed if the command execution is not successful.

EndCommand class for SubarrayNodeLow.

```
class tmcprototype.subarraynodelow.src.subarraynodelow.end_command.EndCommand (*args,  

**kwargs)
```

A class for SubarrayNodeLow's End() command.

```
do ()
```

This command on Subarray Node Low invokes End command on MCCA Subarray Leaf Node.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

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

EndScanCommand class for SubarrayNodeLow.

```
class tmcprototype.subarraynodelow.src.subarraynodelow.end_scan_command.EndScanCommand (*args,  
                                                                                       **kwargs)
```

A class for SubarrayNodeLow's EndScan() command.

**do ()**

Ends the scan. It is invoked on subarrayLow 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.

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

The message is for information purpose only.

**Return type** (ReturnCode, str)

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

OffCommand class for SubarrayNodeLow

```
class tmcprototype.subarraynodelow.src.subarraynodelow.off_command.OffCommand (*args,  
                                                                                       **kwargs)
```

A class for the SubarrayNodes's Off() command.

**do ()**

This command invokes Off Command on MCCSSubarray through mccs subarray leaf node. This command changes Subarray device state from ON to OFF.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

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

OnCommand class for SubarrayNodeLow

```
class tmcprototype.subarraynodelow.src.subarraynodelow.on_command.OnCommand (*args,  
                                                                                       **kwargs)
```

A class for the SubarrayNodeLow's On() command.

**do ()**

This command invokes On Command on MCCSSubarray through MCCS Subarray Leaf node. This command changes Subarray device state from OFF to ON.

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

**Return type** (ResultCode, str)

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

ReleaseAllResourcesCommand for SubarrayNodeLow

```
class tmcprototype.subarraynodelow.src.subarraynodelow.release_all_resources_command.ReleaseAllResourcesCommand
```

A class for SKASubarrayLow's ReleaseAllResources() command.

**do ()**

It invokes ReleaseAllResources command on SubarrayLow.

**Returns** A tuple containing a return code and RELEASEALLRESOURCES command invoked successfully as a string on successful release all resources.

Example: RELEASEALLRESOURCES command invoked successfully as string on successful release all resources.

**Return type** (ResultCode, str)

ScanCommand class for SubarrayNodeLow

```
class tmcprototype.subarraynodelow.src.subarraynodelow.scan_command.ScanCommand(*args,  
**kwargs)
```

A class for SubarrayNodeLow's Scan() command.

**call\_end\_scan\_command**()

**do**(*argin*)

This command accepts id as input. And it Schedule scan on subarray from where scan command is invoked on MCCS subarray Leaf 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.

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

JSON string example as follows:

```
{“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.

**Return type** (ReturnCode, str)

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



## CHAPTER 5

---

### Dish Leaf Node

---

A Leaf control node for DishMaster.

```
class tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode (*args,  
                                                                           **kwargs)
```

A Leaf control node for DishMaster.

**Abort** ()

Invokes Abort command on the DishMaster.

```
class AbortCommand (*args, **kwargs)
```

A class for DishLeafNode's Abort command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**do** ()

Invokes TrackStop command on the DishMaster.

**Raises DevFailed** – If error occurs while invoking TrackStop command on Dish-Master.

**Configure** (argin)

Configures the Dish by setting pointing coordinates for a given observation.

```
class ConfigureCommand (*args, **kwargs)
```

A class for DishLeafNode's Configure() command.

**check\_allowed** ()

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

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

**Return type** boolean

**do** (*argin*)

Configures the Dish by setting pointing coordinates for a given scan. This function accepts the input json and calculate pointing parameters of Dish- Azimuth and Elevation Angle. Calculated parameters are again converted to json and fed to the dish master.

**Parameters** **argin** – A String in a JSON format that includes pointing parameters of Dish- Azimuth and Elevation Angle.

Example:        {“pointing”:{“target”:{“system”:”ICRS”,“name”:”Polaris  
Australis”,“RA”:”21:08:47.92”,“dec”:”-88:57:22.9”}},  
“dish”:{“receiverBand”:”1”}}

**Raises** **DevFailed** – If error occurs while invoking ConfigureBand<> command on DishMaster or if the json string contains invalid data.

**DishMasterFQDN**

Used by autodoc\_mock\_imports.

**EndScan** (*argin*)

Invokes StopCapture command on DishMaster.

**class** **EndScanCommand** (*\*args, \*\*kwargs*)

A class for DishLeafNode’s EndScan() command.

**check\_allowed** ()

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

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

**Return type** boolean

**do** (*argin*)

Invokes StopCapture command on DishMaster.

**Parameters** **argin** – timestamp

**Raises** **DevFailed** – If error occurs while invoking StopCapture command on DishMaster.

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

A class for the TMC DishLeafNode’s init\_device() method.

**do** ()

Initializes the attributes and properties of the DishLeafNode.

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

**Return type** (ResultCode, str)

**Raises** **DevFailed** – If error occurs in creating a DeviceProxy instance for DishMaster

**ObsReset** ()

Invokes ObsReset command on the DishLeafNode.

**class** **ObsResetCommand** (*\*args, \*\*kwargs*)

A class for DishLeafNode’s ObsReset command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean



**do ()**  
Invokes SetStandbyFPMODE command on the DishMaster.

**Raises DevFailed** – If error occurs while invoking SetStandbyFPMODE command on DishMaster.

**Restart ()**  
Invokes Restart command on the DishMaster.

**class RestartCommand (\*args, \*\*kwargs)**  
A class for DishLeafNode's Restart command.

**check\_allowed ()**  
Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**do ()**  
Invokes StopCapture command on the DishMaster.

**Raises DevFailed** – If error occurs while invoking StopCapture command on DishMaster.

**Scan (argin)**  
Invokes Scan command on DishMaster.

**class ScanCommand (\*args, \*\*kwargs)**  
A class for DishLeafNode's Scan() command.

**check\_allowed ()**  
Checks whether this command is allowed to be run in the current device state.

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

**Return type** boolean

**do (argin)**  
Invokes Scan command on DishMaster.

**Parameters argin** – timestamp

**Raises DevFailed** – If error occurs while invoking Scan command on DishMaster.

**SetOperateMode ()**  
Invokes SetOperateMode command on DishMaster.

**class SetOperateModeCommand (\*args, \*\*kwargs)**  
A class for DishLeafNode's SetOperateMode() command.

**do ()**  
Invokes SetOperateMode command on DishMaster.

**Raises DevFailed** – If error occurs while invoking SetOperateMode command on DishMaster.

**SetStandbyFPMODE ()**  
Invokes SetStandbyFPMODE command on DishMaster (Standby-Full power) mode.

**class SetStandbyFPMODECommand (\*args, \*\*kwargs)**  
A class for DishLeafNode's SetStandbyFPMODE() command.

**do ()**  
Invokes SetStandbyFPMODE command on DishMaster (Standby-Full power) mode.

**Raises DevFailed** – If error occurs while invoking SetStandbyFPMODE command on DishMaster.

**SetStandbyLPMode ()**

Invokes SetStandbyLPMode (i.e. Low Power State) command on DishMaster.

**class SetStandbyLPModeCommand (\*args, \*\*kwargs)**

A class for DishLeafNode's SetStandbyLPMode() command.

**do ()**

Invokes SetStandbyLPMode (i.e. Low Power State) command on DishMaster.

**Raises DevFailed** – If error occurs while invoking SetStandbyLPMode command on DishMaster.

**SetStowMode ()**

Invokes SetStowMode command on DishMaster.

**class SetStowModeCommand (\*args, \*\*kwargs)**

A class for DishLeafNode's SetStowMode() command.

**do ()**

Invokes SetStowMode command on DishMaster.

**Raises DevFailed** – If error occurs while invoking SetStowMode command on DishMaster.

**Slew (argIn)**

Invokes Slew command on DishMaster to slew the dish towards the set pointing coordinates.

**class SlewCommand (\*args, \*\*kwargs)**

A class for DishLeafNode's SlewCommand() command.

**check\_allowed ()**

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

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

**Return type** boolean

**do (argIn)**

Invokes Slew command on DishMaster to slew the dish towards the set pointing coordinates.

**Parameters argIn** – list [0] = Azimuth, in degrees [1] = Elevation, in degrees

**Raises DevFailed** – If error occurs while invoking Slew command on DishMaster.

**StartCapture (argIn)**

Triggers the DishMaster to Start capture on the set configured band.

**class StartCaptureCommand (\*args, \*\*kwargs)**

A class for DishLeafNode's StartCapture() command.

**check\_allowed ()**

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

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

**Return type** boolean

**do (argIn)**

Invokes StartCapture command on DishMaster on the set configured band.

**Parameters argIn** – timestamp

**Raises DevFailed** – If error occurs while invoking StartCapture command on DishMaster.

**StopCapture** (*argin*)

Invokes StopCapture command on DishMaster on the set configured band.

**class StopCaptureCommand** (*\*args, \*\*kwargs*)

A class for DishLeafNode's StopCapture() command.

**check\_allowed** ()

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

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

**Return type** boolean

**do** (*argin*)

Invokes StopCapture command on DishMaster on the set configured band.

**Parameters** **argin** – timestamp

**Raises DevFailed** – If error occurs while invoking StopCapture command on DishMaster.

**StopTrack** ()

Invokes StopTrack command on the DishMaster.

**class StopTrackCommand** (*\*args, \*\*kwargs*)

A class for DishLeafNode's StopTrack() command.

**check\_allowed** ()

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

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

**Return type** boolean

**do** ()

Invokes TrackStop command on the DishMaster.

**Raises DevFailed** – If error occurs while invoking TrackStop command on DishMaster.

**Track** (*argin*)

Invokes Track command on the DishMaster.

**class TrackCommand** (*\*args, \*\*kwargs*)

A class for DishLeafNode's Track() command.

**check\_allowed** ()

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

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

**Return type** boolean

**do** (*argin*)

Invokes Track command on the DishMaster.

**Parameters** **argin** – DevString The elevation limit thread allows Dish to track a source till the observation capacity i.e. elevation limit of dish.

The tracking time thread allows dish to track a source for the prespecified Track Duration (provided elevation limit is not reached).

For Track command, argin to be provided is the Ra and Dec values in the following JSON format:

