TMC Prototype Documentation

Release 1.0

NCRA India

Oct 06, 2020

Contents:

1 Central Node 1

2 Central Node 9

3 Subarray Node 15

4 Subarray Node 19

5 Dish Leaf Node 23

6 Dish Master 33

7 CSP Master Leaf Node 39

8 SDP Subarray Leaf Node 43

9 CSP Subarray Leaf Node 55

10 SDP Master Leaf Node 67

11 MCCS Master Leaf Node 73

12 MCCS Subarray Leaf Node 79

13 Indices and tables 85

Python Module Index 87

Index 89

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.

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, receptor id list and SDP block in JSON string format. Upon successful execution, the 'receptorIDList' attribute of the given subarray is populated with the given receptors.Also checking for duplicate allocation of resources is done. If already allocated it will throw error message regarding the prior existence of resource.

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

subarrayID: DevShort. Mandatory.

dish: Mandatory JSON object consisting of

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

sdp: Mandatory JSON object consisting of

id: DevString The SBI id.

max_length: DevDouble Maximum length of the SBI in seconds.

scan_types: array of the blocks each consisting following parameters id:

DevString The scan id.

coordinate_system: DevString
ra: DevString
Dec: DevString

processing_blocks: array of the blocks each consisting following parameters id: DevString The Processing Block id.

workflow:
type: DevString
id: DevString
version: DevString
parameters: {}

Example: {“subarrayID”:1,”dish”:{“receptorIDList”:[“0001”,”0002”]},”sdp”:{“id”:”sbi-mvp01-20200325-00001”, “max_length”:100.0,”scan_types”:[{“id”:”science_A”,”coordinate_system”:”ICRS”,”ra”:”02: ,”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”:{},”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: From Jive, enter above input string without any space.

Returns

A tuple containing a return code and a string in JSON format on successful assignment of given resources. The JSON string contains following values:

dish: Mandatory JSON object consisting of

receptorIDList_success: DevVarStringArray Contains ids of the receptors which are successfully allocated. Empty on unsuccessful allocation.

Example: { “dish”: { “receptorIDList_success”: [“0001”, “0002”] } }

Return type (ResultCode, str)

Raises DevFailed when the API fails to allocate resources.

Note: Enter input without spaces as:{“dish”:{“receptorIDList_success”:[“0001”,”0002”]}}

`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.

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, releaseALL flag and receptorIDList in JSON string format. When the releaseALL flag is True, ReleaseAllResources command is invoked on the respective SubarrayNode. In this case, the receptorIDList tag is empty as all the resources of the Subarray are to be released. When releaseALL is False, ReleaseResources will be invoked on the SubarrayNode and the resources provided in receptorIDList tag, are to be released from the Subarray. The selective release of the resources when releaseALL Flag is False is not yet supported.

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

subarrayID: DevShort. Mandatory.

releaseALL: Boolean(True or False). Mandatory. True when all the resources to be released from Subarray.

receptorIDList: DevVarStringArray. Empty when releaseALL tag is True.

Example:

{ "subarrayID": 1, "releaseALL": true, "receptorIDList": []

}

Note: From Jive, enter input as: {"subarrayID":1,"releaseALL":true,"receptorIDList":[]} without any space.

Returns

A tuple containing a return code and a string in josn format on successful release of all the resources. The JSON string contains following values:

releaseALL: Boolean(True or False). If True, all the resources are successfully released from the Subarray.

receptorIDList: DevVarStringArray. If releaseALL is True, receptorIDList is empty. Else list returns resources (device names) that are noe released from the subarray.

Example: argout = {

”ReleaseAll” : True, “receptorIDList” : []

}

rtype (ResultCode, str)

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

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.

class StandByTelescopeCommand(**args*, ***kwargs*)
A class for 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 CentralNode into OFF state. Invokes the respective command on lower level nodes adn devices.

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, DishLeafNode, CSPMasterLeafNode or SDpMasterLeafNode

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

class StartUpTelescopeCommand(**args*, ***kwargs*)
A class for 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, DishLeafNode, CSPMasterLeafNode or SDpMasterLeafNode

StowAntennas(*argin*)

This command stows the specified receptors.

class StowAntennasCommand(**args*, ***kwargs*)

A class for CentralNode's StowAntennas() 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 the command SetStowMode on the specified receptors.

Parameters **argin** – List of Receptors to be stowed.

Returns None

Raises DevFailed if error occurs while invoking command of DishLeafNode ValueError if error occurs if input argument json string contains invalid value

TMAlarmHandler

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.

health_state_cb(*evt*)

Retrieves the subscribed Subarray health state, aggregates them to calculate the telescope health state.

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

Returns None

Raises KeyError if error occurs while setting Subarray 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

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.

obs_state_cb(*evt*)
Retrieves the subscribed Subarray observation state. When the Subarray obsState is EMPTY, the resource allocation list gets cleared.

Parameters **evt** – A TANGO_CHANGE event on Subarray obsState.

Returns None

Raises KeyError in Subarray obsState callback

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.

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, tile ids list 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

tile_ids: DevArray. Mandatory the list of tiles that should be allocated to the Sub-Array

Example:

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

}

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_i

`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, releaseALL flag in JSON string format. When the releaseALL 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.

releaseALL: Boolean(True or False). Mandatory. True when all the resources to be released from Subarray.

Example:

{ "subarray_id": 1, "releaseALL": true,

}

Note: From Jive, enter input as: {"subarray_id":1,"releaseALL":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.

TMAlarmHandler

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.

CHAPTER 3

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.

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*)
Retrieves the subscribed health states, aggregates them to calculate the overall subarray health state.

Parameters **event** – A TANGO_CHANGE event on Subarray healthState.

Returns None

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

observation_state_cb(*evt*)
Retrieves the subscribed CSP_Subarray AND SDP_Subarray obsState.

Parameters **evt** – A TANGO_CHANGE event on CSP and SDP Subarray obsState.

Returns None

pointing_state_cb(*evt*)
Retrieves the subscribed DishMaster health state, aggregate them to evaluate health state of the Subarray.

Parameters **evt** – A TANGO_CHANGE event on DishMaster healthState.

