TMC Prototype Documentation

Release 1.0

NCRA India

Jan 01, 2020

Contents:

CHAPTER 1

Central Node

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

class tmcprototype.centralnode.src.centralnode.central_node.**CentralNode**(**args*, ***kwargs*)

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

AssignResources(*argin*)

AssignResources command invokes the AssignResources command on lower level devices.

CentralAlarmHandler

Used by autodoc_mock_imports.

CspMasterLeafNodeFQDN

Used by autodoc_mock_imports.

DishLeafNodePrefix

Used by autodoc_mock_imports.

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

A class for the TMC CentralNode's init_device() method.

do()

Initializes the attributes and properties of the Central Node.

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

Return type (ReturnCode, str)

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

NumDishes

Used by autodoc_mock_imports.

ReleaseResources(*argin*)

Release all the resources assigned to the given Subarray.

SdpMasterLeafNodeFQDN
Used by autodoc_mock_imports.

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

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

StowAntennas(*argin*)
This command stows the specified receptors.

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.

init_command_objects()
Initialises the command handlers for commands supported by this device.

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

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

Return type boolean

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

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

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

Return type boolean

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

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

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

Return type boolean

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

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

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

Return type boolean

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

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

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

Return type boolean

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

read_activityMessage()
Internal construct of TANGO. Returns activity message.

read_subarray1HealthState()
Internal construct of TANGO. Returns Subarray1 health state.

read_subarray2HealthState()
Internal construct of TANGO. Returns Subarray2 health state.

read_subarray3HealthState()
Internal construct of TANGO. Returns Subarray3 health state.

read_telescopeHealthState()
Internal construct of TANGO. Returns the Telescope health state.

subarray1HealthState
Used by autodoc_mock_imports.

subarray2HealthState
Used by autodoc_mock_imports.

subarray3HealthState
Used by autodoc_mock_imports.

telescopeHealthState
Used by autodoc_mock_imports.

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activity message.

tmcprototype.centralnode.src.centralnode.central_node.**main**(*args=None*, ***kwargs*)
Runs the CentralNode. :param args: Arguments internal to TANGO

Parameters **kwargs** – Arguments internal to TANGO

Returns CentralNode TANGO object.

tmcprototype.centralnode.src.centralnode.central_node.**AssignResources**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**DeviceData**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**ObsStateAggregator**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**ReleaseResources**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**StandByTelescope**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**StartUpTelescope**
Used by autodoc_mock_imports.

tmcprototype.centralnode.src.centralnode.central_node.**StowAntennas**
Used by autodoc_mock_imports.

CHAPTER 2

Central Node

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

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

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

AssignResources(*argin*)

AssignResources command invokes the AssignResources command on lower level devices.

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

A class for CentralNode's AssignResources() command.

check_allowed()

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

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

Return type boolean

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

do(*argin*)

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

Parameters **argin** – The string in JSON format. The JSON contains following values:
subarray_id: DevShort. Mandatory. Sub-Array to allocate resources to
station_ids: DevArray. Mandatory list of stations contributing beams to the data set
channels: DevArray. Mandatory list of frequency channels used
station_beam_ids: DevArray. Mandatory logical ID of beam

Example:

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

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

Returns None

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

Note: Enter input without spaces as:{“subarray_id”:1,”station_ids”:[1,2],”channels”:[1,2,3,4,5,6,7,8],”station_beam_ids

CentralAlarmHandler
Used by autodoc_mock_imports.

class InitCommand(**args*, ***kwargs*)
A class for the TMC CentralNode’s init_device() method.

do()
Initializes the attributes and properties of the Central Node Low.
Returns A tuple containing a return code and a string message indicating status. The message is for information purpose only.
Return type (ReturnCode, str)
Raises DevFailed if error occurs while initializing the CentralNode device or if error occurs while creating device proxy for any of the devices like SubarrayNodeLow or MccsMasterLeafNode.

MCCSMasterLeafNodeFQDN
Used by autodoc_mock_imports.