```
{“pointing”:{“target”:{“system”:”ICRS”,”name”:”Polaris
Australis”,”RA”:”21:08:47.92”,”dec”:”-88:57:22.9”}},
“dish”:{“receiverBand”:”1”}}
```

**Raises DevFailed** – If error occurs while invoking Track command on DishMaster.

**activityMessage**

Used by autodoc\_mock\_imports.

**attribute\_event\_handler** (*event\_data*)

Retrieves the subscribed attribute of DishMaster.

**Parameters** **evt** – A TANGO\_CHANGE event on attribute.

**convert\_radec\_to\_azel** (*target, timestamp*)

Converts RaDec coordinate in to AzEl coordinate using KATPoint library.

**Parameters**

- **target** – str Argin to be provided is the Ra and Dec values in the following format: radec,21:08:47.92,89:15:51.4
- **timestamp** – str 2020-12-11 10:06:34.970731

**Returns** list Azimuth and elevation angle, in degrees

**Raises ValueError** – If error occurs when creating katpoint Target or Timestamp.

**dishHealthState**

Used by autodoc\_mock\_imports.

**dishPointingState**

Used by autodoc\_mock\_imports.

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**is\_Abort\_allowed** ()

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**is\_Configure\_allowed** ()

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

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

**Return type** boolean

**is\_EndScan\_allowed** ()

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

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

**Return type** boolean

**is\_ObsReset\_allowed** ()

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**is\_Restart\_allowed()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**is\_Scan\_allowed()**

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

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

**Return type** boolean

**is\_Slew\_allowed()**

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

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

**Return type** boolean

**is\_StartCapture\_allowed()**

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

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

**Return type** boolean

**is\_StopCapture\_allowed()**

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

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

**Return type** boolean

**is\_StopTrack\_allowed()**

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

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

**Return type** boolean

**is\_Track\_allowed()**

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

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

**Return type** boolean

**read\_activityMessage()**

Returns the activityMessage

**set\_dish\_name\_number()**

**set\_observer\_lat\_long\_alt()**

**track\_thread()**

This thread writes coordinates to desiredPointing on DishMaster at the rate of 20 Hz.

**write\_activityMessage(value)**

Internal construct of TANGO. Sets the activityMessage

`tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.main(args=None,  
**kwargs)`

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



## CHAPTER 6

---

Dish Master

---





---

## CSP Master Leaf Node

---

CSP Master Leaf node monitors the CSP Master and issues control actions during an observation.

**class** tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.**CspMasterLeafNode**

**Properties:**

- CspMasterFQDN - Property to provide FQDN of CSP Master Device

**Attributes:**

- cspHealthState - Forwarded attribute to provide CSP Master Health State
- activityMessage - Attribute to provide activity message

**CspMasterFQDN**

Used by autodoc\_mock\_imports.

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

A class for the TMC CSP Master Leaf Node's init\_device() method.

**do** ()

Initializes the attributes and properties of the CspMasterLeafNode.

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

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while creating the device proxy for CSP Master or subscribing the events.

**class** **OffCommand** (\*args, \*\*kwargs)

A class for CspMasterLeafNode's Off() command.

**do** ()

Invokes Off command on the CSP Element.

**Parameters** **argin** – None.

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

**Return type** (ResultCode, str)

**off\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the Off command has been successfully invoked on CSPMaster.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**class OnCommand** (*\*args, \*\*kwargs*)

A class for CspMasterLeafNode's On() command.

**do** ()

Invokes On command on the CSP Element.

**Parameters** **argin** – None

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

**Return type** (ResultCode, str)

**Raises** DevFailed on communication failure with CspMaster or CspMaster is in error state.

**on\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the On command has been successfully invoked on CSPMaster.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**Standby** (*argin*)

Sets Standby Mode on the CSP Element.

**class StandbyCommand** (*\*args, \*\*kwargs*)

A class for CspMasterLeafNode's Standby() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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.

**do** (*argin*)

It invokes the STANDBY command on CSP Master.

**Parameters** *argin* – DevStringArray.

If the array length is 0, the command applies to the whole CSP Element. If the array length is > 1, each array element specifies the FQDN of the CSP SubElement to put in STANDBY mode.

**Returns** None

**Raises** DevFailed on communication failure with CspMaster or CspMaster is in error state.

**standby\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the StandBy command has been successfully invoked on CSPMaster.

**Parameters** *event* – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- *device* : (DeviceProxy) The DeviceProxy object on which the call was executed.
- *cmd\_name* : (str) The command name
- *argout\_raw* : (DeviceData) The command argout
- *argout* : The command argout
- *err* : (bool) A boolean flag set to true if the command failed. False otherwise
- *errors* : (sequence<DevError>) The error stack
- *ext*

**Returns** none

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook** ()

Internal construct of TANGO.

**cspHealthState**

Used by autodoc\_mock\_imports.

**csp\_cbf\_health\_state\_cb** (*evt*)

Retrieves the subscribed cspCbfHealthState attribute of CSPMaster.

**Parameters** *evt* – A TANGO\_CHANGE event on cspCbfHealthState attribute.

**Returns** None

**csp\_pss\_health\_state\_cb** (*evt*)

Retrieves the subscribed cspPssHealthState attribute of CSPMaster.

**Parameters** *evt* – A TANGO\_CHANGE event on cspPssHealthState attribute.

**Returns** None

**csp\_pst\_health\_state\_cb** (*evt*)

Retrieves the subscribed cspPstHealthState attribute of CSPMaster.

**Parameters** *evt* – A TANGO\_CHANGE event on cspPstHealthState attribute.

**Returns** None

**delete\_device** ()

Internal construct of TANGO.

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**is\_Standby\_allowed** ()

Checks whether this command is allowed to be run in current device 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\_activityMessage** ()

Internal construct of TANGO. Returns the activityMessage.

**write\_activityMessage** (*value*)

Internal construct of TANGO. Sets the activityMessage.

tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.**main** (*args=None*,  
\*\**kwargs*)

Runs the CspMasterLeafNode.

**Parameters**

- **args** – Arguments internal to TANGO
- **kwargs** – Arguments internal to TANGO

**Returns** CspMasterLeafNode TANGO object.

## CHAPTER 8

---

### SDP Subarray Leaf Node

---



## Subarray Node

SDP Subarray Leaf node is to monitor the SDP Subarray and issue control actions during an observation. It also acts as a SDP contact point for Subarray Node for observation execution.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray\_leaf\_node.**SdpS**

SDP Subarray Leaf node is to monitor the SDP Subarray and issue control actions during an observation.

**Abort ()**

Invoke Abort on SdpSubarrayLeafNode.

**AssignResources (argin)**

Assigns resources to given SDP subarray.

**Configure (argin)**

Invokes Configure on SdpSubarrayLeafNode.

**End ()**

This command invokes End command on SDP subarray to end the current Scheduling block.

**EndScan ()**

Invokes EndScan on SdpSubarrayLeafNode.

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

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

**do ()**

Initializes the attributes and properties of the SdpSubarrayLeafNode.

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

**Return type** (ResultCode, str)

**ObsReset ()**

Invoke ObsReset command on SdpSubarrayLeafNode.

**ReleaseAllResources ()**

Invokes ReleaseAllResources command on SdpSubarrayLeafNode.

**Restart ()**

Invoke Restart command on SdpSubarrayLeafNode.

**Scan (*argin*)**

Invoke Scan command to SDP subarray.

**SdpSubarrayFQDN**

Used by autodoc\_mock\_imports.

**activeProcessingBlocks**

Used by autodoc\_mock\_imports.

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook ()**

Internal construct of TANGO.

**delete\_device ()**

Internal construct of TANGO.

**init\_command\_objects ()**

Initialises the command handlers for commands supported by this device.

**is\_Abort\_allowed ()**

Checks whether this command is allowed to be run in current device 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\_AssignResources\_allowed ()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**is\_Configure\_allowed ()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**is\_EndScan\_allowed ()**

Checks whether this command is allowed to be run in current device state. :return: True if this command is allowed to be run in current device state. :rtype: boolean

**is\_End\_allowed ()**

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**is\_ObsReset\_allowed ()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean



**is\_ReleaseAllResources\_allowed()**

Checks whether this command is allowed to be run in current device 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\_Restart\_allowed()**

Checks whether this command is allowed to be run in current device 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\_Scan\_allowed()**

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**read\_activeProcessingBlocks()**

Internal construct of TANGO. Returns Active Processing Blocks. activeProcessingBlocks is a forwarded attribute from SDP Subarray which depicts the active Processing Blocks in the SDP Subarray

**read\_activityMessage()**

Internal construct of TANGO. Returns Activity Messages. activityMessage is a String providing information about the current activity in SDP Subarray Leaf Node

**read\_receiveAddresses()**

Internal construct of TANGO. Returns the Receive Addresses. receiveAddresses is a forwarded attribute from SDP Master which depicts State of the SDP.

**receiveAddresses**

Used by autodoc\_mock\_imports.

**sdpSubarrayHealthState**

Used by autodoc\_mock\_imports.

**sdpSubarrayObsState**

Used by autodoc\_mock\_imports.

**validate\_obs\_state()**

**write\_activityMessage(value)**

Internal construct of TANGO. Sets the Activity Message. activityMessage is a String providing information about the current activity in SDP Subarray Leaf Node.

**write\_receiveAddresses(value)**

Internal construct of TANGO. Sets the Receive Addresses. receiveAddresses is a forwarded attribute from SDP Master which depicts State of the SDP.

tmcprototype.sdpSubarrayLeafNode.src.sdpSubarrayLeafNode.sdp\_subarray\_leaf\_node.**main**(args=None, \*\*kwargs)

Runs the SdpSubarrayLeafNode

**Parameters**

- **args** – Arguments internal to TANGO
- **kwargs** – Arguments internal to TANGO

**Returns** SdpSubarrayLeafNode TANGO object

On class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.on\_command.**On** (\*args,  
\*\*kwargs)

A class for SDP Subarray's On() command.

**do** ()

Invokes On command on the SDP Subarray.

**Parameters** **argin** – None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**on\_cmd\_ended\_cb** (event)

Callback function executes when the command invoked asynchronously returns from the server.

**Parameters** **event** – A CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

Off class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.off\_command.**Off** (\*args,  
\*\*kwargs)

A class for SDP Subarray's Off() command.

**do** ()

Invokes Off command on the SDP Subarray.

**Parameters** **argin** – None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**off\_cmd\_ended\_cb** (event)

Callback function executes when the command invoked asynchronously returns from the server.

**Parameters** **event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

AssignResources class for SDPSubarrayLeafNode.

**class** tmcprototype.sdparrayleafnode.src.sdparrayleafnode.assign\_resources\_command.**AssignResourcesCommand**

A class for SdpSubarrayLeafNode's AssignResources() command.

**AssignResources\_ended** (*event*)

This is the callback method of AssignResources command of the SDP Subarray. It checks whether the AssignResources command on SDP subarray is successful.

**Parameters** **argin** – event: response from SDP Subarray for the invoked assign resource command.

**Returns** None

**check\_allowed** ()

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception.

**do** (*argin*)

Assigns resources to given SDP subarray. This command is provided as a noop placeholder from SDP subarray. Eventually this will likely take a JSON string specifying the resource request.

**Parameters** **argin** – The string in JSON format. The JSON contains following values:

**SBI ID and maximum length of the SBI:** Mandatory JSON object consisting of

**SBI ID** : String

**max\_length**: Float

**Scan types:** Consist of Scan type id name

**scan\_type**: DevVarStringArray

**Processing blocks:** Mandatory JSON object consisting of

**processing\_blocks**: DevVarStringArray

**Example:** {"id": "sbi-mvp01-20200325-00001", "max\_length": 100.0, "scan\_types": [{"id": "science\_A", "coordinate\_system": "ICRS", "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]]}], {"id": "calibration\_B", "coordinate\_system": "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": [{"id": "pb-mvp01-20200325-00001", "workflow": {"type": "realtime", "id": "vis\_receive", "version": "0.1.0"}, "parameters": {}}, {"id": "pb-mvp01-20200325-00002", "workflow": {"type": "realtime", "id": "test\_realtime", "version": "0.1.0"}, "parameters": {}}, {"id": "pb-mvp01-20200325-00003", "workflow": {"type": "batch", "id": "ical", "version": "0.1.0"}, "parameters": {}}, {"id": "pb-mvp01-20200325-00004", "workflow": {"type": "batch", "id": "dpreb", "version": "0.1.0"}, "parameters": {}}, {"dependencies": [{"pb\_id": "pb-mvp01-20200325-00001", "type": ["visibilities"]}], {"id": "pb-mvp01-20200325-00004", "workflow": {"type": "batch", "id": "dpreb", "version": "0.1.0"}, "parameters": {}}, {"dependencies": [{"pb\_id": "pb-mvp01-20200325-00003", "type": ["calibration"]}]}]}

Note: Enter input without spaces

**Returns** None

**Raises** ValueError if input argument json string contains invalid value. DevFailed if the command execution is not successful.

ReleaseResources class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.release\_resources\_command.**ReleaseResources**

A class for SdpSubarrayLeafNode's ReleaseAllResources() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception

**do()**

Releases all the resources of given SDPSubarrayLeafNode. It accepts the subarray id, releaseALL flag and receptorIDList in JSON string format.

**Parameters** **argin** – None.

**Returns** None

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

**releaseallresources\_cmd\_ended\_cb(event)**

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the releaseallresources command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name

- `argout_raw` : (DeviceData) The command argout
- `argout` : The command argout
- `err` : (bool) A boolean flag set to true if the command failed. False otherwise
- `errors` : (sequence<DevError>) The error stack
- `ext`

**Returns** none

Configure class for SDPSubarrayLeafNode.

**class** `tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.configure_command.Configure`

A class for SdpSubarrayLeafNode's Configure() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception

**configure\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the configure command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- `device` : (DeviceProxy) The DeviceProxy object on which the call was executed.
- `cmd_name` : (str) The command name
- `argout_raw` : (DeviceData) The command argout
- `argout` : The command argout
- `err` : (bool) A boolean flag set to true if the command failed. False otherwise
- `errors` : (sequence<DevError>) The error stack
- `ext`

**Returns** none

**do** (*argin*)

Configures the SDP Subarray device by providing the SDP PB configuration needed to execute the receive workflow

**Parameters** **argin** – The string in JSON format. The JSON contains following values:

Example:

```
{ "scan_type": "science_A" }
```

**Returns** None

**Raises** ValueError if input argument json string contains invalid value. KeyError if input argument json string contains invalid key. DevFailed if the command execution is not successful

Scan class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan\_command.**Scan** (\*args, \*\*kwargs)

A class for SdpSubarrayLeafNode's Scan() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception.

**do** (argin)

Invoke Scan command to SDP subarray.

**Parameters** **argin** – The string in JSON format. The JSON contains following values:

Example: {"id":1}

Note: Enter input as without spaces:{"id":1}

**Returns** None

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

**scan\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the scan command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object.

**This class is used to pass data to the callback method in asynchronous callback model for command execution.**

**Type** CmdDoneEvent object

**It has the following members:**

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

EndScan class for SDPSubarrayLeafNode.

---

```
class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.endscan_command.EndScan (*args,
**kwargs)
```

A class for SdpSubarrayLeafNode's EndScan() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device state

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception.

**do()**

It invokes EndScan command on SdpSubarray. This command is allowed when SdpSubarray is in SCANNING state.

**Parameters** **argin** – None

**Returns** None

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

**endscan\_cmd\_ended\_cb(event)**

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the endscan command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

End class for SDPSubarrayLeafNode.

```
class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.end_command.End (*args,
**kwargs)
```

A class for SdpSubarrayLeafNode's End() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device state.

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

**Return type** boolean

**Raises** Exception if command execution throws any type of exception.

**do()**

This command invokes End command on SDP subarray to end the current Scheduling block.

**Returns** None

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

**end\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the end command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

Abort class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpSubarrayLeafNode.src.sdpSubarrayLeafNode.abort\_command.**Abort** (*\*args, \*\*kwargs*)

A class for sdpSubarrayLeafNode's Abort() command.

**abort\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the abort command has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack



- ext

**Returns** none

**check\_allowed()**

Checks whether this command is allowed to be run in current device 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

**do()**

Command to abort the current operation being done on the SDP subarray.

**Returns** None

**Raises** DevFailed if error occurs while invoking command on CSPSubarray.

Restart class for SDPSubarrayLeafNode.

**class** tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.restart\_command.**Restart** (\*args, \*\*kwargs)

A class for sdpSubarrayLeafNode's Restart() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device 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

**do()**

Command to restart the SDP subarray and bring it to its ON state.

**Returns** None

**Raises** DevFailed if error occurs while invoking command on SDPSubarray.

**restart\_cmd\_ended\_cb(event)**

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the restart command has been successfully invoked on SDP Subarray.

**Parameters event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

ObsReset class for SDPSubarrayLeafNode.

**class** tmcprototype.sdparrayleafnode.src.sdparrayleafnode.obsreset\_command.**ObsReset** (\*c  
\*\*:

A class for SdpSubarrayLeafNode's ObsResetCommand() command.

**check\_allowed()**

Checks whether this command is allowed to be run in current device 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

**do()**

Command to reset the SDP subarray and bring it to its RESETTING state.

**Parameters** **argin** – None

**Returns** None

**Raises** DevFailed if error occurs while invoking command on SDPSubarray.

**obsreset\_cmd\_ended\_cb(event)**

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the ObsResetCommand has been successfully invoked on SDP Subarray.

**Parameters** **event** – A CmdDoneEvent object.

This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object

**It has the following members:**

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

# CHAPTER 10

---

## CSP Subarray Leaf Node

---

CSP Subarray Leaf node monitors the CSP Subarray and issues control actions during an observation. It also acts as a CSP contact point for Subarray Node for observation execution for TMC.

**class** `tmctype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode`

CSP Subarray Leaf node monitors the CSP Subarray and issues control actions during an observation.

**Abort** ()

Invokes Abort command on CspSubarrayLeafNode

**class** **AbortCommand** (\*args, \*\*kwargs)

A class for CspSubarrayLeafNode's Abort() command.

**abort\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do** ()

This command invokes Abort command on CSP Subarray.

**Returns** None

**Raises** DevFailed if error occurs while invoking command on CSPSubarray.

**AssignResources** (*argin*)

Invokes AssignResources command on CspSubarrayLeafNode.

**class AssignResourcesCommand** (*\*args, \*\*kwargs*)

A class for CspSubarrayLeafNode's AssignResources() command.

**add\_receptors\_ended** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**Raises** DevFailed if this command is not allowed to be run

in current device state

**check\_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

**do** (*argin*)

It accepts receptor id list in JSON string format and invokes AddReceptors command on CspSubarray with receptorIDList (list of integers) as an input argument.

:param argin:DevString. The string in JSON format. The JSON contains following values:

**dish:** Mandatory JSON object consisting of

**receptorIDList:** DevVarString The individual string should contain dish numbers in string format with preceding zeroes upto 3 digits. E.g. 0001, 0002.

Example: {

  “dish”: {

```

        "receptorIDList": [ "0001", "0002"
    ]
}
}

```

Note: Enter the json string without spaces as an input.

**Returns** None

**Raises** ValueError if input argument json string contains invalid value KeyError if input argument json string contains invalid key DevFailed if the command execution is not successful

**Configure** (*argIn*)

Invokes Configure command on CspSubarrayLeafNode

**class ConfigureCommand** (*\*args, \*\*kwargs*)

A class for CspSubarrayLeafNode's Configure() command.

**check\_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

**configure\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**do** (*argIn*)

This command configures a scan. It accepts configuration information in JSON string format and invokes Configure command on CspSubarray.

:param argIn:DevString. The string in JSON format. The JSON contains following values:

Example: {"id": "sbi-mvp01-20200325-00001-science\_A", "frequencyBand": "1", "fsp": [{"fspID": 1, "functionMode": "CORR", "frequencySliceID": 1, "integrationTime": 1400, "corrBandwidth": 0, "channelAveragingMap": [[0, 2], [744, 0]], "fspChannelOffset": 0, "outputLinkMap": [[0, 0], [200, 1]], "outputHost": ["0", "192.168.1.1"], "outputPort": [0, 9000, 1]]}, {"fspID": 2, "functionMode": "CORR", "frequencySliceID": 2, "integrationTime": 1400, "corrBandwidth": 0, "channelAveragingMap": [[0, 2], [744, 0]],