Returns None

read_activityMessage()
Internal construct of TANGO. Returns activityMessage. Example: “Subarray node is initialized successfully” //result occured 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.

receive_addresses_cb(*event*)
Retrieves the receiveAddresses attribute of SDP Subarray.

Parameters **`event`** – A TANGO_CHANGE event on SDP Subarray receiveAddresses attribute.

Returns None

`receptorIDList`
Used by autodoc_mock_imports.

`remove_receptors_from_group`()
Deletes tango group of the resources allocated in the subarray.

Note: Currently there are only receptors allocated so the group contains only receptor ids.
Parameters **`argin`** – DevVoid

Returns DevVoid

`sbID`
Used by autodoc_mock_imports.

`scanID`
Used by autodoc_mock_imports.

`validate_obs_state`()

`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.

CHAPTER 4

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.subarraynodelow.src.subarraynodelow.subarray_node_low.**SubarrayNode**(**args*, ***kwa*

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.

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*)
Retrieves the subscribed health states, aggregates them to calculate the overall subarray health state.

Parameters **event** – A TANGO_CHANGE event on Subarray healthState.

Returns None

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

observation_state_cb(*evt*)
Retrieves the subscribed CSP_Subarray AND SDP_Subarray obsState.

Parameters **evt** – A TANGO_CHANGE event on CSP and SDP Subarray obsState.

Returns None

pointing_state_cb(*evt*)
Retrieves the subscribed DishMaster health state, aggregate them to evaluate health state of the Subarray.

Parameters **evt** – A TANGO_CHANGE event on DishMaster healthState.

Returns None

read_activityMessage()
Internal construct of TANGO. Returns activityMessage. Example: “Subarray node is initialized successfully” //result occured 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.

receive_addresses_cb(*event*)
Retrieves the receiveAddresses attribute of SDP Subarray.

Parameters **`event`** – A TANGO_CHANGE event on SDP Subarray receiveAddresses attribute.

Returns None

`receptorIDList`
Used by autodoc_mock_imports.

`remove_receptors_from_group``()`
Deletes tango group of the resources allocated in the subarray.

Note: Currently there are only receptors allocated so the group contains only receptor ids.
Parameters **`argin`** – DevVoid

Returns DevVoid

`sbID`
Used by autodoc_mock_imports.

`scanID`
Used by autodoc_mock_imports.

`validate_obs_state``()`

`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.

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

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

`do`()

Invokes Abort command on the DishMaster.

Parameters **`argin`** – DevVoid

Returns None

Raises DevFailed if error ocuurs while invoking command on DishMaster.

`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

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

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”}}

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 this command on DishMaster. ValueError if argin is not in valid JSON format. KeyError if JSON key is not present in argin

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

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

do(*argin*)

Invokes EndScan command on DishMaster.

Parameters **argin** – timestamp

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

Return type (ResultCode, str)

Example: 10.0

Raises ValueError if argin is of invalid (other than float) data type while invoking this command.

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 proxy for DishMaster or in subscribing the event on 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

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

do()

Command to reset the Dishleaf Node and bring it to its RESETTING state.

Parameters **argin** – None

Returns None

Raises DevFailed if error occurs while invoking command on Dishleaf Node.

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

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

do()

Invokes Restart command on the DishMaster.

Parameters **argin** – DevVoid

Returns None

raises: DevFailed if error occurs while invoking command on DishMaster Exception if error occurs while executing the command

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

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

Return type (ResultCode, str)

Example: 10.0

Raises ValurError if argin is of invalid (other than float) data type while invoking this command.

SetOperateMode()

Invokes SetOperateMode command on DishMaster.

class SetOperateModeCommand(**args*, ***kwargs*)

A class for DishLeafNode's SetOperateMode() 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 SetOperateMode command on DishMaster.

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

Return type (ResultCode, str)

SetStandByLPMode()

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

class SetStandByLPModeCommand(**args*, ***kwargs*)

A class for DishLeafNode's SetStandByLPMode() 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

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

do()

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

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

Return type (ResultCode, str)

SetStandbyFPMode()

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

class SetStandbyFPModeCommand(**args*, ***kwargs*)
A class for DishLeafNode's SetStandByFPMode() 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()
Invokes SetStandbyFPMode command on DishMaster (Standby-Full power) mode.

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

Return type (ResultCode, str)

SetStowMode()
Invokes SetStowMode command on DishMaster.

class SetStowModeCommand(**args*, ***kwargs*)
A class for DishLeafNode's SetStowMode() 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()
Invokes SetStowMode command on DishMaster.

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

Return type (ResultCode, str)

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

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

do(*argin*)
Invokes Slew command on DishMaster to slew the dish towards the set pointing coordinates.

Parameters **argin** – timestamp

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 ValueError if argin is not in valid JSON format while invoking this 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

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

`do`(*argin*)
Invokes StartCapture command on DishMaster on the set configured band.

Parameters **`argin`** – timestamp

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 ValueError if argin is not in valid JSON format while invoking this 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

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

`do`(*argin*)
Invokes StopCapture command on DishMaster on the set configured band.

Parameters **`argin`** – timestamp

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 ValueError if argin is not in valid JSON format while invoking this 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

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

do()
Invokes StopTrack command on the DishMaster.

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 this 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

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

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"}}

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 JSONDecodeError if argin is not a valid JSON format, KeyError if JSON key is not present in argin while invoking this command on DishMaster.

TrackDuration
Used by autodoc_mock_imports.

activityMessage
Used by autodoc_mock_imports.

always_executed_hook()
Internal construct of TANGO.

attribute_event_handler(*event_data*)
Retrieves the subscribed attribute of DishMaster.

Parameters **evt** – A TANGO_CHANGE event on attribute.

Returns None

convert_radec_to_azel(*data*)
Converts RaDec coordinate in to AzEl coordinate using KATPoint library.

Parameters **data** – DevVarStringArray

Argin to be provided is the Ra and Dec values in the following format: radec|21:08:47.92|89:15:51.4 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 None.

Raises Exception if error occurs in Ra-Dec to Az-El conversion

delete_device()
Internal construct of TANGO.

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_SetOperateMode_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_SetStandByLPMode_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_SetStandbyFPMode_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_SetStowMode_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_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 invokes Track command on DishMaster at the rate of 20 Hz.