ReleaseResources(*argin*)
Release all the resources assigned to the given Subarray.

class ReleaseResourcesCommand(**args*, ***kwargs*)
A class for CentralNode’s ReleaseResources() command.

check_allowed()
Checks whether this command is allowed to be run in current device state
Returns True if this command is allowed to be run in current device state
Return type boolean
Raises DevFailed if this command is not allowed to be run in current device state

do(*argin*)
Release all the resources assigned to the given Subarray. It accepts the subarray id, release_all flag in JSON string format. When the release_all flag is True, ReleaseAllResources command is invoked on the respective SubarrayNode.
Parameters **argin** – The string in JSON format. The JSON contains following values:
subarray_id: DevShort. Mandatory.
release_all: Boolean(True or False). Mandatory. True when all the resources to be released from Subarray.
Example:
{ “subarray_id”: 1, “release_all”: true,
}
Note: From Jive, enter input as: {“subarray_id”:1,”release_all”:true} without any space.

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

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

class StandByTelescopeCommand(**args*, ***kwargs*)
A class for Low CentralNode’s StandByTelescope() command.

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

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

Return type boolean

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

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

param argin: None.

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

The message is for information purpose only.

Return type (ResultCode, str)

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

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

class StartUpTelescopeCommand(**args*, ***kwargs*)
A class for Low CentralNode's StartupCommand() command.

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

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

Return type boolean

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

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

Parameters **argin** – None.

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

The message is for information purpose only.

Return type (ResultCode, str)

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

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.

OnCommand class for SubarrayNode

class tmcprototype.subarraynode.src.subarraynode.on_command.**OnCommand**(**args*, ***kwargs*)
A class for the SubarrayNode's On() command.

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

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

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful

OffCommand class for SubarrayNode

class tmcprototype.subarraynode.src.subarraynode.off_command.**OffCommand**(**args*, ***kwargs*)
A class for the SubarrayNodes's Off() command.

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

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

The message is for information purpose only.

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful

AssignResourcesCommand class for SubarrayNode.

class tmcprototype.subarraynode.src.subarraynode.assign_resources_command.**AssignResourcesC**

A class for SubarrayNode's AssignResources() command.

add_receptors_in_group(*argin*)

Creates a tango group of the successfully allocated resources in the subarray. Device proxy for each of the resources is created. The healthState and pointintgState attributes from all the devices in the group are subscribed so that the changes in the respective device are received at Subarray Node.

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

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

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

assign_csp_resources(*argin*)

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

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

Example: ['0001', '0002']

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

assign_sdp_resources(*argin*)

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

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

Returns

List of strings.

Example: ['PB1', 'PB2']

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

do(*argin*)

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

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

Parameters **argin** – DevString.

Example:

{“dish”:{“receptorIDList”:[“0002”,”0001”]},”sdp”:{“id”: “sbi-mvp01-20200325-00001”,”max_length”:100.0,”scan_types”:[{“id”:”science_A”, “coordinate_system”:”ICRS”,”ra”:”02:42:40.771”,”dec”:”-00:00:47.84”,”channels”: [{“count”:744,”start”:0,”stride”:2,”freq_min”:0.35e9,”freq_max”:0.368e9, “link_map”:[[0,0],[200,1],[744,2],[944,3]]},{“count”:744,”start”:2000,”stride”:1, “freq_min”:0.36e9,”freq_max”:0.368e9,”link_map”:[[2000,4],[2200,5]]}]},{“id”: “calibration_B”,”coordinate_system”:”ICRS”,”ra”:”12:29:06.699”,”dec”:”02:03:08.598”, “channels”:[{“count”:744,”start”:0,”stride”:2,”freq_min”:0.35e9, “freq_max”:0.368e9,”link_map”:[[0,0],[200,1],[744,2],[944,3]] “start”:2000,”stride”:1,”freq_min”:0.36e9,”freq_max”:0.368e9,”link_map”:[[2000,4], [2200,5]]}]}],”processing_blocks”:[{“id”:”pb-mvp01-20200325-00001”,”workflow”: {“type”:”realtime”,”id”:”vis_receive”,”version”:”0.1.0”},”parameters”:{}}, {“id”:”pb-mvp01-20200325-00002”,”workflow”:{“type”:”realtime”,”id”:”test_realtime”, “version”:”0.1.0”},”parameters”:{}},{“id”:”pb-mvp01-20200325-00003”,”workflow”: {“type”:”batch”,”id”:”ical”,”version”:”0.1.0”},”parameters”:{},”dependencies”:[{“pb_id”:”pb-mvp01-20200325-00001”,”type”:[“visibilities”]}]},{“id”: “pb-mvp01-20200325-00004”,”workflow”:{“type”:”batch”,”id”:”dpreb”,”version”:”0.1.0”}, “parameters”:{},”dependencies”:[{“pb_id”:”pb-mvp01-20200325-00003”,”type”: [“calibration”]}]}]}}

Returns

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

[“0001”,”0002”]

as argout if allocation successful.