```
“fspChannelOffset”:744,”outputLinkMap”:[[0,4],[200,5]],”outputHost”:  
[[0,”192.168.1.1”]], “outputPort”:[[0,9744,1]]],”delayModelSubscriptionPoint”:  
“ska_mid/tm_leaf_node/csp_subarray01/delayModel”,”pointing”:{“target”:{“system”:”ICRS”,  
“name”:”Polaris Australis”,”RA”:”21:08:47.92”,”dec”:”-88:57:22.9”}}
```

Note: Enter the json string without spaces as an input.

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

**Return type** (ReturnCode, str)

**Raises** DevFailed if the command execution is not successful ValueError if input argument json string contains invalid value

#### **CspSubarrayFQDN**

Used by autodoc\_mock\_imports.

#### **EndScan ()**

Invokes EndScan command on CspSubarrayLeafNode

#### **class EndScanCommand (\*args, \*\*kwargs)**

A class for CspSubarrayLeafNode’s EndScan() command.

#### **check\_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

#### **do ()**

It invokes EndScan command on CspSubarray. This command is allowed when CspSubarray is in obsState SCANNING

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

**Return type** (ReturnCode, str)

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

#### **endscan\_cmd\_ended\_cb (event)**

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise

- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**GoToIdle()**

Invokes GoToIdle command on CspSubarrayLeafNode.

**class GoToIdleCommand** (\*args, \*\*kwargs)

A class for CspSubarrayLeafNode's GoToIdle() command.

**check\_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

**do()**

This command invokes GoToIdle command on CSP Subarray in order to end current scheduling block.

**Returns** None

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

**gotoidle\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

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

A class for the CspSubarrayLeafNode's init\_device() method"

**do()**

Initializes the attributes and properties of the CspSubarrayLeafNode.

**Returns** A tuple containing a return code and a string message indicating status. The message is

for information purpose only.

**Return type** (ReturnCode, str)

**Raises** DevFailed if error occurs in creating proxy for CSPSubarray.

**ObsReset ()**

Invokes ObsReset command on cspsubarrayleafnode

**class ObsResetCommand (\*args, \*\*kwargs)**

A class for CSPSubarrayLeafNode's ObsReset() command.

**check\_allowed ()**

Checks whether this command is allowed to be run in current device 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

**do ()**

Command to reset the CSP subarray and bring it to its RESETTING state.

**Parameters** **argin** – None

**Returns** None

**Raises** DevFailed if error occurs while invoking the command on CSpSubarray.

**obsreset\_cmd\_ended\_cb (event)**

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**ReleaseAllResources ()**

Invokes ReleaseAllResources command on CspSubarrayLeafNode

**class ReleaseAllResourcesCommand (\*args, \*\*kwargs)**

A class for CspSubarrayLeafNode's ReleaseAllResources() command.

**check\_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

**do ()**

It invokes RemoveAllReceptors command on CspSubarray and releases all the resources assigned to CspSubarray.



**Returns** None

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

**releaseallresources\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**Restart** ()

Invokes Restart command on cspsubarrayleafnode

**class RestartCommand** (*\*args, \*\*kwargs*)

A class for CSPSubarrayLeafNode's Restart() command.

**check\_allowed** ()

Checks whether this command is allowed to be run in current device 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

**do** ()

This command invokes Restart command on CSPSubarray.

**Returns** None

**Raises** DevFailed if error occurs while invoking the command on CSpSubarray.

**restart\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout

- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**StartScan** (*argin*)

Invokes StartScan command on cspsubarrayleafnode

**class StartScanCommand** (*\*args, \*\*kwargs*)

A class for CspSubarrayLeafNode's StartScan() command.

**check\_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

**do** (*argin*)

This command invokes Scan command on CspSubarray. It is allowed only when CspSubarray is in ObsState READY.

**Parameters** **argin** – JSON string consists of scan id (int).

Example: {"id":1}

Note: Enter the json string without spaces as a input.

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

**Return type** (ReturnCode, str)

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

**startscan\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook()**

Internal construct of TANGO.

**calculate\_geometric\_delays**(*time\_t0*)

This method calculates geometric delay values (in Second) using KATPoint library. It requires delay correction object, timestamp t0 and target RaDec. Numpy library is used to convert delay values (in Seconds) to fifth order polynomial coefficients. Six timestamps from the time-frame t0 to t+10, are used to calculate delays per antenna. These six delay values are then used to calculate fifth order polynomial coefficients. In order to calculate delays in advance, timestamp t0 is considered to be one minute ahead of the the current timestamp.

**Parameters** *argin* – time\_t0

**Returns** Dictionary containing fifth order polynomial coefficients per antenna per fsp.

**cspSubarrayObsState**

Used by autodoc\_mock\_imports.

**cspsubarrayHealthState**

Used by autodoc\_mock\_imports.

**delayModel**

Used by autodoc\_mock\_imports.

**delay\_model\_calculator**(*argin*)

This method calculates the delay model for consumption of CSP subarray. The epoch value is the current timestamp value. Delay calculation starts when configure command is invoked. It calls the function which internally calculates delay values using KATPoint library and converts them to fifth order polynomial coefficients.

**Parameters** *argin* – int. The argument contains delay model update interval in seconds.

**Returns** None.

**delete\_device()**

Internal construct of TANGO.

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**is\_Abort\_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\_AssignResources\_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\_Configure\_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\_EndScan\_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\_GoToIdle\_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\_ObsReset\_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()**

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\_Restart\_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\_StartScan\_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\_activityMessage()**

Internal construct of TANGO. Returns activity message.

**read\_delayModel()**

Internal construct of TANGO. Returns the delay model.

**read\_versionInfo()**

Internal construct of TANGO. Returns the version information.

**update\_config\_params()**

In this method parameters related to the resources assigned, are updated every time assign, release or configure commands are executed.

**Parameters** *argin* – None

**Returns** None

**validate\_obs\_state()**

**versionInfo**

Used by autodoc\_mock\_imports.

**write\_activityMessage(value)**

Internal construct of TANGO. Sets the activity message.

**write\_delayModel(value)**

Internal construct of TANGO. Sets in to the delay model.

`tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.main(args=None, **kwargs)`

Runs the CspSubarrayLeafNode.

**Parameters**

- **args** – Arguments internal to TANGO
- **kwargs** – Arguments internal to TANGO

**Returns** CspSubarrayLeafNode TANGO object.



---

## SDP Master Leaf Node

---

The primary responsibility of the SDP Subarray Leaf node is to monitor the SDP Subarray and issue control actions during an observation. It also acts as a SDP contact point for Subarray Node for observation execution. There is one to one mapping between SDP Subarray Leaf Node and SDP subarray.

**class** tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.**SdpMasterL**

The primary responsibility of the SDP Subarray Leaf node is to monitor the SDP Subarray and issue control actions during an observation.

**Disable()**

Sets the OperatingState to Disable.

**Parameters** **argin** – None

**Returns** None

**class** **DisableCommand** (\*args, \*\*kwargs)

A class for SDP master's Disable() command.

**check\_allowed()**

Check Whether this command is allowed to be run in current device state.

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

**rtype** boolean

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

**disable\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the disable command has been successfully invoked on SDP Master.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- `device` : (DeviceProxy) The DeviceProxy object on which the call was executed.
- `cmd_name` : (str) The command name
- `argout_raw` : (DeviceData) The command argout
- `argout` : The command argout
- `err` : (bool) A boolean flag set to true if the command failed. False otherwise
- `errors` : (sequence<DevError>) The error stack
- `ext`

**Returns** none

**do** ()

Sets the OperatingState to Disable.

**Parameters** `argin` – None.

**Returns** None

**class** `InitCommand` (\*args, \*\*kwargs)

A class for the SDP master's `init_device()` method"

**do** ()

Initializes the attributes and properties of the `SdpMasterLeafNode`.

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

The message is for information purpose only.

**Return type** (ReturnCode, str)

**Raises**

**class** `OffCommand` (\*args, \*\*kwargs)

A class for SDP master's `Off()` command.

**do** ()

Sets the OperatingState to Off.

**Parameters** `argin` – None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**off\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the OFF command has been successfully invoked on SDP Master.

**Parameters** `event` – a `CmdDoneEvent` object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** `CmdDoneEvent` object It has the following members:

- `device` : (DeviceProxy) The DeviceProxy object on which the call was executed.
- `cmd_name` : (str) The command name
- `argout_raw` : (DeviceData) The command argout



- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**class OnCommand** (\*args, \*\*kwargs)

A class for SDP master's On() command.

**do** ()

Informs the SDP that it can start executing Processing Blocks. Sets the OperatingState to ON.

**Parameters** **argin** – None.

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

The message is for information purpose only.

**Return type** (ResultCode, str)

**on\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the On command has been successfully invoked on SDP Master.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**ProcessingBlockList**

Used by autodoc\_mock\_imports.

**SdpMasterFQDN**

Used by autodoc\_mock\_imports.

**Standby** ()

Invokes Standby command .

**Parameters** **argin** – None

**Returns** None

**class StandbyCommand** (\*args, \*\*kwargs)

A class for SDP Master's Standby() command.

**check\_allowed** ()

Check Whether this command is allowed to be run in current device state.

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

**rtype** boolean

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

**do** ()

Notifies the SDP to stop any executing Processing. To get into the STANDBY state all running PBs will be aborted. In normal operation we expect disable should be triggered without first going into STANDBY.

**Parameters** **argin** – None.

**Returns** None

**is\_Standby\_allowed** ()

Checks Whether this command is allowed to be run in current device 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.

**standby\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns. Checks whether the standby command has been successfully invoked on SDP Master.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook** ()

Internal construct of TANGO.

**delete\_device** ()

Internal construct of TANGO.

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**is\_Disable\_allowed** ()

Checks Whether this command is allowed to be run in current device 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\_ProcessingBlockList** ()

Internal construct of TANGO. :return:

**read\_activityMessage** ()

Internal construct of TANGO. String providing information about the current activity in SDPLeafNode.

**read\_versionInfo** ()

Internal construct of TANGO. Version information of TANGO device.

**sdpHealthState**

Used by autodoc\_mock\_imports.

**versionInfo**

Used by autodoc\_mock\_imports.

**write\_activityMessage** (*value*)

Internal construct of TANGO. Sets the activity message.

tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.**main** (*args=None*,  
\*\**kwargs*)



---

## MCCS Master Leaf Node

---

```
class tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode
```

**Properties:**

- MccsMasterFQDN - Property to provide FQDN of MCCS Master Device

**Attributes:**

- mccsHealthState - Forwarded attribute to provide MCCS Master Health State
- activityMessage - Attribute to provide activity message

**AssignResources** (*argIn*)

Invokes AssignResources command on Mcccs Master

**class AssignResourcesCommand** (*\*args, \*\*kwargs*)

A class for MccsMasterLeafNode's AssignResources() command.

**allocate\_ended** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**Raises** DevFailed if this command is not allowed to be run

in current device state

**check\_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

**do(argin)**

It accepts stationIDList list, channels and stationBeamIDList in JSON string format and invokes allocate command on MccsMaster with JSON string as an input argument.

:param argin:StringType. The string in JSON format.

Example: {

```
    "subarray_id": 1, "station_ids": [1,2], "channels": [1,2,3,4,5,6,7,8], "sta-
    tion_beam_ids": [1]
```

}

**Returns** None

Note: Enter the json string without spaces as an input.

**Raises** ValueError if input argument json string contains invalid value KeyError if input argument json string contains invalid key DevFailed if the command execution is not successful

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

A class for the TMC MCCS Master Leaf Node's init\_device() method.

**do()**

Initializes the attributes and properties of the MccsMasterLeafNode.

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

**Return type** (ResultCode, str)

**Raises** DevFailed if error occurs while creating the device proxy for Mccs Master or subscribing the events.

**MccsMasterFQDN**

Used by autodoc\_mock\_imports.

**class OffCommand(\*args, \*\*kwargs)**

A class for MccsMasterLeafNode's Off() command.

**do()**

Invokes Off command on the MCCS Element.

**Parameters** **argin** – None.

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

**Return type** (ResultCode, str)

**off\_cmd\_ended\_cb(event)**

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**class OnCommand** (\*args, \*\*kwargs)

A class for MccsMasterLeafNode's On() command.

**do** ()

Invokes On command on the MCCS Element.

**Parameters** **argin** – None

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

**Return type** (ResultCode, str)

**on\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**ReleaseResources** (argin)

Invokes ReleaseResources command on MccsMasterLeafNode

**class ReleaseResourcesCommand** (\*args, \*\*kwargs)

A class for MccsMasterLeafNode's ReleaseResources() command.

**check\_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** ValueError if input argument json string contains invalid value DevFailed if this command is not allowed to be run in current device state

**do** (*argin*)

It invokes ReleaseResources command on MccsMaster and releases all the resources assigned to MccsMaster.

:param argin:StringType. The string in JSON format.

**Example:**