Returns None.

`tracking_time_thread``()`

This thread allows the dish to track the source for a specified Duration.

Returns None.

`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

SKA Dish Master TANGO device server

class `tmcprototype.dishmaster.src.dishmaster.dish_master.`**`DishMaster`**(**args*, ***kwargs*)

SKA Dish Master TANGO device server

`Abort`()

This command aborts the Track or Scan operation when invoked.

`AzElOffset`

Used by autodoc_mock_imports.

`Configure`(*argin*)

Configures the pointing parameters of the dish.

Parameters **`argin`** –

DevString. JSON string consists of Azimuth(decimal degrees), Elevation(decimal degrees) and receiverBand.

Example:

{ “pointing”: {“AZ”: 1.0,”EL”: 1.0},

”dish”: {“receiverBand”:”1”}

}

Returns None.

Input from jive: {“pointing”:{“AZ”:1.0,”EL”:1.0},”dish”:{“receiverBand”:”1”}} without any space.

`ConfiguredBand`

Used by autodoc_mock_imports.

`ObsReset`()

Currently this command works similar to the Restart command. It clears Dish configuration parameters.

`ReceptorNumber`

Used by autodoc_mock_imports.

`Restart`()
: This command restarts the Dish.

`Scan`(*argin*)
: Triggers the dish to start scanning at the set pointing coordinates and capture the data at the input timestamp.

  Parameters **`argin`** – DevString. Timestamp in UTC at which command should be executed.

  Returns None

`SetMaintenanceMode`()
: Triggers the Dish to transition into the MAINTENANCE Dish Element Mode. This mode will also enable engineers and maintainers to upgrade SW and FW. Dish also enters this mode when an emergency stop button is pressed.

`SetOperateMode`()
: Triggers the Dish to transition into the OPERATE Dish Element Mode.

  Returns None

`SetStandbyFPMode`()
: Triggers the Dish to transition into the STANDBY-FP (Standby-Full power) Dish Element Mode.

  Returns None

`SetStandbyLPMode`()
: Triggers the Dish to transition into the STANDBY-LP (Standby-Low power) Dish Element Mode. Standby-LP is the default mode when the Dish is configured for low power consumption. It is the mode wherein Dish ends after a start up procedure.

`SetStowMode`()
: Triggers the Dish to transition into the STOW Dish Element Mode. Used to point the dish in a direction that minimises the wind loads on the structure,for survival in strong wind conditions. The Dish is able to observe in the stove position, for the purpose of transient detection.

`Slew`(*argin=0*)
: Triggers the Dish to move (or slew) at the commanded pointing coordinates.

  Parameters **`argin`** – DevString. Timestamp in UTC at which command should be executed.

  Returns None

`StartCapture`(*argin*)
: Triggers the dish to start capturing the data on the configured band.

  Parameters **`argin`** – DevString. Timestamp in UTC at which command should be executed.

  Returns None

`StopCapture`(*argin*)
: Triggers the dish to stop capturing the data on the configured band.

  Parameters **`argin`** – DevString. Timestamp in UTC at which command should be executed.

  Returns None

`StopTrack`()
: This command is created only for making pointingState = 0 in Track command.

`Track`(*argin*)
: Triggers Track on Dish. It accepts changes in DesiredPointing attribute value and tracks the source. When difference between Achieved and Desired pointing attributes is more than the pointing limits, Dish starts

to slewing. Whereas Dish moves in tracking mode, when the difference becomes less than the pointing limits.

Parameters **`argin`** – DevString. Timestamp in UTC at which command should be executed.

Returns None

`WindSpeed`
Used by autodoc_mock_imports.

`achievedPointing`
Used by autodoc_mock_imports.

`always_executed_hook`()
Internal construct of TANGO.

`azimuth`()
Calculates the azimuth angle difference.

`azimuthOverWrap`
Used by autodoc_mock_imports.

`band1SamplerFrequency`
Used by autodoc_mock_imports.

`band2SamplerFrequency`
Used by autodoc_mock_imports.

`band3SamplerFrequency`
Used by autodoc_mock_imports.

`band4SamplerFrequency`
Used by autodoc_mock_imports.

`band5aSamplerFrequency`
Used by autodoc_mock_imports.

`band5bSamplerFrequency`
Used by autodoc_mock_imports.

`capturing`
Used by autodoc_mock_imports.

`check_slew`()
Waits until the Dish is slewing and stows it later.

Returns None

`decrement_position`(*argin*)
Decrements the current pointing coordinates gradually to match the desired pointing coordinates.

Parameters **`argin`** – Difference between current and desired Azimuth/Elevation angle.

Returns None

`delete_device`()
Internal construct of TANGO.

`desiredPointing`
Used by autodoc_mock_imports.

`dishMode`
Used by autodoc_mock_imports.

`elevation`()
Calculates the elevation angle difference.

increment_position(*argin*)
Increments the current pointing coordinates gradually to match the desired pointing coordinates.
Parameters **argin** – Difference between current and desired Azimuth/Elevation angle.

Returns None

init_device()
Initializes the properties and attributes of DishMaster.
Returns None

is_Scan_allowed()
Checks if the Scan is allowed in the current state of DishMaster.

is_SetMaintenanceMode_allowed()
Checks if SetMaintenanceMode is allowed in the current state of DishMaster.

is_SetOperateMode_allowed()
Checks if SetOperateMode is allowed in the current state of DishMaster.

is_SetStandbyFPMode_allowed()
Checks if the SetStandbyFPMode is allowed in the current state of DishMaster.

is_SetStandbyLPMode_allowed()
Checks if the SetStandbyLPMode is allowed in the current pointing state of DishMaster.

is_SetStowMode_allowed()
Checks if the SetStowMode is allowed in the current state of DishMaster.

is_StartCapture_allowed()
Checks if the StartCapture is allowed in the current state of DishMaster.

is_StopCapture_allowed()
Checks if the StopCapture is allowed in the current state of DishMaster.

point()
Points the dish towards the desired pointing coordinates.

pointingState
Used by autodoc_mock_imports.

read_AzElOffset()
Internal construct of TANGO. Returns Azimuth and Elevation pointing limits of Dish.