Return type (ResultCode, str)

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

ReleaseAllResourcesCommand for SubarrayNode

class tmcprototype.subarraynode.src.subarraynode.release_all_resources_command.**ReleaseAllRe**

A class for SKASubarray’s ReleaseAllResources() command.

do()

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

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

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

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful

release_csp_resources()

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

Parameters **argin** – DevVoid

Returns DevVoid

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

Parameters **argin** – DevVoid

Returns DevVoid

ConfigureCommand class for SubarrayNode.

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

A class for SubarrayNode's Configure() command.

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

Parameters **argin** – DevString.

JSON string that includes pointing parameters of Dish - Azimuth and Elevation Angle, CSP Configuration and SDP Configuration parameters. JSON string example is: {"pointing":{"target":{"system":"ICRS","name":"Polaris Australis","RA":"21:08:47.92","dec":"-88:57:22.9"}}, "dish":{"receiverBand":"1"},"csp":{"id":"sbi-mvp01-20200325-00001-science_A","frequencyBand":"1", "fsp":[{"fspID":1,"functionMode":"CORR","frequencySliceID":1,"integrationTime":14 "sdp":{"scan_type":"science_A"},"tmc":{"scanDuration":10.0}} CSP block in json string is as per earlier implementation and not aligned to SP-872 Note: While invoking this command from JIVE, provide above JSON string without any space.

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

Return type (ReturnCode, str)

Raises JSONDecodeError if input argument json string contains invalid value

class tmcprototype.subarraynode.src.subarraynode.configure_command.**ElementDeviceData**

static build_up_csp_cmd_data(*scan_config*, *attr_name_map*, *receive_addresses_map*)
Here the input data for CSP is build which is used in configuration of CSP. Below is the csp_config_schema variable value generated by using ska_telmodel library. {'id': 'sbi-mvp01-20200325-00001-science_A', 'frequencyBand': '1', 'fsp': [{'fspID': 1, 'functionMode' : 'CORR', 'frequencySliceID': 1, 'integrationTime': 1400, 'corrBandwidth': 0, 'channelAveragingMap': [[0, 2], [744, 0]], 'fspChannelOffset': 0, 'outputLinkMap': [[0, 0], [200, 1]], 'outputHost': [[0, '192.168.0.1'], [400, '192.168.0.2']], 'outputMac': [[0, '06-00-00-00-00-00']], 'outputPort': [[0, 9000, 1], [400, 9000, 1]]}, {'fspID': 2, 'functionMode': 'CORR', 'frequencySliceID': 2, 'integrationTime': 1400, 'corrBandwidth': 0, 'channelAveragingMap': [[0, 2], [744, 0]], 'fspChannelOffset': 744, 'outputLinkMap': [[0, 4], [200, 5]], 'outputHost': [[0, '192.168.0.3'], [400, '192.168.0.4']], 'outputMac': [[0, '06-00-00-00-00-01']], 'outputPort': [[0, 9000, 1], [400, 9000, 1]]}]}

Returns csp confiuration schema

static build_up_dsh_cmd_data(*scan_config*, *only_dishconfig_flag*)

static build_up_sdp_cmd_data(*scan_config*)

ScanCommand class for SubarrayNode

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

A class for SubarrayNode's Scan() command.

call_end_scan_command()

do(*argin*)

This command accepts id as input. And it Schedule scan on subarray from where scan command is invoked on respective CSP and SDP subarray node for the provided interval of time. It checks whether the scan is already in progress. If yes it throws error showing duplication of command.

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

JSON string example as follows:

{"id": 1}

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

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if the command execution is not successful

EndScanCommand class for SubarrayNode.

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

A class for SubarrayNode's EndScan() command.

do()

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

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if the command execution is not successful.