```
{ "subarray_id": 1, "release_all": true,
}
```

**Returns** None.

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

**releaseresources\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook** ()

Internal construct of TANGO.

**delete\_device** ()

Internal construct of TANGO.

**init\_command\_objects** ()

Initialises the command handlers for commands supported by this device.

**is\_AssignResources\_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



**is\_ReleaseResources\_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

**mccsHealthState**

Used by autodoc\_mock\_imports.

**read\_activityMessage()**

**write\_activityMessage**(*value*)

tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_leaf\_node.**main**(*args=None*,  
\*\**kwargs*)

Runs the MccsMasterLeafNode.

**Parameters**

- **args** – Arguments internal to TANGO
- **kwargs** – Arguments internal to TANGO

**Returns** An object of CompletedProcess class returned by the subprocess.



---

## MCCS Subarray Leaf Node

---

MCCS Subarray Leaf node monitors the MCCS Subarray and issues control actions during an observation. It also acts as a MCCS contact point for Subarray Node for observation execution for TMC.

**class** `tmctype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.M`

MCCS Subarray Leaf node monitors the MCCS Subarray and issues control actions during an observation.

**Configure** (*argin*)

Invokes Configure command on MccsSubarrayLeafNode

**class ConfigureCommand** (*\*args, \*\*kwargs*)

A class for MccsSubarrayLeafNode's Configure() command.

**check\_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

**configure\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise

- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**do** (*argin*)

This command configures a scan. It accepts configuration information in JSON string format and invokes Configure command on MccsSubarray.

:param argin:DevString. The string in JSON format. The JSON contains following values:

Example: {“stations”:[{“station\_id”:1},{“station\_id”:2}],“station\_beam\_pointings”:[{“station\_beam\_id”:1,”target”:

Note: Enter the json string without spaces as a input.

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

**Return type** (ReturnCode, str)

**Raises** DevFailed if the command execution is not successful ValueError if input argument json string contains invalid value KeyError if input argument json string contains invalid key

**End** ()

Invokes End command on MccsSubarrayLeafNode.

**class EndCommand** (*\*args, \*\*kwargs*)

A class for MccsSubarrayLeafNode’s End() command.

**check\_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

**do** ()

This command invokes End command on MCCS Subarray in order to end current scheduling block.

**Returns** None

**Return type** Void

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

**end\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise

- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

**EndScan()**

Invokes EndScan command on MccsSubarray.

**class EndScanCommand** (\*args, \*\*kwargs)

A class for MccsSubarrayLeafNode's EndScan() command.

**check\_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

**do()**

This command invokes EndScan command on MccsSubarray. It is allowed only when MccsSubarray is in ObsState SCANNING.

**Raises** DevFailed if the command execution is not successful. AssertionError if MccsSubarray is not in SCANNING obsState.

**endscan\_cmd\_ended\_cb** (event)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- device : (DeviceProxy) The DeviceProxy object on which the call was executed.
- cmd\_name : (str) The command name
- argout\_raw : (DeviceData) The command argout
- argout : The command argout
- err : (bool) A boolean flag set to true if the command failed. False otherwise
- errors : (sequence<DevError>) The error stack
- ext

**Returns** none

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

A class for the MccsSubarrayLeafNode's init\_device() method"

**do()**

Initializes the attributes and properties of the MccsSubarrayLeafNode.

**Returns** A tuple containing a return code and a string message indicating status. The message is

for information purpose only.

**Return type** (ReturnCode, str)

**Raises** DevFailed if error occurs in creating proxy for MCCSSubarray.

**MccsSubarrayFQDN**

Used by autodoc\_mock\_imports.

**Scan** (*argin*)

Invokes Scan command on mcssubarrayleafnode

**class ScanCommand** (*\*args, \*\*kwargs*)

A class for MccsSubarrayLeafNode's Scan() command.

**check\_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

**do** (*argin*)

This command invokes Scan command on MccsSubarray. It is allowed only when MccsSubarray is in ObsState READY.

**Parameters** **argin** – JSON string consists of scan id (int).

Example: {"id":1}

Note: Enter the json string without spaces as a input.

**Returns** None

**Return type** Void

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

**scan\_cmd\_ended\_cb** (*event*)

Callback function immediately executed when the asynchronous invoked command returns.

**Parameters** **event** – a CmdDoneEvent object. This class is used to pass data to the callback method in asynchronous callback model for command execution.

**Type** CmdDoneEvent object It has the following members:

- **device** : (DeviceProxy) The DeviceProxy object on which the call was executed.
- **cmd\_name** : (str) The command name
- **argout\_raw** : (DeviceData) The command argout
- **argout** : The command argout
- **err** : (bool) A boolean flag set to true if the command failed. False otherwise
- **errors** : (sequence<DevError>) The error stack
- **ext**

**Returns** none

**activityMessage**

Used by autodoc\_mock\_imports.

**always\_executed\_hook** ()

Internal construct of TANGO.

**delete\_device** ()

Internal construct of TANGO.

**init\_command\_objects()**

Initialises the command handlers for commands supported by this device.

**is\_Configure\_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\_EndScan\_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\_End\_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\_Scan\_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

**mccsSubarrayHealthState**

Used by autodoc\_mock\_imports.

**mccsSubarrayObsState**

Used by autodoc\_mock\_imports.

**read\_activityMessage()**

**write\_activityMessage(value)**

tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_subarray\_leaf\_node.**main**(arg  
\*\*k





## CHAPTER 14

---

### Indices and tables

---

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



---

## Python Module Index

---

### t

|   |   |
|---|---|
|   | tmcprototype.subarraynode.src.subarraynode.abort_command                  |
| tmcprototype.centralnode.src.centralnode.central_node,                              | 18  |
| 1   | tmcprototype.subarraynode.src.subarraynode.assign_resources_command       |
| tmcprototype.centralnodelow.src.centralnodelow.central_node_low,                    | 14  |
| 5   | tmcprototype.subarraynode.src.subarraynode.configure_command              |
| tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node,          | 16  |
| 37  | tmcprototype.subarraynode.src.subarraynode.end_command                    |
| tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node,    | 17  |
| 55  | tmcprototype.subarraynode.src.subarraynode.end_scan_command               |
| tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node,                          | 17  |
| 27  | tmcprototype.subarraynode.src.subarraynode.obsreset_command               |
| tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node,       | 19  |
| 73  | tmcprototype.subarraynode.src.subarraynode.off_command                    |
| tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node,          | 13  |
| 67  | tmcprototype.subarraynode.src.subarraynode.on_command                     |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.abort_command,             | 13  |
| 52  | tmcprototype.subarraynode.src.subarraynode.release_resources_command      |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.assign_resources_command,  | 15  |
| 47  | tmcprototype.subarraynode.src.subarraynode.restart_command                |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.configure_command,         | 18  |
| 49  | tmcprototype.subarraynode.src.subarraynode.scan_command                   |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.end_command,               | 16  |
| 51  | tmcprototype.subarraynode.src.subarraynode.subarraynode                   |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.endscan_command,           | 11  |
| 50  | tmcprototype.subarraynode.src.subarraynode.track_command                  |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset_command,          | 19  |
| 54  | tmcprototype.subarraynodelow.src.subarraynodelow.assign_resources_command |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.off_command,               | 23  |
| 46  | tmcprototype.subarraynodelow.src.subarraynodelow.configure_command        |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.on_command,                | 23  |
| 46  | tmcprototype.subarraynodelow.src.subarraynodelow.end_command              |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.release_resources_command, | 23  |
| 48  | tmcprototype.subarraynodelow.src.subarraynodelow.end_scan_command         |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.restart_command,           | 24  |
| 53  | tmcprototype.subarraynodelow.src.subarraynodelow.obsreset_command         |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan_command,              | 24  |
| 50  | tmcprototype.subarraynodelow.src.subarraynodelow.off_command              |
| tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node,    | 24  |
| 43  | tmcprototype.subarraynodelow.src.subarraynodelow.on_command               |

`tmcprototype.subarraynodelow.src.subarraynodelow.scan_command,`  
25  
`tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low,`  
21

## A

Abort (class in tmcproto- activityMessage (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.abort\_command), 52  
 attribute), 39  
 Abort (class in tmcproto- activityMessage (tmcproto-  
 type.subarraynode.src.subarraynode.abort\_command), 18  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray-  
 attribute), 62  
 Abort (in module tmcproto- activityMessage (tmcproto-  
 type.subarraynode.src.subarraynode.subarray\_node), 13  
 type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
 attribute), 32  
 Abort () (tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.CspSubarrayLeafNode  
 method), 55  
 type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_l-  
 attribute), 76  
 Abort () (tmcprototype.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
 method), 27  
 activityMessage (tmcproto-  
 type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_suba-  
 attribute), 82  
 Abort () (tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray-  
 method), 43  
 abort\_cmd\_ended\_cb () (tmcproto- activityMessage (tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray-  
 attribute), 55  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray-  
 attribute), 70  
 abort\_cmd\_ended\_cb () (tmcproto- activityMessage (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.abort\_command), 52  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray-  
 attribute), 44  
 abort\_csp () (tmcproto- activityMessage (tmcproto-  
 type.subarraynode.src.subarraynode.abort\_command.Abort), 18  
 type.subarraynode.src.subarraynode.subarray\_node.SubarrayNode  
 attribute), 11  
 abort\_dishes () (tmcproto- activityMessage (tmcproto-  
 type.subarraynode.src.subarraynode.abort\_command.Abort), 18  
 type.subarraynode\_low.src.subarraynode\_low.subarray\_node\_low.S  
 attribute), 21  
 abort\_sdp () (tmcproto- add\_receptors\_ended () (tmcproto-  
 type.subarraynode.src.subarraynode.abort\_command.Abort), 18  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray-  
 method), 56  
 activeProcessingBlocks (tmcproto- allocate\_ended () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray-  
 attribute), 44  
 type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_l-  
 method), 73  
 activityMessage (tmcproto- always\_executed\_hook () (tmcproto-  
 type.centralnode.src.centralnode.central\_node.CentralNode  
 attribute), 2  
 method), 2  
 activityMessage (tmcproto- always\_executed\_hook () (tmcproto-  
 type.centralnode\_low.src.centralnode\_low.central\_node\_low.Centra  
 method), 7  
 type.centralnode\_low.src.centralnode\_low.central\_node\_low.Centra  
 method), 7

`always_executed_hook()` (tmcproto- `AssignResources()` (tmcproto-  
`type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode`  
`method`), 39 `method`), 43  
`always_executed_hook()` (tmcproto- `AssignResources_ended()` (tmcproto-  
`type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode`  
`method`), 62 `method`), 47  
`always_executed_hook()` (tmcproto- `AssignResourcesCommand` (class in tmcproto-  
`type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.McCsMasterLeafNode`  
`method`), 76 23  
`always_executed_hook()` (tmcproto- `attribute_event_handler()` (tmcproto-  
`type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.McCsSubarrayLeafNode`  
`method`), 82 `method`), 32  
`always_executed_hook()` (tmcproto- **B**  
`type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode`  
`method`), 70 `build_up_csp_cmd_data()` (tmcproto-  
`always_executed_hook()` (tmcproto- `type.subarraynode.src.subarraynode.configure_command.Element`  
`type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode`  
`method`), 44 `method`), 46  
`always_executed_hook()` (tmcproto- `build_up_dsh_cmd_data()` (tmcproto-  
`type.subarraynode.src.subarraynode.subarray_node.SubarrayNode`  
`method`), 12 `method`), 16  
`always_executed_hook()` (tmcproto- `build_up_sdp_cmd_data()` (tmcproto-  
`type.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNodeLow`  
`method`), 21 `method`), 14  
`assign_csp_resources()` (tmcproto- **C**  
`type.subarraynode.src.subarraynode.assign_resources_command.AssignResources`  
`method`), 14 `calculate_geometric_delays()` (tmcproto-  
`assign_sdp_resources()` (tmcproto- `type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_`  
`type.subarraynode.src.subarraynode.assign_resources_command.AssignResources`  
`method`), 14 `method`), 63  
`AssignResources` (class in tmcproto- `calculate_observation_state()` (tmcproto-  
`type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.assign_resources_command`  
47 `method`), 21  
`AssignResources` (class in tmcproto- `type.subarraynodelow.src.subarraynodelow.subarray_node_low.S`  
`type.subarraynode.src.subarraynode.assign_resources_command`  
14 `method`), 17  
`AssignResources` (in module tmcproto- `call_end_scan_command()` (tmcproto-  
`type.centralnode.src.centralnode.central_node`), 3 `method`), 25  
`AssignResources` (in module tmcproto- `type.subarraynodelow.src.subarraynodelow.scan_command.Scan`  
`type.subarraynode.src.subarraynode.subarray_node`  
12 `method`), 25  
`AssignResources()` (tmcproto- `CentralAlarmHandler` (tmcproto-  
`type.centralnode.src.centralnode.central_node.CentralNode`  
`method`), 1 `attribute`), 1  
`AssignResources()` (tmcproto- `CentralAlarmHandler` (tmcproto-  
`type.centralnodelow.src.centralnodelow.central_node_low.Centra`  
`method`), 1 `attribute`), 6  
`AssignResources()` (tmcproto- `CentralNode` (class in tmcproto-  
`type.centralnodelow.src.centralnodelow.central_node_low.CentralNode`  
`method`), 5 `method`), 1  
`AssignResources()` (tmcproto- `CentralNode` (class in tmcproto-  
`type.centralnodelow.src.centralnodelow.central_node_low`), 5  
`AssignResources()` (tmcproto- `type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode`  
`method`), 56 `method`), 56  
`AssignResources()` (tmcproto- (class in tmcproto-  
`type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.McCsMasterLeafNode`  
`method`), 73 `method`), 73

```

CentralNode.InitCommand (class in tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node),
                        1 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
                        method), 61
CentralNode.InitCommand (class in tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node_low),type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
                        6 method), 62
CentralNode.ReleaseResourcesCommand check_allowed() (tmcproto-
                        (class in tmcproto- type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        type.centralnode.src.centralnode.central_node_low),method), 27
                        6 check_allowed() (tmcproto-
CentralNode.StandByTelescopeCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        (class in tmcproto- method), 27
                        type.centralnode.src.centralnode.central_node_low),method), 27
                        6 check_allowed() (tmcproto-
CentralNode.StartupTelescopeCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        (class in tmcproto- method), 28
                        type.centralnode.src.centralnode.central_node_low),type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        7 check_allowed() (tmcproto-
                        method), 28
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node_low:CentralNode.AssignResourcesCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 5 method), 29 method), 29
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node_low:CentralNode.ReleaseResourcesCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 6 method), 29 method), 29
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node_low:CentralNode.StandByTelescopeCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 6 method), 30 method), 30
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.centralnode.src.centralnode.central_node_low:CentralNode.StartupTelescopeCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 7 method), 30 method), 30
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspmasterleafnode.src.cspmasterleafnode.csp_master_ type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 39 method), 31 method), 31
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.AssignResourcesCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 55 method), 31 method), 31
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.AssignResourcesCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 56 method), 31 method), 31
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.ConfigureCrossmaster_L type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 57 method), 74 method), 74
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.EndScanCommand type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 58 method), 75 method), 75
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.CycleOffSubarray mode type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 59 method), 79 method), 79
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.CycleOffSubarray mode type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 60 method), 80 method), 80
check_allowed() (tmcproto- check_allowed() (tmcproto-
                        type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.csp_subarrayleafnode.ReleaseSubarray mode type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.
                        method), 60 method), 81 method), 81

```