read_ConfiguredBand()
Internal construct of TANGO. Returns the band configured for the Dish.

read_WindSpeed()
Internal construct of TANGO. Returns the Wind speed.

read_achievedPointing()
Internal construct of TANGO. Returns the achieved pointing coordinates of Dish.

read_azimuthOverWrap()
Internal construct of TANGO. Returns boolean to notify if Dish Azimuth is beyond Azimuth wrap limit.

read_capturing()
Internal construct of TANGO. Returns true if the dish is capturing the data, else false.

read_desiredPointing()
Internal construct of TANGO. Returns the desired pointing coordinates of Dish.

read_dishMode()
Internal construct of TANGO. Returns the dishMode.

read_pointingState()
Internal construct of TANGO. Returns the pointingState.

read_toggleFault()
Internal construct of TANGO.Returns the toggleFault .

toggleFault
Used by autodoc_mock_imports.

track_slew()
Completes slewing of Dish in 10 steps.

Returns None

write_WindSpeed(*value*)
Internal construct of TANGO. Sets the wind speed.

Parameters **value** – WindSpeed

Returns None

write_band1SamplerFrequency(*value*)
Internal construct of TANGO. Sets the band1 sampler frequency.

Parameters **value** – band1SamplerFrequency

Returns None

write_band2SamplerFrequency(*value*)
Internal construct of TANGO. Sets the band2 sampler frequency.

Parameters **value** – band2SamplerFrequency

Returns None

write_band3SamplerFrequency(*value*)
Internal construct of TANGO. Sets the band3 sampler frequency.

Parameters **value** – band3SamplerFrequency

Returns None

write_band4SamplerFrequency(*value*)
Internal construct of TANGO. Sets band4 sampler frequency.

Parameters **value** – band4SamplerFrequency

Returns None

write_band5aSamplerFrequency(*value*)
Internal construct of TANGO. Sets the band5a sampler frequency.

Parameters **value** – band5aSamplerFrequency

Returns None

write_band5bSamplerFrequency(*value*)
Internal construct of TANGO. Sets the band5b sampler frequency.

Parameters **value** – band5bSamplerFrequency

Returns None

write_desiredPointing(*value*)
Internal construct of TANGO. Sets the desired pointing coordinates of Dish.

Parameters **value** – desiredPointing

Returns None

write_toggleFault(*value*)
Internal construct of TANGO

`tmcprototype.dishmaster.src.dishmaster.dish_master.`**`main`**`(`*args=None*, ***kwargs*`)`

Runs the DishMaster.

Parameters

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

Returns DishMaster TANGO object.

CHAPTER 7

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.`**`CspMasterLe`**

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 evennts.

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)

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

Return type (ResultCode, str)

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

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.`**`SdpSu`**

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

`Abort``()`

Invoke Abort on SdpSubarrayLeafNode.

class **`AbortCommand`**`(*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 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 invoking command on CSPSubarray.

AssignResources(*argin*)

Assigns resources to given SDP subarray.

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

A class for SdpSubarayLeafNode'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” :{},”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 A tuple containing a return code and a string message indicating status.

The message is for information purpose only.

Return type (ResultCode, str)

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

`Configure`(*argin*)
Invokes Configure on SdpSubarrayLeafNode.

`class ConfigureCommand`(**args*, ***kwargs*)
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 A tuple containing a return code and a string message indicating status. The message is for information purpose only.

Return type (ResultCode, str)

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

EndSB()
This command invokes EndSB command on SDP subarray to end the current Scheduling block.

class EndSBCommand(**args*, ***kwargs*)
A class for SdpSubarrayLeafNode’s EndSB() 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 EndSB command on SDP subarray to end the current Scheduling block.

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.

endsb_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns. Checks whether the endsb 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``()`

Invokes EndScan on SdpSubarrayLeafNode.

`class EndScanCommand``(*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 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.

`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

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.

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

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

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 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

class OnCommand(**args*, ***kwargs*)
A class for SDP Subarray's On() command.

do()

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 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

`ReleaseAllResources`()
Invokes ReleaseAllResources command on SdpSubarrayLeafNode.

`class ReleaseAllResourcesCommand`(**args*, ***kwargs*)
A class for SdpSubarayLeafNode'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 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.

`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

`Restart``()`

Invoke Restart command on SdpSubarrayLeafNode.

`class RestartCommand``(*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 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 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

`Scan`(*argin*)
Invoke Scan command to SDP subarray.

`class ScanCommand`(**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 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.

`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

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_EndSB_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_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=N*
***kwarg*

Runs the SdpSubarrayLeafNode
Parameters

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

Returns SdpSubarrayLeafNode TANGO object

CHAPTER 9

CSP Subarray Leaf Node

CSP Subarray Leaf node is 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 `tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.`**`CspSu`**

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

The message is for information purpose only.

Return type (ResultCode, str)

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 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

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 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

`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 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

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 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 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=N*
***kwarg*

Runs the CspSubarrayLeafNode.

Parameters

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

Returns CspSubarrayLeafNode TANGO object.

CHAPTER 10

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.`**`SdpMasterLe`**

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

The message is for information purpose only.

Return type (ResultCode, str)

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

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)

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*)

CHAPTER 11

MCCS Master Leaf Node

class `tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.`**`MccsMast`**

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], "station_beam_ids": [1], "tile_ids": [1,2,3,4]

}

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 evennts.

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``()`

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

do()

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

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()

`validate_obs_state``()`

`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.