EndCommand class for SubarrayNode.

class tmcprototype.subarraynode.src.subarraynode.end_command.**EndCommand**(**args*, ***kwargs*)

A class for SubarrayNode's End() command.

do()

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

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

The message is for information purpose only.

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful.

stop_dish_tracking()

AbortCommand for SubarrayNode.

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

A class for SubarrayNode's Abort() command.

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

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

Return type (ResultCode, str)

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

RestartCommand for SubarrayNode.

class tmcprototype.subarraynode.src.subarraynode.restart_command.**RestartCommand**(**args*, ***kwargs*)
A class for SubarrayNode's Restart() command.

do()
This command invokes Restart command on CSPSubarrayLeafNode, SDpSubarrayLeafNode and DishLeafNode.

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

Return type (ResultCode, str)

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

ObsResetCommand for SubarrayNode.

class tmcprototype.subarraynode.src.subarraynode.obsreset_command.**ObsResetCommand**(**args*, ***kwargs*)
A class for SubarrayNode's ObsReset() command.

do()
This command invokes ObsReset command on CspSubarrayLeafNode, SdpSubarrayLeafNode and DishLeafNode.

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

Return type (ResultCode, str)

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

TrackCommand class for SubarrayNode.

class tmcprototype.subarraynode.src.subarraynode.track_command.**TrackCommand**(**args*, ***kwargs*)
A class for SubarrayNode's Track command.

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

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

Return type boolean

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

do(*argin*)
Invokes Track command on the Dishes assigned to the Subarray.

Parameters **argin** – DevString

Example: radec|21:08:47.92|-88:57:22.9 as argin Argin to be provided is the Ra and Dec values where first value is tag that is radec, second value is Ra in Hr:Min:Sec, and third value is Dec in Deg:Min:Sec.

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

The message is for information purpose only.

Return type (ResultCode, str)

CHAPTER 4

Subarray Node

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

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

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

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

A class for the TMC SubarrayNode's init_device() method.

do()

Initializes the attributes and properties of the Subarray Node.

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if the error while subscribing the tango attribute

MccsSubarrayFQDN

Used by autodoc_mock_imports.

MccsSubarrayLNFQDN

Used by autodoc_mock_imports.

activityMessage

Used by autodoc_mock_imports.

always_executed_hook()

Internal construct of TANGO.

calculate_observation_state()

Calculates aggregated observation state of Subarray.

command_class_object()

Sets up the command objects :return: None

delete_device()
Internal construct of TANGO.

get_deviceproxy(*device_fqdn*)
Returns device proxy for given FQDN.

health_state_cb(*event*)
Receives the subscribed health states, aggregates them to calculate the overall subarray health state.

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

Type Event object It has the following members:

- date (event timestamp)
- reception_date (event reception timestamp)
- type (event type)
- dev_name (device name)
- name (attribute name)
- value (event value)

Returns None

init_command_objects()
Initialises the command handlers for commands supported by this device.

observation_state_cb(*evt*)
Receives the subscribed MCCS Subarray obsState.

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

Type Event object It has the following members:

- date (event timestamp)
- reception_date (event reception timestamp)
- type (event type)
- dev_name (device name)
- name (attribute name)
- value (event value)

Returns None

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

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

read_scanID()
Internal construct of TANGO. Returns the Scan ID.

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

scanID
Used by autodoc_mock_imports.

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activityMessage.

tmcprototype.subarraynodelow.src.subarraynodelow.subarray_node_low.**main**(*args=None*, ***kwargs*)

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

AssignResourcesCommand class for SubarrayNodeLow.

class tmcprototype.subarraynodelow.src.subarraynodelow.assign_resources_command.**AssignResou**

A class for SubarrayNodelow's AssignResources() command.

do(*argin*)

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

Parameters **argin** – DevString in JSON form containing following fields: station_ids: list of integers

channels: list of integers

station_beam_ids: list of integers

Example:

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

Returns A tuple containing ResultCode and string.

ConfigureCommand class for SubarrayNodeLow.

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

A class for SubarrayNodeLow's Configure() command.

do(*argin*)

Configures the resources assigned to the Mccs Subarray.

Parameters **argin** – DevString.