```

check_allowed() (tmcproto- Configure() (tmcproto-
    type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_type.cspsubarrayleafnode.src.cspsubarray
    method), 82 method), 57
check_allowed() (tmcproto- Configure() (tmcproto-
    type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_type.disableSdpMasterLeafNodeDisableConfmode.DishLeafNode
    method), 67 method), 27
check_allowed() (tmcproto- Configure() (tmcproto-
    type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_type.mccssubarrayleafnode.src.mccssubarray
    method), 69 method), 79
check_allowed() (tmcproto- Configure() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.abort_command.sdpsubarrayleafnode.src.sdpsubarray
    method), 53 method), 43
check_allowed() (tmcproto- configure_cmd_ended_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.assign_resource_command.AssignResource
    method), 47 method), 57
check_allowed() (tmcproto- configure_cmd_ended_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.configure_command.Configure
    method), 49 method), 79
check_allowed() (tmcproto- configure_cmd_ended_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.end_command.sdpsubarrayleafnode.src.sdpsubarray
    method), 51 method), 49
check_allowed() (tmcproto- ConfigureCommand (class in tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.endscan_command.EndScan
    method), 51 23
check_allowed() (tmcproto- convert_radec_to_azel() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset_command.ObsReset.dishleafnode.dish_leaf_node.DishLeafNode
    method), 54 method), 32
check_allowed() (tmcproto- csp_cbf_health_state_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.release_resource_command.ReleaseAllResources
    method), 48 method), 39
check_allowed() (tmcproto- csp_pss_health_state_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.restart_command.Restart
    method), 53 method), 40
check_allowed() (tmcproto- csp_pst_health_state_cb() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan_command.Scan
    method), 50 method), 40
check_allowed() (tmcproto- cspHealthState (tmcproto-
    type.subarraynode.src.subarraynode.track_command.Track type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n
    method), 19 attribute), 39
check_only_dish_config() (tmcproto- CspMasterFQDN (tmcproto-
    type.subarraynode.src.subarraynode.configure_command.Configure
    method), 16 attribute), 37
command_class_object() (tmcproto- CspMasterLeafNode (class in tmcproto-
    type.subarraynodelow.src.subarraynodelow.subarray_node_type.SubarrayNode
    method), 21 37
Configure (class in tmcproto- CspMasterLeafNode.InitCommand
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.configure_command), in tmcproto-
    49 type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n
Configure (class in tmcproto- 37
    type.subarraynode.src.subarraynode.configure_command.CspMasterLeafNode.OffCommand
    16 (class in tmcproto-
Configure (in module tmcproto- type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n
    type.subarraynode.src.subarraynode.subarray_node), 37
    12 CspMasterLeafNode.OnCommand

```



```

    (class in tmcproto- CspSubarrayLeafNode.RestartCommand
    type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node), in tmcproto-
38 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
CspMasterLeafNode.StandbyCommand 61
    (class in tmcproto- CspSubarrayLeafNode.StartScanCommand
    type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node), in tmcproto-
39 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
CspMasterLeafNodeFQDN (tmcproto- 62
    type.centralnode.src.centralnode.central_node.CentralNode (tmcproto-
    attribute), 1 type.subarraynode.src.subarraynode.subarray_node.SubarrayNode
CspSubarrayFQDN (tmcproto- attribute), 11
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode (tmcproto-
    attribute), 58 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
CspSubarrayFQDN (tmcproto- attribute), 63
    type.subarraynode.src.subarraynode.subarray_node.SubarrayNode
    attribute), 11
D
cspsubarrayHealthState (tmcproto- delay_model_calculator() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    attribute), 63 method), 63
CspSubarrayLeafNode (class in tmcproto- delayModel (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), (tmcproto-
    attribute), 63
CspSubarrayLeafNode.AbortCommand delete_device() (tmcproto-
    (class in tmcproto- type.centralnode.src.centralnode.central_node.CentralNode
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node),
    55
    delete_device() (tmcproto-
CspSubarrayLeafNode.AssignResourcesCommand type.centralnode.src.centralnode_low.central_node_low.CentralNodeLow
    (class in tmcproto- method), 7
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), (tmcproto-
    56 type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node
CspSubarrayLeafNode.ConfigureCommand method), 40
    (class in tmcproto- delete_device() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), (tmcproto-
    57 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_
    method), 63
CspSubarrayLeafNode.EndScanCommand delete_device() (tmcproto-
    (class in tmcproto- type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node),
    58 method), 1
CspSubarrayLeafNode.GoToIdleCommand delete_device() (tmcproto-
    (class in tmcproto- type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), (tmcproto-
    59 method), 82
CspSubarrayLeafNode.InitCommand delete_device() (tmcproto-
    (class in tmcproto- type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), (tmcproto-
    59 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_
    method), 44
CspSubarrayLeafNode.ObsResetCommand delete_device() (tmcproto-
    (class in tmcproto- type.subarraynode.src.subarraynode.subarray_node.SubarrayNode
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node),
    60 method), 1
CspSubarrayLeafNode.ReleaseAllResourcesCommand delete_device() (tmcproto-
    (class in tmcproto- type.subarraynode_low.src.subarraynode_low.subarray_node_low.SubarrayNodeLow
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node), module tmcproto-
    60 type.centralnode.src.centralnode.central_node),

```

```

3
Disable() (tmcproto- type.dishleafnode.src.dishleafnode.dish_leaf_node),
type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode
method), 67
disable_cmd_ended_cb() (tmcproto- (class in tmcproto-
type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNodeDisableCommand),
method), 67 30
dishHealthState (tmcproto- DishLeafNode.StopCaptureCommand
type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute), 32 in tmcproto-
attribute), 32 type.dishleafnode.src.dishleafnode.dish_leaf_node),
DishLeafNode (class in tmcproto- 31
type.dishleafnode.src.dishleafnode.dish_leaf_node), 27 DishLeafNode.StopTrackCommand
DishLeafNode.AbortCommand (class in tmcproto- type.dishleafnode.src.dishleafnode.dish_leaf_node),
type.dishleafnode.src.dishleafnode.dish_leaf_node), 27 31
DishLeafNode.ConfigureCommand (class in tmcproto-
type.dishleafnode.src.dishleafnode.dish_leaf_node), 27 31 DishLeafNodePrefix
DishLeafNode.EndScanCommand (class in tmcproto- (tmcproto-
type.dishleafnode.src.dishleafnode.dish_leaf_node), 28 type.centralnode.src.centralnode.central_node.CentralNode
attribute), 1
DishLeafNode.InitCommand (class in tmcproto- DishLeafNodePrefix (tmcproto-
type.dishleafnode.src.dishleafnode.dish_leaf_node), 28 type.subarraynode.src.subarraynode.subarray_node.SubarrayNode
attribute), 11
DishLeafNode.ObsResetCommand dishMasterFQDN (tmcproto-
(class in tmcproto- type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode
type.dishleafnode.src.dishleafnode.dish_leaf_node), 28 attribute), 28
DishLeafNode.RestartCommand dishPointingState (tmcproto-
(class in tmcproto- type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode
type.dishleafnode.src.dishleafnode.dish_leaf_node), 28 attribute), 32
DishLeafNode.ScanCommand (class in tmcproto- do() (tmcprototype.centralnode.src.centralnode.central_node.CentralNode
type.dishleafnode.src.dishleafnode.dish_leaf_node), 29 method), 1
DishLeafNode.SetOperateModeCommand do() (tmcprototype.centralnode.src.centralnode.central_node_low.central_node_low.CentralNodeLow), 5
(class in tmcproto- method), 5
type.dishleafnode.src.dishleafnode.dish_leaf_node), 29 do() (tmcprototype.centralnode.src.centralnode.central_node_low.central_node_low.CentralNodeLow), 6
DishLeafNode.SetStandbyFPModeCommand do() (tmcprototype.centralnode.src.centralnode.central_node_low.central_node_low.CentralNodeLow), 6
(class in tmcproto- method), 6
type.dishleafnode.src.dishleafnode.dish_leaf_node), 29 do() (tmcprototype.centralnode.src.centralnode.central_node_low.central_node_low.CentralNodeLow), 7
DishLeafNode.SetStandbyLPModeCommand do() (tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode), 37
(class in tmcproto- method), 37
type.dishleafnode.src.dishleafnode.dish_leaf_node), 29 do() (tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode), 38
DishLeafNode.SetStowModeCommand do() (tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode), 39
(class in tmcproto- method), 39
type.dishleafnode.src.dishleafnode.dish_leaf_node), 30 do() (tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_node.CspSubarrayNode), 56
DishLeafNode.StartCaptureCommand do() (tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_node.CspSubarrayNode), 56
(class in tmcproto- method), 56
type.dishleafnode.src.dishleafnode.dish_leaf_node), 30

```

Index 95

```
do () (tmcprototype.subarraynode.src.subarraynode.end_command.EndScanCommand) method), 17
do () (tmcprototype.subarraynode.src.subarraynode.end_scan_command.EndScanCommand) method), 17
do () (tmcprototype.subarraynode.src.subarraynode.obsreset_command.ObsResetCommand) method), 19
do () (tmcprototype.subarraynode.src.subarraynode.off_command.OffCommand) method), 13
do () (tmcprototype.subarraynode.src.subarraynode.on_command.OnCommand) method), 13
do () (tmcprototype.subarraynode.src.subarraynode.release_all_resources_command.ReleaseAllResourcesCommand) method), 15
do () (tmcprototype.subarraynode.src.subarraynode.restart_command.RestartCommand) method), 18
do () (tmcprototype.subarraynode.src.subarraynode.scan_command.ScanCommand) method), 16
do () (tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode.InitCommand) method), 11
do () (tmcprototype.subarraynode.src.subarraynode.track_command.TrackCommand) method), 19
do () (tmcprototype.subarraynodelow.src.subarraynodelow.assign_resources_command.AssignResourcesCommand) method), 23
do () (tmcprototype.subarraynodelow.src.subarraynodelow.configure_command.ConfigureCommand) method), 23
do () (tmcprototype.subarraynodelow.src.subarraynodelow.end_command.EndCommand) method), 23
do () (tmcprototype.subarraynodelow.src.subarraynodelow.endscan_command.EndScanCommand) method), 24
do () (tmcprototype.subarraynodelow.src.subarraynodelow.off_command.OffCommand) method), 24
do () (tmcprototype.subarraynodelow.src.subarraynodelow.on_command.OnCommand) method), 24
do () (tmcprototype.subarraynodelow.src.subarraynodelow.release_all_resources_command.ReleaseAllResourcesCommand) method), 24
do () (tmcprototype.subarraynodelow.src.subarraynodelow.scan_command.ScanCommand) method), 25
do () (tmcprototype.subarraynodelow.src.subarraynodelow.subarray_type.sdp_subarray_init.SdpSubarrayInitCommand) method), 21
ElementDeviceData (class in tmcprototype.subarraynode.src.subarraynode.configure_command) method), 16
End (class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.end_command) method), 51
End (class in tmcprototype.subarraynode.src.subarraynode.end_command) method), 17
End (in module tmcprototype.type.subarraynode.src.subarraynode.subarray_node) method), 13
End () (tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode) method), 80
```



`type.subarraynode.src.subarraynode.end_scan_command_objects()` (tmcproto-  
 24 `type.subarraynode.src.subarraynode.subarray_node_low.S`  
 method), 22

**G**

`get_deviceproxy()` (tmcproto- `type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_`  
`type.subarraynode.src.subarraynode.subarray_node_low.SubarrayNode`  
 method), 22

`GoToIdle()` (tmcproto- `type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode`  
`type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.CspSubarrayLeafNode`  
 method), 59

`gotoidle_cmd_ended_cb()` (tmcproto- `type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_`  
`type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.CspSubarrayLeafNode.GoToIdleCommand`  
 method), 59

**H**

`health_state_cb()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.centralnode.src.centralnode.central_node_low.CentralNode`  
 method), 7

`health_state_cb()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.subarraynode.src.subarraynode.subarray_node_low.SubarrayNode`  
 method), 22

**I**

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.centralnode.src.centralnode.central_node.CentralNode`  
 method), 2

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.centralnode.src.centralnode.central_node_low.CentralNode`  
 method), 8

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode`  
 method), 40

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarrayleafnode.CspSubarrayLeafNode`  
 method), 63

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode`  
 method), 32

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode`  
 method), 76

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode`  
 method), 82

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode`  
 method), 70

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode`  
 method), 44

`init_command_objects()` (tmcproto- `is_AssignResources_allowed()` (tmcproto-  
`type.subarraynode.src.subarraynode.subarray_node.SubarrayNode`  
 method), 12