CHAPTER 12

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 `tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.`**`Mc`**

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,"tile_ids":[1,2],},{"station_id":2,"tile_ids":[3,4]},],"station_beam_pointings":

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 mccssubarrayleafnode

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

mccsSubarrayObsState
Used by autodoc_mock_imports.

mccssubarrayHealthState
Used by autodoc_mock_imports.

read_activityMessage()

write_activityMessage(*value*)

tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.**main**(*arg*
***k*

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

tmcprototype.centralnode.src.centralnode.central_node, 1

tmcprototype.centralnodelow.src.centralnodelow.central_node_low, 9

tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node, 39

tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node, 55

tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node, 23

tmcprototype.dishmaster.src.dishmaster.dish_master, 33

tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node, 73

tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node, 67

tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node, 43

tmcprototype.subarraynode.src.subarraynode.subarray_node, 15

tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low, 19

Index

A

Abort() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 55

Abort() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 23

Abort() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 33

Abort() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 43

abort_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode.AbortCommand method*), 55

abort_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AbortCommand method*), 43

achievedPointing (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

activeProcessingBlocks (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode attribute*), 53

activityMessage (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 5

activityMessage (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode attribute*), 12

activityMessage (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode attribute*), 41

activityMessage (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode attribute*), 63

activityMessage (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute*), 29

activityMessage (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode attribute*), 76

activityMessage (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba attribute*), 82

activityMessage (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n attribute*), 70

activityMessage (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray attribute*), 53

activityMessage (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNod attribute*), 15

activityMessage (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S attribute*), 19

add_receptors_ended() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 56

allocate_ended() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 73

always_executed_hook() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 5

always_executed_hook() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 12

always_executed_hook() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n method*), 41

always_executed_hook() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 63

always_executed_hook() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 29

always_executed_hook() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35

always_executed_hook() (*tmcproto-*

type.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method), 76

always_executed_hook() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 82

always_executed_hook() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 70

always_executed_hook() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53

always_executed_hook() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16

always_executed_hook() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

AssignResources() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 1

AssignResources() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 9

AssignResources() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 56

AssignResources() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 73

AssignResources() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 44

AssignResources_ended() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AssignResourcesComman method*), 44

attribute_event_handler() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30

AzElOffset (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 33

azimuth() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35

azimuthOverWrap (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

B

band1SamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

band2SamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

band3SamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

band4SamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

band5aSamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

band5bSamplerFrequency (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

C

calculate_geometric_delays() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 63

calculate_observation_state() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16

calculate_observation_state() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20

capturing (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

CentralAlarmHandler (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 2

CentralAlarmHandler (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra attribute*), 10

CentralNode (*class in tmcprototype.centralnode.src.centralnode.central_node*), 1

CentralNode (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 9

CentralNode.AssignResourcesCommand (*class in tmcprototype.centralnode.src.centralnode.central_node*), 1

CentralNode.AssignResourcesCommand (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 9

CentralNode.InitCommand (*class in tmcprototype.centralnode.src.centralnode.central_node*), 2

CentralNode.InitCommand (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 10

CentralNode.ReleaseResourcesCommand

(*class in tmcprototype.centralnode.src.centralnode.central_node*), 3

CentralNode.ReleaseResourcesCommand (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 10

CentralNode.StandByTelescopeCommand (*class in tmcprototype.centralnode.src.centralnode.central_node*), 4

CentralNode.StandByTelescopeCommand (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 11

CentralNode.StartUpTelescopeCommand (*class in tmcprototype.centralnode.src.centralnode.central_node*), 4

CentralNode.StartUpTelescopeCommand (*class in tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 11

CentralNode.StowAntennasCommand (*class in tmcprototype.centralnode.src.centralnode.central_node*), 5

check_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.AssignResourcesCommand method*), 1

check_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.ReleaseResourcesCommand method*), 3

check_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StandByTelescopeCommand method*), 4

check_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StartUpTelescopeCommand method*), 4

check_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StowAntennasCommand method*), 5

check_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.AssignResourcesCommand method*), 9

check_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.ReleaseResourcesCommand method*), 10

check_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.StandByTelescopeCommand method*), 11

check_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.StartUpTelescopeCommand method*), 11

check_allowed() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n method*), 41

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 55

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 56

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 57

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 58

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 59

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 60

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 61

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 61

check_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 62

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 23

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 23

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 24

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 25

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 25

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 25

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 26

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode. method*), 27 check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.SetStandbyLPModeCommand method*), 26

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.SetStowModeCommand method*), 27

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.SlewCommand method*), 27

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.StartCaptureCommand method*), 28

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.StopCaptureCommand method*), 28

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.StopTrackCommand method*), 28

check_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.TrackCommand method*), 29

check_allowed() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.AssignResourcesCommand method*), 74

check_allowed() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.ReleaseResourcesCommand method*), 75

check_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.ConfigureCommand method*), 79

check_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.EndCommand method*), 80

check_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.EndScanCommand method*), 81

check_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.ScanCommand method*), 82

check_allowed() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.DisableCommand method*), 67

check_allowed() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.StandbyCommand method*), 69

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AbortCommand method*), 44

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AssignResourcesCommand method*), 44

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ConfigureCommand method*), 45

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 46

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 47

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 48

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 50

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 51

check_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 52

check_slew() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35

command_class_object() (*tmcprototype.[illegible] method*), 16

command_class_object() (*tmcprototype.[illegible] method*), 20

Configure() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 57

Configure() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 23

Configure() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 33

Configure() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 79

Configure() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 45

configure_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 57

configure_cmd_ended_cb() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 79

configure_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 45

ConfiguredBand (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 33 convert_radec_to_azel() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30

csp_cbf_health_state_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 41

csp_pss_health_state_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 41

csp_pst_health_state_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 42

cspHealthState (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode attribute*), 41

CspMasterFQDN (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode attribute*), 39

CspMasterLeafNode (*class in tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 39

CspMasterLeafNode.InitCommand (*class in tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 39

CspMasterLeafNode.OffCommand (*class in tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 39

CspMasterLeafNode.OnCommand (*class in tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 40

CspMasterLeafNode.StandbyCommand (*class in tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 40

CspMasterLeafNodeFQDN (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 2

CspSubarrayFQDN (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode attribute*), 58

CspSubarrayFQDN (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode attribute*), 15

CspSubarrayFQDN (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode attribute*), 19

cspsubarrayHealthState (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode attribute*), 63

CspSubarrayLeafNode (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node*), 55

CspSubarrayLeafNode.AbortCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 55

CspSubarrayLeafNode.AssignResourcesCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 56

CspSubarrayLeafNode.ConfigureCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 57

CspSubarrayLeafNode.EndScanCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 58

CspSubarrayLeafNode.GoToIdleCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 59

CspSubarrayLeafNode.InitCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 59

CspSubarrayLeafNode.ObsResetCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 60