JSON string example is:

{"mccs":{"stations":[{"station_id":1,},{"station_id":2,}],"station_beam_pointings": [{"station_beam_id":1,"target":{"system":"HORIZON","name":"DriftScan","Az":180.0,"El":45.0}, "update_rate":0.0,"channels":[1,2,3,4,5,6,7,8]}]},"tmc":{"scanDuration":10.0}}

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

Return type (ReturnCode, str)

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

EndCommand class for SubarrayNodeLow.

class tmcprototype.subarraynodelow.src.subarraynodelow.end_command.**EndCommand**(**args*, ***kwargs*)

A class for SubarrayNodeLow's End() command.

do()

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

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

The message is for information purpose only.

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful.

EndScanCommand class for SubarrayNodeLow.

class tmcprototype.subarraynodelow.src.subarraynodelow.end_scan_command.**EndScanCommand**(**args*, ***kw*

A class for SubarrayNodeLow's EndScan() command.

do()

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

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if the command execution is not successful.

OffCommand class for SubarrayNodeLow

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

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

do()

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

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

The message is for information purpose only.

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful

OnCommand class for SubarrayNodeLow

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

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

do()

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

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

Return type (ResultCode, str)

Raises DevFailed if the command execution is not successful

ReleaseAllResourcesCommand for SubarrayNodeLow

class tmcprototype.subarraynodelow.src.subarraynodelow.release_all_resources_command.**Releas**

A class for SKASubarrayLow's ReleaseAllResources() command.

do()

It invokes ReleaseAllResources command on Subarraylow.

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

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

Return type (ResultCode, str)

ScanCommand class for SubarrayNodeLow

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

A class for SubarrayNodeLow's Scan() command.

call_end_scan_command()

do(*argin*)

This command accepts id as input. And it Schedule scan on subarray from where scan command is invoked on MCCS subarray Leaf Node for the provided interval of time. It checks whether the scan is already in progress. If yes it throws error showing duplication of command.

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

JSON string example as follows:

{"id": 1}

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

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if the command execution is not successful

CHAPTER 5

Dish Leaf Node

A Leaf control node for DishMaster.

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

A Leaf control node for DishMaster.

Abort()

Invokes Abort command on the DishMaster.

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

A class for DishLeafNode's Abort command.

check_allowed()

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

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

Return type boolean

do()

Invokes TrackStop command on the DishMaster.

Raises DevFailed – If error occurs while invoking TrackStop command on 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

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

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

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

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

DishMasterFQDN
Used by autodoc_mock_imports.

EndScan(*argin*)
Invokes StopCapture command on DishMaster.

class EndScanCommand(**args*, ***kwargs*)
A class for DishLeafNode's EndScan() command.

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

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

Return type boolean

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

Parameters **argin** – timestamp

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

class InitCommand(**args*, ***kwargs*)
A class for the TMC DishLeafNode's init_device() method.

do()
Initializes the attributes and properties of the DishLeafNode.

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

Return type (ResultCode, str)

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

ObsReset()
Invokes ObsReset command on the DishLeafNode.

class ObsResetCommand(**args*, ***kwargs*)
A class for DishLeafNode's ObsReset command.

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

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

Return type boolean

do()
Invokes SetStandbyFPMode command on the DishMaster.

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

Restart()
Invokes Restart command on the DishMaster.

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

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

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

Return type boolean

do()
Invokes StopCapture command on the DishMaster.

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

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

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

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

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

Return type boolean

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

Parameters **argin** – timestamp

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

SetOperateMode()
Invokes SetOperateMode command on DishMaster.

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

do()
Invokes SetOperateMode command on DishMaster.

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

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

class SetStandbyFPModeCommand(**args*, ***kwargs*)
A class for DishLeafNode's SetStandbyFPMode() command.

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

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

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

class SetStandbyLPModeCommand(**args*, ***kwargs*)
A class for DishLeafNode's SetStandbyLPMode() command.

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

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

SetStowMode()
Invokes SetStowMode command on DishMaster.

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

do()
Invokes SetStowMode command on DishMaster.

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

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

class SlewCommand(**args*, ***kwargs*)
A class for DishLeafNode's SlewCommand() command.

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

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

Return type boolean

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

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

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

StartCapture(*argin*)
Triggers the DishMaster to Start capture on the set configured band.

class StartCaptureCommand(**args*, ***kwargs*)
A class for DishLeafNode's StartCapture() command.