```

is_EndScan_allowed() (tmcproto- is_Standby_allowed() (tmcproto-
    type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode
    method), 83 method), 40
is_EndScan_allowed() (tmcproto- is_Standby_allowed() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode
    method), 44 method), 70
is_GoToIdle_allowed() (tmcproto- is_StandByTelescope_allowed() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    method), 64 method), 2
is_ObsReset_allowed() (tmcproto- is_StandByTelescope_allowed() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    method), 64 method), 8
is_ObsReset_allowed() (tmcproto- is_StartCapture_allowed() (tmcproto-
    type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode
    method), 32 method), 33
is_ObsReset_allowed() (tmcproto- is_StartScan_allowed() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode
    method), 44 method), 64
is_ReleaseAllResources_allowed() (tmcproto- is_StartUpTelescope_allowed() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    method), 64 method), 2
is_ReleaseAllResources_allowed() (tmcproto- is_StartUpTelescope_allowed() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode
    method), 44 method), 8
is_ReleaseResources_allowed() (tmcproto- is_StopCapture_allowed() (tmcproto-
    type.centralnode.src.centralnode.central_node.CentralNode
    method), 2 method), 33
is_ReleaseResources_allowed() (tmcproto- is_StopTrack_allowed() (tmcproto-
    type.centralnodelow.src.centralnodelow.central_node_low.CentralNodeLow
    method), 8 method), 33
is_ReleaseResources_allowed() (tmcproto- is_StowAntennas_allowed() (tmcproto-
    type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode
    method), 76 method), 3
is_Restart_allowed() (tmcproto- is_Track_allowed() (tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    method), 64 method), 33
is_Restart_allowed() (tmcproto- is_Track_allowed() (tmcproto-
    type.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode
    method), 32 method), 12
is_Restart_allowed() (tmcproto-
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode
    method), 45
is_Scan_allowed() (tmcproto- main() (in module tmcproto-
    type.centralnode.src.centralnode.central_node),
    method), 33
is_Scan_allowed() (tmcproto- main() (in module tmcproto-
    type.centralnodelow.src.centralnodelow.central_node_low),
    type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode
    method), 83
is_Scan_allowed() (tmcproto- main() (in module tmcproto-
    type.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode
    type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode
    method), 45
is_Slew_allowed() (tmcproto- main() (in module tmcproto-
    type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode
    method), 33

```