CspSubarrayLeafNode.ReleaseAllResourcesCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 60

CspSubarrayLeafNode.RestartCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 61

CspSubarrayLeafNode.StartScanCommand (*class in tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_* 62

CspSubarrayLNFQDN (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNod attribute*), 15

CspSubarrayLNFQDN (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S attribute*), 19

cspsubarrayObsState (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ attribute*), 63

D

decrement_position() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster*

method), 35
delay_model_calculator() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 63
delayModel (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode attribute*), 63
delete_device() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 5
delete_device() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 12
delete_device() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 42
delete_device() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 63
delete_device() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30
delete_device() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35
delete_device() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 76
delete_device() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 82
delete_device() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 70
delete_device() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53
delete_device() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16
delete_device() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20
desiredPointing (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35
Disable() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 67
disable_cmd_ended_cb() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.DisableCommand method*), 67
dishHealthState (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute*), 30
DishLeafNode (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 23
DishLeafNode.AbortCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 23
DishLeafNode.ConfigureCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 23
DishLeafNode.EndScanCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 24
DishLeafNode.InitCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 24
DishLeafNode.ObsResetCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 25
DishLeafNode.RestartCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 25
DishLeafNode.ScanCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 25
DishLeafNode.SetOperateModeCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 26
DishLeafNode.SetStandbyFPModeCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 26
DishLeafNode.SetStandByLPModeCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 26
DishLeafNode.SetStowModeCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 27
DishLeafNode.SlewCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 27
DishLeafNode.StartCaptureCommand (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 28
DishLeafNode.StopCaptureCommand (*class in tmcproto-*

type.dishleafnode.src.dishleafnode.dish_leaf_node), 28

`DishLeafNode.StopTrackCommand` (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 28

`DishLeafNode.TrackCommand` (*class in tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 29

`DishLeafNodePrefix` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 2

`DishLeafNodePrefix` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode attribute*), 15

`DishLeafNodePrefix` (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode attribute*), 19

`DishMaster` (*class in tmcprototype.dishmaster.src.dishmaster.dish_master*), 33

`DishMasterFQDN` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute*), 24

`dishMode` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35

`dishPointingState` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute*), 30

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.AssignResourcesCommand method*), 1

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.InitCommand method*), 3

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.ReleaseResourcesCommand method*), 3

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StandByTelescopeCommand method*), 4

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StartUpTelescopeCommand method*), 5

`do()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode.StowAntennasCommand method*), 5

`do()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.AssignResourcesCommand method*), 9

`do()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.InitCommand method*), 10

`do()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.ReleaseResourcesCommand method*), 10

`do()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.StandByTelescopeCommand method*), 11

`do()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode.StartUpTelescopeCommand method*), 11

`do()` (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode.InitCommand method*), 39

`do()` (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode.OffCommand*

method), 39

`do()` (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_*
method), 40

`do()` (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_*
method), 41

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 56

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 56

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 57

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 58

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 59

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 60

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 60

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 61

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 61

`do()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_sub*
method), 62

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 23

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 24

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 24

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 24

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 25

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 25

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 26

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 26

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 27

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 26

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 27

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 27

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 28

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*
method), 28

`do()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLea*

method), 29
do() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode.TrackCommand method*), 29
do() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.AssignResourcesCommand method*), 74
do() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.InitCommand method*), 74
do() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.OffCommand method*), 74
do() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.OnCommand method*), 75
do() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.ReleaseResourcesCon method*), 76
do() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.ConfigureCo method*), 80
do() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.EndComman method*), 80
do() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.EndScanCom method*), 81
do() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.InitCommand method*), 81
do() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode.ScanComma method*), 82
do() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.DisableCommand method*), 68
do() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.InitCommand method*), 68
do() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.OffCommand method*), 68
do() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.OnCommand method*), 69
do() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.StandbyCommand method*), 70
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AbortCommand method*), 44
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.AssignResourcesC method*), 44
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ConfigureComman method*), 46
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.EndSBCommand method*), 46
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.EndScanCommand method*), 47
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.InitCommand method*), 48
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ObsResetComman method*), 48
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.OffCommand method*), 49
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.OnCommand method*), 49
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ReleaseAllResourc method*), 50
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.RestartCommand*

method), 51
do() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_sub method*), 52
do() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode.AssignResourcesCom method*), 15
do() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_ method*), 19

E

Elevation() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35
End() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mcc method*), 80
end_cmd_ended_cb() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 80
EndSB() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp method*), 46
endsb_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 46
EndScan() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 58
EndScan() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 24
EndScan() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 81
EndScan() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 47
endscan_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 58
endscan_cmd_ended_cb() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 81
endscan_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 47

G

get_deviceproxy() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNod method*), 16
get_deviceproxy() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20
GoToIdle() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 59

gotoidle_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode.GoToIdleCommand method*), 59

H

health_state_cb() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 5

health_state_cb() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 12

health_state_cb() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16

health_state_cb() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

I

increment_position() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 35

init_command_objects() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 5

init_command_objects() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 12

init_command_objects() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 42

init_command_objects() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 63

init_command_objects() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30

init_command_objects() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 76

init_command_objects() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 82

init_command_objects() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 70

init_command_objects() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53

init_command_objects() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16

init_command_objects() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode*

method), 20

init_device() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

is_Abort_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 63

is_Abort_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30

is_Abort_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 53

is_AssignResources_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

is_AssignResources_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 12

is_AssignResources_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 64

is_AssignResources_allowed() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 76

is_AssignResources_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 53

is_Configure_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 64

is_Configure_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30

is_Configure_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 83

is_Configure_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 53

is_Disable_allowed() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_ method*), 71

is_End_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 83

is_EndSB_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 53

is_EndScan_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 64

is_EndScan_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode*

method), 30
is_EndScan_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 83
is_EndScan_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53
is_GoToIdle_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 64
is_ObsReset_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 64
is_ObsReset_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30
is_ObsReset_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53
is_ReleaseAllResources_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 64
is_ReleaseAllResources_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53
is_ReleaseResources_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6
is_ReleaseResources_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 12
is_ReleaseResources_allowed() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 76
is_Restart_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 64
is_Restart_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30
is_Restart_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 53
is_Scan_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 30
is_Scan_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_Scan_allowed() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 83
is_Scan_allowed() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode*