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

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

Return type boolean

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

Parameters **argin** – timestamp

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

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

class StopCaptureCommand(**args*, ***kwargs*)
A class for DishLeafNode's StopCapture() command.

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

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

Return type boolean

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

Parameters **argin** – timestamp

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

StopTrack()
Invokes StopTrack command on the DishMaster.

class StopTrackCommand(**args*, ***kwargs*)
A class for DishLeafNode's StopTrack() command.

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

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

Return type boolean

do()
Invokes TrackStop command on the DishMaster.

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

Track(*argin*)
Invokes Track command on the DishMaster.

class TrackCommand(**args*, ***kwargs*)
A class for DishLeafNode's Track() command.

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

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

Return type boolean

do(*argin*)
Invokes Track command on the DishMaster.

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

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

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

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

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

activityMessage
Used by autodoc_mock_imports.

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

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

convert_radec_to_azel(*target*, *timestamp*)
Converts RaDec coordinate in to AzEl coordinate using KATPoint library.

Parameters

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

Returns list Azimuth and elevation angle, in degrees

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

dishHealthState
Used by autodoc_mock_imports.

dishPointingState
Used by autodoc_mock_imports.

init_command_objects()
Initialises the command handlers for commands supported by this device.

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

read_activityMessage()
Returns the activityMessage

set_dish_name_number()

set_observer_lat_long_alt()

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

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activityMessage

tmcprototype.dishleafnode.src.dishleafnode.dish_leaf_node.**main**(*args=None*, ***kwargs*)
Runs the DishLeafNode. :param args: Arguments internal to TANGO :param kwargs: Arguments internal to TANGO :return: DishLeafNode TANGO object.

CHAPTER 6

Dish Master

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)

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

on_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns. Checks whether the On command has been successfully invoked on CSPMaster.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

Standby(*argin*)
Sets Standby Mode on the CSP Element.

class StandbyCommand(**args*, ***kwargs*)
A class for CspMasterLeafNode's Standby() command.

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

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

Return type boolean

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

do(*argin*)
It invokes the STANDBY command on CSP Master.

Parameters **argin** – DevStringArray.

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

Returns None

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

standby_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns. Checks whether the StandBy command has been successfully invoked on CSPMaster.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

activityMessage
Used by autodoc_mock_imports.

always_executed_hook()
Internal construct of TANGO.

cspHealthState
Used by autodoc_mock_imports.

csp_cbf_health_state_cb(*evt*)
Retrieves the subscribed cspCbfHealthState attribute of CSPMaster.

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

Returns None

csp_pss_health_state_cb(*evt*)
Retrieves the subscribed cspPssHealthState attribute of CSPMaster.

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

Returns None

csp_pst_health_state_cb(*evt*)
Retrieves the subscribed cspPstHealthState attribute of CSPMaster.

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

Returns None

delete_device()
Internal construct of TANGO.

init_command_objects()
Initialises the command handlers for commands supported by this device.

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

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

Return type boolean

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

read_activityMessage()
Internal construct of TANGO. Returns the activityMessage.

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activityMessage.

tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.**main**(*args=None*, ***kwargs*)
Runs the CspMasterLeafNode.

Parameters

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

Returns CspMasterLeafNode TANGO object.

CHAPTER 8

SDP Subarray Leaf Node

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 None

Raises DevFailed if error occurs while invoking command on CSPSubarray.

AssignResources(*argin*)
Invokes AssignResources command on CspSubarrayLeafNode.

class AssignResourcesCommand(**args*, ***kwargs*)
A class for CspSubarrayLeafNode's AssignResources() command.

add_receptors_ended(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

Type CmdDoneEvent object It has the following members:

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

Returns none

Raises DevFailed if this command is not allowed to be run

in current device state

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

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

Return type boolean

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

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

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

dish: Mandatory JSON object consisting of

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

Example: {

"dish": {

“receptorIDList”: [“0001”, “0002”

]

}

}

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

Returns None

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

Configure(*argin*)
Invokes Configure command on CspSubarrayLeafNode

class ConfigureCommand(**args*, ***kwargs*)
A class for CspSubarrayLeafNode’s Configure() command.