```

type.dishleafnode.src.dishleafnode.dish_leaf_node), attribute), 81
33 MccsSubarrayFQDN (tmcproto-
main() (in module tmcproto- type.subarraynodelow.src.subarraynodelow.subarray_node_low.S
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
77 mccsSubarrayHealthState (tmcproto-
main() (in module tmcproto- type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node),
83 MccsSubarrayLeafNode (class in tmcproto-
main() (in module tmcproto- type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
type.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node),
71 MccsSubarrayLeafNode.ConfigureCommand
main() (in module tmcproto- (class in tmcproto-
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node),
45 79 MccsSubarrayLeafNode.EndCommand
main() (in module tmcproto- MccsSubarrayLeafNode.EndScanCommand
type.subarraynode.src.subarraynode.subarray_node), (class in tmcproto-
12 type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
main() (in module tmcproto- 80 MccsSubarrayLeafNode.EndScanCommand
type.subarraynodelow.src.subarraynodelow.subarray_nodelow), (class in tmcproto-
22 type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
mccsHealthState (tmcproto- type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode
attribute), 77 MccsSubarrayLeafNode.InitCommand
MccsMasterFQDN (tmcproto- (class in tmcproto-
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode
attribute), 74 81 MccsSubarrayLeafNode.ScanCommand
MccsMasterLeafNode (class in tmcproto- MccsSubarrayLeafNode.ScanCommand
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node), in tmcproto-
73 type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
MccsMasterLeafNode.AssignResourcesCommand 82
(class in tmcproto- MccsSubarrayLNFQDN (tmcproto-
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
73 attribute), 21
MccsMasterLeafNode.InitCommand mccsSubarrayObsState (tmcproto-
(class in tmcproto- type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
74 attribute), 82
MccsMasterLeafNode.OffCommand N
(class in tmcproto- NumDishes (tmcproto-
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
74 type.centralnode.src.centralnode.central_node.CentralNode
attribute), 1
MccsMasterLeafNode.OnCommand O
(class in tmcproto- observation_state_cb() (tmcproto-
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
75 type.subarraynodelow.src.subarraynodelow.subarray_node_low.S
MccsMasterLeafNode.ReleaseResourcesCommand method), 22
(class in tmcproto- ObsReset (class in tmcproto-
type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node),
75 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset_com
54
MCCSMasterLeafNodeFQDN (tmcproto- ObsReset (class in tmcproto-
type.centralnodelow.src.centralnodelow.central_node_low.CentralNode
attribute), 6 19 type.subarraynode.src.subarraynode.obsreset_command),
MccsSubarrayFQDN (tmcproto- ObsReset (in module tmcproto-
type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode
type.subarraynode.src.subarraynode.subarray_node),

```

13  
 ObsReset () (tmcproto- On (class in tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarrayleafnode.CspSubarrayLeafNode\_command),  
 method), 59

ObsReset () (tmcproto- On (in module tmcproto-  
 type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode\_subarraynode.src.subarraynode.subarray\_node),  
 method), 28

ObsReset () (tmcproto- on\_cmd\_ended\_cb () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode.SdpSubarrayLeafNode\_leafnode.csp\_master\_leaf\_n  
 method), 43

obsreset\_cmd\_ended\_cb () (tmcproto- on\_cmd\_ended\_cb () (tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarrayleafnode.CspSubarrayLeafNode\_obsreset\_cmd\_ended\_cb  
 method), 60

obsreset\_cmd\_ended\_cb () (tmcproto- on\_cmd\_ended\_cb () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset\_cmd\_ended\_cb.SdpSubarrayLeafNode\_obsreset\_cmd\_ended\_cb  
 method), 54

obsreset\_csp () (tmcproto- on\_cmd\_ended\_cb () (tmcproto-  
 type.subarraynode.src.subarraynode.obsreset\_command.ObsResetCommand\_obsreset\_csp.SdpSubarrayLeafNode\_obsreset\_csp  
 method), 19

obsreset\_dsh\_grp () (tmcproto- OnCommand (class in tmcproto-  
 type.subarraynode.src.subarraynode.obsreset\_command.ObsResetCommand\_obsreset\_dsh\_grp.SdpSubarrayNode\_low.src.subarraynode\_low.on\_command),  
 method), 19

obsreset\_sdp () (tmcproto-  
 type.subarraynode.src.subarraynode.obsreset\_command.ObsResetCommand\_obsreset\_sdp.SdpSubarrayNode\_low.src.subarraynode\_low.on\_command),  
 method), 19

ObsStateAggregator (in module tmcproto-  
 type.centralnode.src.centralnode.central\_node),  
 3

Off (class in tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.off\_command),  
 46

Off (class in tmcproto-  
 type.subarraynode.src.subarraynode.off\_command),  
 13

Off (in module tmcproto-  
 type.subarraynode.src.subarraynode.subarray\_node),  
 13

off\_cmd\_ended\_cb () (tmcproto-  
 type.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.CspMasterLeafNode\_off\_command),  
 method), 37

off\_cmd\_ended\_cb () (tmcproto-  
 type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_leaf\_node.MccsMasterLeafNode\_off\_command),  
 method), 74

off\_cmd\_ended\_cb () (tmcproto-  
 type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode\_off\_command),  
 method), 68

off\_cmd\_ended\_cb () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.off\_command.OffCommand\_off\_cmd\_ended\_cb.DishLeafNode  
 method), 46

OffCommand (class in tmcproto-  
 type.subarraynode\_low.src.subarraynode\_low.off\_command),  
 24

On (class in tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.on\_command),  
 method), 83

46  
 (class in tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarrayleafnode.CspSubarrayLeafNode\_command),  
 13  
 (in module tmcproto-  
 type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode\_subarraynode.src.subarraynode.subarray\_node),  
 13  
 on\_cmd\_ended\_cb () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode.SdpSubarrayLeafNode\_leafnode.csp\_master\_leaf\_n  
 method), 38  
 on\_cmd\_ended\_cb () (tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarrayleafnode.CspSubarrayLeafNode\_obsreset\_cmd\_ended\_cb  
 method), 75  
 on\_cmd\_ended\_cb () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset\_cmd\_ended\_cb.SdpSubarrayLeafNode\_obsreset\_cmd\_ended\_cb  
 method), 69  
 on\_cmd\_ended\_cb () (tmcproto-  
 type.subarraynode.src.subarraynode.obsreset\_command.ObsResetCommand\_obsreset\_csp.SdpSubarrayLeafNode\_obsreset\_csp  
 method), 46  
 OnCommand (class in tmcproto-  
 type.subarraynode\_low.src.subarraynode\_low.on\_command),  
 24  
 P  
 ProcessingBlockList (tmcproto-  
 type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode\_processing\_block\_list.SdpMasterLeafNode\_processing\_block\_list  
 attribute), 69  
 R  
 read\_activityMessage () (tmcproto-  
 type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarraynode.SdpSubarrayNode\_read\_activity\_message.SdpSubarrayNode\_read\_activity\_message  
 method), 45  
 read\_activityMessage () (tmcproto-  
 type.centralnode.src.centralnode.central\_node.CentralNode\_read\_activity\_message.CentralNode\_read\_activity\_message  
 method), 3  
 read\_activityMessage () (tmcproto-  
 type.centralnode\_low.src.centralnode\_low.central\_node\_low.CentralNode\_low\_read\_activity\_message.CentralNode\_low\_read\_activity\_message  
 method), 8  
 read\_activityMessage () (tmcproto-  
 type.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.CspMasterLeafNode\_read\_activity\_message.CspMasterLeafNode\_read\_activity\_message  
 method), 40  
 read\_activityMessage () (tmcproto-  
 type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarraynode.CspSubarrayNode\_read\_activity\_message.CspSubarrayNode\_read\_activity\_message  
 method), 65  
 read\_activityMessage () (tmcproto-  
 type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode\_read\_activity\_message.DishLeafNode\_read\_activity\_message  
 method), 33  
 read\_activityMessage () (tmcproto-  
 type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_leaf\_node.MccsMasterLeafNode\_read\_activity\_message.MccsMasterLeafNode\_read\_activity\_message  
 method), 77  
 read\_activityMessage () (tmcproto-  
 type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_subarraynode.MccsSubarrayNode\_read\_activity\_message.MccsSubarrayNode\_read\_activity\_message  
 method), 83





method), 76

Restart (class in tmcproto-type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan\_command), 50

Restart (class in tmcproto-type.subarraynode.src.subarraynode.scan\_command.Scan), 17

Restart (in module tmcproto-type.subarraynode.src.subarraynode.subarray\_node), 17

Restart () (tmcproto-type.subarraynode.src.subarraynode.scan\_command.ScanThread), 17

Restart () (tmcproto-type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node.CspSubarrayLeafNode), 61

Restart () (tmcproto-type.subarraynode.src.subarraynode.scan\_command), 29

Restart () (tmcproto-type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan\_command), 43

restart\_cmd\_ended\_cb () (tmcproto-ScanThread (class in tmcproto-type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node.CspSubarrayLeafNodeRestartCommand), 61

restart\_cmd\_ended\_cb () (tmcproto-sdpHealthState (tmcproto-type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.restart\_command.SdpRestartCommand), 53

restart\_dsh\_grp () (tmcproto-SdpMasterFQDN (tmcproto-type.subarraynode.src.subarraynode.restart\_command.RestartCommand), 18

restart\_leaf\_nodes () (tmcproto-SdpMasterLeafNode (class in tmcproto-type.subarraynode.src.subarraynode.restart\_command.RestartCommand), 18

S

sbID (tmcprototype.subarraynode.src.subarraynode.subarray\_node.SubarrayNode (class in tmcproto-type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode), 12

Scan (class in tmcproto-type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.scan\_command), 50

Scan (class in tmcproto-type.subarraynode.src.subarraynode.scan\_command), 16

Scan (in module tmcproto-type.subarraynode.src.subarraynode.subarray\_node), 12

Scan () (tmcprototype.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode (class in tmcproto-type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode), 29

Scan () (tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_subarray\_leaf\_node.MccsSubarrayLeafNode), 82

Scan () (tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray\_leaf\_node.SdpSubarrayLeafNode (class in tmcproto-type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode), 44

scan\_cmd\_ended\_cb () (tmcproto-SdpMasterLeafNode.FQDN (tmcproto-type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_subarray\_leaf\_node.MccsSubarrayLeafNode), 82

scan\_cmd\_ended\_cb () (tmcproto-type.centralnode.src.centralnode.central\_node.CentralNode), 1

SdpSubarrayFQDN (tmcproto- Standby () (tmcproto-  
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode.sdp\_master\_leaf\_node.  
attribute), 44 method), 69

SdpSubarrayFQDN (tmcproto- standby\_cmd\_ended\_cb () (tmcproto-  
type.subarraynode.src.subarraynode.subarray\_node.SubarrayNode.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.  
attribute), 11 method), 39

sdpSubarrayHealthState (tmcproto- standby\_cmd\_ended\_cb () (tmcproto-  
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode.sdp\_master\_leaf\_node.sdp\_master\_leaf\_node.  
attribute), 45 method), 70

SdpSubarrayLeafNode (class in tmcproto- StandByTelescope (in module tmcproto-  
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode), 3  
43

SdpSubarrayLeafNode.InitCommand StandByTelescope () (tmcproto-  
(class in tmcproto- type.centralnode.src.centralnode.central\_node.CentralNode  
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode),  
43 method), 43

SdpSubarrayLNFQDN (tmcproto- type.centralnode.src.centralnode.central\_node\_low.CentralNodeLow  
type.subarraynode.src.subarraynode.subarray\_node.SubarrayNode), 6  
attribute), 11

sdpSubarrayObsState (tmcproto- type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarrayleafnode.SdpSubarrayLeafNode  
attribute), 45 method), 45

set\_csp\_client () (tmcproto- type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_node.CspSubarrayNode  
type.subarraynode.src.subarraynode.on\_command.On method), 62  
method), 13

set\_dish\_name\_number () (tmcproto- type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_node.CspSubarrayNode  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode), 62  
method), 33

set\_observer\_lat\_long\_alt () (tmcproto- type.centralnode.src.centralnode.central\_node),  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
method), 33

set\_sdp\_client () (tmcproto- type.centralnode.src.centralnode.central\_node.CentralNode  
type.subarraynode.src.subarraynode.on\_command.On method), 2  
method), 13

set\_up\_dish\_data () (tmcproto- type.centralnode\_low.src.centralnode\_low.central\_node\_low.CentralNodeLow  
type.subarraynode.src.subarraynode.assign\_resources\_command.AssignResources  
method), 15

SetOperateMode () (tmcproto- type.subarraynode.src.subarraynode.end\_command.End  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode), 18  
method), 29

SetStandbyFPMode () (tmcproto- type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode), 31  
method), 29

SetStandbyLPMode () (tmcproto- type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode), 31  
method), 30

SetStowMode () (tmcproto- type.centralnode.src.centralnode.central\_node),  
type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
method), 30

Slew () (tmcproto- type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode, type.centralnode.src.centralnode.central\_node.CentralNode  
method), 30

Standby () (tmcproto- subarray1HealthState (tmcproto-  
type.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node.CspMasterLeafNode, type.centralnode.src.centralnode.central\_node.CentralNode  
method), 38 attribute), 3

subarray1HealthState (tmcproto- tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 type.centralnode.src.centralnode.central\_node\_low.CentralNode( (module), 47  
 attribute), 8 tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 subarray2HealthState (tmcproto- (module), 49  
 type.centralnode.src.centralnode.central\_node.CentralNode tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 attribute), 3 (module), 51  
 subarray3HealthState (tmcproto- tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 type.centralnode.src.centralnode.central\_node.CentralNode(module), 50  
 attribute), 3 tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 SubarrayNode (class in tmcproto- (module), 54  
 type.subarraynode.src.subarraynode.subarray\_node\_low.CentralNode tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 11 (module), 46  
 SubarrayNode (class in tmcproto- tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 type.subarraynode.src.subarraynode.subarray\_node\_low.CentralNode (module), 46  
 21 tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 SubarrayNode.InitCommand (class in tmcproto- (module), 48  
 type.subarraynode.src.subarraynode.subarray\_node\_low.CentralNode tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 11 (module), 53  
 SubarrayNode.InitCommand (class in tmcproto- tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 type.subarraynode.src.subarraynode.subarray\_node\_low.CentralNode (module), 50  
 21 tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.  
 (module), 43  
 T tmcprototype.subarraynode.src.subarraynode.abort\_command (module), 18  
 telescopeHealthState (tmcproto- (module), 18  
 type.centralnode.src.centralnode.central\_node.CentralNode tmcprototype.subarraynode.src.subarraynode.assign\_command (module), 14  
 attribute), 3 (module), 14  
 telescopeHealthState (tmcproto- tmcprototype.subarraynode.src.subarraynode.configure\_command (module), 16  
 type.centralnode.src.centralnode.central\_node\_low.CentralNode tmcprototype.subarraynode.src.subarraynode.end\_command (module), 17  
 attribute), 8 tmcprototype.subarraynode.src.subarraynode.end\_scan\_command (module), 17  
 TMAAlarmHandler (tmcproto- (module), 17  
 type.centralnode.src.centralnode.central\_node.CentralNode tmcprototype.subarraynode.src.subarraynode.obsreset\_command (module), 19  
 attribute), 2 tmcprototype.subarraynode.src.subarraynode.off\_command (module), 13  
 TMAAlarmHandler (tmcproto- tmcprototype.subarraynode.src.subarraynode.on\_command (module), 13  
 type.centralnode.src.centralnode.central\_node\_low.CentralNode tmcprototype.subarraynode.src.subarraynode.release\_command (module), 13  
 attribute), 7 tmcprototype.subarraynode.src.subarraynode.restart\_command (module), 15  
 tmcprototype.centralnode.src.centralnode.central\_node\_low.CentralNode tmcprototype.subarraynode.src.subarraynode.scan\_command (module), 16  
 (module), 1 tmcprototype.subarraynode.src.subarraynode.subarraynode\_low (module), 13  
 tmcprototype.centralnode.src.centralnode.central\_node\_low.CentralNode tmcprototype.subarraynode.src.subarraynode.release\_command (module), 13  
 (module), 5 tmcprototype.subarraynode.src.subarraynode.restart\_command (module), 15  
 tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_node (module), 37  
 tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node (module), 55  
 tmcprototype.dishleafnode.src.dishleafnode.dish\_leaf\_node (module), 27  
 tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_leaf\_node (module), 73  
 tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_subarray\_leaf\_node (module), 79  
 tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node (module), 67  
 tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdpsubarray\_leaf\_node (module), 52  
 tmcprototype.subarraynode.src.subarraynode.abort\_command (module), 23  
 tmcprototype.subarraynode.src.subarraynode.end\_scan\_command (module), 23

tmcprototype.subarraynodelow.src.subarraynodelow.end\_scan\_command  
     (module), 24      write\_activityMessage() (tmcproto-  
 tmcprototype.subarraynodelow.src.subarraynodelow.off\_command  
     (module), 24      type.centralnode.src.centralnode.central\_node.CentralNode  
                         method), 3  
 tmcprototype.subarraynodelow.src.subarraynodelow.on\_command  
     (module), 24      write\_activityMessage() (tmcproto-  
                         type.centralnodelow.src.centralnodelow.central\_node\_low.Centra  
                         method), 9  
 tmcprototype.subarraynodelow.src.subarraynodelow.release\_all\_resources\_command  
     (module), 24      write\_activityMessage() (tmcproto-  
 tmcprototype.subarraynodelow.src.subarraynodelow.scan\_command  
     (module), 25      type.cspmasterleafnode.src.cspmasterleafnode.csp\_master\_leaf\_n  
                         method), 40  
 tmcprototype.subarraynodelow.src.subarraynodelow.subarray\_node\_low  
     (module), 21      write\_activityMessage() (tmcproto-  
                         type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_  
                         method), 65  
 TMLowSubarrayNodes (tmcproto-  
     type.centralnodelow.src.centralnodelow.central\_node\_low.CentralNode  
     attribute), 7      write\_activityMessage() (tmcproto-  
                         type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
                         method), 33  
 TMMidSubarrayNodes (tmcproto-  
     type.centralnode.src.centralnode.central\_node.CentralNode  
     attribute), 2      write\_activityMessage() (tmcproto-  
                         type.mccsmasterleafnode.src.mccsmasterleafnode.mccs\_master\_l  
                         method), 77  
 Track (class in tmcproto-  
     type.subarraynode.src.subarraynode.track\_command)  
     19      write\_activityMessage() (tmcproto-  
                         type.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs\_suba  
                         method), 83  
 Track (in module tmcproto-  
     type.subarraynode.src.subarraynode.subarray\_node)  
     13      write\_activityMessage() (tmcproto-  
                         type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_n  
                         method), 71  
 Track() (tmcprototype.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
     method), 31      write\_activityMessage() (tmcproto-  
 Track() (tmcprototype.subarraynode.src.subarraynode.subarray\_node.SubarrayNode  
     method), 11      type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray\_  
                         method), 45  
 track\_thread() (tmcproto-  
     type.dishleafnode.src.dishleafnode.dish\_leaf\_node.DishLeafNode  
     method), 33      write\_activityMessage() (tmcproto-  
                         type.subarraynode.src.subarraynode.subarray\_node.SubarrayNode  
                         method), 12  
 turn\_on\_leaf\_node() (tmcproto-  
     type.subarraynode.src.subarraynode.on\_command.On  
     method), 13      write\_activityMessage() (tmcproto-  
                         type.subarraynodelow.src.subarraynodelow.subarray\_node\_low.S  
                         method), 22  
                         write\_delayModel() (tmcproto-  
                         type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_  
                         method), 65  
 update\_config\_params() (tmcproto-  
     type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node.CspSubarrayLeafNode  
     method), 65      type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray\_  
                         method), 45  
 U  
 V  
 validate\_obs\_state() (tmcproto-  
     type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node.CspSubarrayLeafNode  
     method), 65  
 validate\_obs\_state() (tmcproto-  
     type.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp\_subarray\_leaf\_node.SdpSubarrayLeafNode  
     method), 45  
 versionInfo (tmcproto-  
     type.cspsubarrayleafnode.src.cspsubarrayleafnode.csp\_subarray\_leaf\_node.CspSubarrayLeafNode  
     attribute), 65  
 versionInfo (tmcproto-  
     type.sdpmasterleafnode.src.sdpmasterleafnode.sdp\_master\_leaf\_node.SdpMasterLeafNode  
     attribute), 71