method), 54
is_SetMaintenanceMode_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_SetOperateMode_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_SetOperateMode_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_SetStandbyFPMode_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_SetStandbyFPMode_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_SetStandByLPMode_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_SetStandbyLPMode_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_SetStowMode_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_SetStowMode_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_Slew_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_Standby_allowed() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n method*), 42
is_Standby_allowed() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n method*), 70
is_StandByTelescope_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6
is_StandByTelescope_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 13
is_StartCapture_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_StartCapture_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_StartScan_allowed() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 65
is_StartUpTelescope_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode*

method), 6
is_StartUpTelescope_allowed() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 13
is_StopCapture_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_StopCapture_allowed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36
is_StopTrack_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_StowAntennas_allowed() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6
is_Track_allowed() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
is_Track_allowed() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16
is_Track_allowed() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

M

main() (*in module tmcprototype.centralnode.src.centralnode.central_node*), 7
main() (*in module tmcprototype.centralnodelow.src.centralnodelow.central_node_low*), 13
main() (*in module tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node*), 42
main() (*in module tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node*), 65
main() (*in module tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node*), 32
main() (*in module tmcprototype.dishmaster.src.dishmaster.dish_master*), 37
main() (*in module tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node*), 77
main() (*in module tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node*), 83
main() (*in module tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node*), 71
main() (*in module tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_*), 54
main() (*in module tmcprototype.subarraynode.src.subarraynode.subarray_node*), 17
main() (*in module tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low*), 21
mccsHealthState (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l attribute*), 76
MccsMasterFQDN (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l attribute*), 74
MccsMasterLeafNode (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 73
MccsMasterLeafNode.AssignResourcesCommand (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 73
MccsMasterLeafNode.InitCommand (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 74
MccsMasterLeafNode.OffCommand (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 74
MccsMasterLeafNode.OnCommand (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 75
MccsMasterLeafNode.ReleaseResourcesCommand (*class in tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l*), 75
MCCSMasterLeafNodeFQDN (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra attribute*), 10
MccsSubarrayFQDN (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba attribute*), 81
mccssubarrayHealthState (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba attribute*), 83
MccsSubarrayLeafNode (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba*), 79
MccsSubarrayLeafNode.ConfigureCommand (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba*), 79

MccsSubarrayLeafNode.EndCommand (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node*), 80

MccsSubarrayLeafNode.EndScanCommand (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node*), 81

MccsSubarrayLeafNode.InitCommand (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node*), 81

MccsSubarrayLeafNode.ScanCommand (*class in tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node*), 82

mccsSubarrayObsState (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode attribute*), 83

N

NumDishes (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 3

O

obs_state_cb() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

observation_state_cb() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16

observation_state_cb() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

ObsReset() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 60

ObsReset() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 25

ObsReset() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 33

ObsReset() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 48

obsreset_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode.ObsResetCommand method*), 60

obsreset_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ObsResetCommand method*), 48

off_cmd_ended_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n method*), 39

off_cmd_ended_cb() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 74

off_cmd_ended_cb() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n method*), 68

off_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 49

on_cmd_ended_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_n method*), 40

on_cmd_ended_cb() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 75

on_cmd_ended_cb() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n method*), 69

on_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 49

P

point() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

pointing_state_cb() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16

pointing_state_cb() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20

pointingState (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 36

ProcessingBlockList (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n attribute*), 69

R

read_achievedPointing() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

read_activeProcessingBlocks() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 54

read_activityMessage() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

read_activityMessage() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 13 `read_activityMessage()` (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 42

`read_activityMessage()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 65

`read_activityMessage()` (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31

`read_activityMessage()` (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 76

`read_activityMessage()` (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_subarray_leaf_node.MccsSubarrayLeafNode method*), 83

`read_activityMessage()` (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 71

`read_activityMessage()` (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 54

`read_activityMessage()` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 16

`read_activityMessage()` (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

`read_AzElOffset()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_azimuthOverWrap()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_capturing()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_ConfiguredBand()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_delayModel()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 65

`read_desiredPointing()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_dishMode()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_pointingState()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`read_ProcessingBlockList()` (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 71

`read_receiveAddresses()` (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 54

`read_receptorIDList()` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16

`read_receptorIDList()` (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20

`read_sbID()` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16

`read_sbID()` (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20

`read_scanID()` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16

`read_scanID()` (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 20

`read_subarray1HealthState()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

`read_subarray1HealthState()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 13

`read_subarray2HealthState()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

`read_subarray3HealthState()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

`read_telescopeHealthState()` (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 6

`read_telescopeHealthState()` (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 13

`read_toggleFault()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37

`read_versionInfo()` (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 65

`read_versionInfo()` (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_n method*), 71

`read_WindSpeed()` (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 36

`receive_addresses_cb()` (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNo method*), 16 receive_addresses_cb() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 20

receiveAddresses (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode attribute*), 54

receptorIDList (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode attribute*), 17

receptorIDList (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode attribute*), 21

ReceptorNumber (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 33

ReleaseAllResources() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 60

ReleaseAllResources() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 50

releaseallresources_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode.ReleaseAllResourc method*), 61

releaseallresources_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode.ReleaseAllResourc method*), 50

ReleaseResources() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 3

ReleaseResources() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 10

ReleaseResources() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode method*), 75

releaseresources_cmd_ended_cb() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.MccsMasterLeafNode.ReleaseResourcesComma method*), 76

remove_receptors_from_group() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 17

remove_receptors_from_group() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 21

Restart() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 61

Restart() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 25

Restart() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 33

Restart() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 51

restart_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 62

restart_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 51

S

sbID (*tmcprototype.subarraynode.src.subarraynode.subarray_node.Subar attribute*), 17

sbID (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_ attribute*), 21

Scan() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishL method*), 25

Scan() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34

Scan() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.m method*), 82

Scan() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_ method*), 52

scan_cmd_ended_cb() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 82

scan_cmd_ended_cb() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_ method*), 52

scanID (*tmcprototype.subarraynode.src.subarraynode.subarray_node.Sub attribute*), 17

scanID (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_no attribute*), 21

sdpHealthState (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_ attribute*), 71

SdpMasterFQDN (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_ attribute*), 69

SdpMasterLeafNode (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_*), 67

SdpMasterLeafNode.DisableCommand (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_*), 67

SdpMasterLeafNode.InitCommand (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_*), 68

SdpMasterLeafNode.OffCommand (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_*), 68

SdpMasterLeafNode.OnCommand (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node*), 69

SdpMasterLeafNode.StandbyCommand (*class in tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node*), 69

SdpMasterLeafNodeFQDN (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 4

SdpSubarrayFQDN (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode attribute*), 53

SdpSubarrayFQDN (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode attribute*), 15

SdpSubarrayFQDN (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode attribute*), 19

sdpSubarrayHealthState (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode attribute*), 54

SdpSubarrayLeafNode (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 43

SdpSubarrayLeafNode.AbortCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 43

SdpSubarrayLeafNode.AssignResourcesCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 44

SdpSubarrayLeafNode.ConfigureCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 45

SdpSubarrayLeafNode.EndSBCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 46

SdpSubarrayLeafNode.EndScanCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 47

SdpSubarrayLeafNode.InitCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 48

SdpSubarrayLeafNode.ObsResetCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node*), 48

SdpSubarrayLeafNode.OffCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray*), 49

SdpSubarrayLeafNode.OnCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray*), 49

SdpSubarrayLeafNode.ReleaseAllResourcesCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray*), 50

SdpSubarrayLeafNode.RestartCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray*), 51

SdpSubarrayLeafNode.ScanCommand (*class in tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray*), 52

SdpSubarrayLNFQDN (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNoc attribute*), 15

SdpSubarrayLNFQDN (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S attribute*), 19

sdpSubarrayObsState (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray attribute*), 54

set_dish_name_number() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31

set_observer_lat_long_alt() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31

SetMaintenanceMode() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34

SetOperateMode() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 26

SetOperateMode() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34

SetStandbyFPMode() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 26

SetStandbyFPMode() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34

SetStandByLPMode() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 26

SetStandbyLPMode() (*tmcproto-*

type.dishmaster.src.dishmaster.dish_master.DishMaster method), 34
SetStowMode() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 27
SetStowMode() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
Slew() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 27
Slew() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
Standby() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode method*), 40
Standby() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode method*), 69
standby_cmd_ended_cb() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.CspMasterLeafNode.StandbyCommand method*), 41
standby_cmd_ended_cb() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.SdpMasterLeafNode.StandbyCommand method*), 70
StandByTelescope() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 4
StandByTelescope() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 11
StartCapture() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 28
StartCapture() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
StartScan() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 62
startscan_cmd_ended_cb() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode.StartScanCommand method*), 62
StartUpTelescope() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 4
StartUpTelescope() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode method*), 11
StopCapture() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 28
StopCapture() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
StopTrack() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 28
StopTrack() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
StowAntennas() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 5
subarray1HealthState (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 7
subarray1HealthState (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra attribute*), 13
subarray2HealthState (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 7
subarray3HealthState (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 7
SubarrayNode (*class in tmcprototype.subarraynode.src.subarraynode.subarray_node*), 15
SubarrayNode (*class in tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low*), 19
SubarrayNode.InitCommand (*class in tmcprototype.subarraynode.src.subarraynode.subarray_node*), 15
SubarrayNode.InitCommand (*class in tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low*), 19

T

telescopeHealthState (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 7
telescopeHealthState (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra attribute*), 13
TMAlarmHandler (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 5
TMAlarmHandler (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra attribute*), 12
tmcprototype.centralnode.src.centralnode.central_no (*module*), 1
tmcprototype.centralnodelow.src.centralnodelow.cent (*module*), 9
tmcprototype.cspmasterleafnode.src.cspmasterleafnoc (*module*), 39
tmcprototype.cspsubarrayleafnode.src.cspsubarraylea

(*module*), 55
tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node (*module*), 23
tmcprototype.dishmaster.src.dishmaster.dish_master (*module*), 33
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.sdp_subarray_leaf_node (*module*), 43
tmcprototype.subarraynode.src.subarraynode.subarray_node (*module*), 15
tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low (*module*), 19
TMLowSubarrayNodes (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.CentralNode attribute*), 12
TMMidSubarrayNodes (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode attribute*), 5
toggleFault (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 37
Track() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 29
Track() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 34
Track() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNode method*), 15
Track() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 19
track_slew() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
track_thread() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 31
TrackDuration (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode attribute*), 29
tracking_time_thread() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 32

U

update_config_params() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 65

V

validate_obs_state() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 65
validate_obs_state() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 76
validate_obs_state() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 54
validate_obs_state() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNod method*), 17
validate_obs_state() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.S method*), 21
versionInfo (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ attribute*), 65
versionInfo (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_ attribute*), 71

W

WindSpeed (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster attribute*), 35
write_activityMessage() (*tmcprototype.centralnode.src.centralnode.central_node.CentralNode method*), 7
write_activityMessage() (*tmcprototype.centralnodelow.src.centralnodelow.central_node_low.Centra method*), 13
write_activityMessage() (*tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_ method*), 42
write_activityMessage() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_ method*), 65
write_activityMessage() (*tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.DishLeafNode method*), 32
write_activityMessage() (*tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_l method*), 77
write_activityMessage() (*tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.mccs_suba method*), 83
write_activityMessage() (*tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_ method*), 71
write_activityMessage() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray method*), 54
write_activityMessage() (*tmcprototype.subarraynode.src.subarraynode.subarray_node.SubarrayNod

method), 17
write_activityMessage() (*tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.SubarrayNode method*), 21
write_band1SamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_band2SamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_band3SamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_band4SamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_band5aSamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_band5bSamplerFrequency() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_delayModel() (*tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.CspSubarrayLeafNode method*), 65
write_desiredPointing() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_receiveAddresses() (*tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.SdpSubarrayLeafNode method*), 54
write_toggleFault() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37
write_WindSpeed() (*tmcprototype.dishmaster.src.dishmaster.dish_master.DishMaster method*), 37