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

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

Return type boolean

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

configure_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

do(*argin*)
This command configures a scan. It accepts configuration information in JSON string format and invokes Configure command on CspSubarray.

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

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

"fspChannelOffset":744,"outputLinkMap":[[0,4],[200,5]],"outputHost":
[[0,"192.168.1.1"]], "outputPort":[[0,9744,1]]}],"delayModelSubscriptionPoint":

"ska_mid/tm_leaf_node/csp_subarray01/delayModel","pointing":{"target":{"system":"ICRS",
"name":"Polaris Australis","RA":"21:08:47.92","dec":"-88:57:22.9"}}}

Note: Enter the json string without spaces as 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 None

Raises DevFailed if the command execution is not successful

gotoidle_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

class InitCommand(**args*, ***kwargs*)
A class for the CspSubarrayLeafNode's init_device() method"

do()
Initializes the attributes and properties of the CspSubarrayLeafNode.

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

for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if error occurs in creating proxy for CSPSubarray.

ObsReset()
Invokes ObsReset command on cspsubarrayleafnode

class ObsResetCommand(**args*, ***kwargs*)
A class for CSPSubarrayLeafNode's ObsReset() command.

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

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

Return type boolean

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

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

Parameters **argin** – None

Returns None

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

obsreset_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

ReleaseAllResources()
Invokes ReleaseAllResources command on CspSubarrayLeafNode

class ReleaseAllResourcesCommand(**args*, ***kwargs*)
A class for CspSubarrayLeafNode's ReleaseAllResources() command.

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

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

Return type boolean

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

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

Returns None

Raises DevFailed if the command execution is not successful

releaseallresources_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

Restart()
Invokes Restart command on cspsubarrayleafnode

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

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

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

Return type boolean

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

do()
This command invokes Restart command on CSPSubarray.

Returns None

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

restart_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

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

Returns none

StartScan(*argin*)
Invokes StartScan command on cspsubarrayleafnode

class StartScanCommand(**args*, ***kwargs*)
A class for CspSubarrayLeafNode's StartScan() command.

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

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

Return type boolean

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

do(*argin*)
This command invokes Scan command on CspSubarray. It is allowed only when CspSubarray is in ObsState READY.

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

Example: {"id":1}

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

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

Return type (ReturnCode, str)

Raises DevFailed if the command execution is not successful

startscan_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

activityMessage
Used by autodoc_mock_imports.

always_executed_hook()
Internal construct of TANGO.

calculate_geometric_delays(*time_t0*)
This method calculates geometric delay values (in Second) using KATPoint library. It requires delay correction object, timestamp t0 and target RaDec. Numpy library is used to convert delay values (in Seconds) to fifth order polynomial coefficients. Six timestamps from the time-frame t0 to t+10, are used to calculate delays per antenna. These six delay values are then used to calculate fifth order polynomial coefficients. In order to calculate delays in advance, timestamp t0 is considered to be one minute ahead of the the current timestamp.

Parameters **argin** – time_t0

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

cspSubarrayObsState
Used by autodoc_mock_imports.

cspsubarrayHealthState
Used by autodoc_mock_imports.

delayModel
Used by autodoc_mock_imports.

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

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

Returns None.

delete_device()
Internal construct of TANGO.

init_command_objects()
Initialises the command handlers for commands supported by this device.

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

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

Return type boolean

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

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

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

Return type boolean

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

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

Returns True if this command is allowed to be run in

current device state

Return type boolean

Raises DevFailed if this command is not allowed to be run

in current device state

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

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

Return type boolean

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

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

Returns True if this command is allowed to be run in

current device state

Return type boolean

Raises DevFailed if this command is not allowed to be run

in current device state

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

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

Return type boolean

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

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

Returns True if this command is allowed to be run in

current device state

Return type boolean

Raises DevFailed if this command is not allowed to be run

in current device state

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

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

Return type boolean

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

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

Returns True if this command is allowed to be run in

current device state

Return type boolean

Raises DevFailed if this command is not allowed to be run

in current device state

read_activityMessage()
Internal construct of TANGO. Returns activity message.

read_delayModel()
Internal construct of TANGO. Returns the delay model.

read_versionInfo()
Internal construct of TANGO. Returns the version information.

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

Parameters **argin** – None

Returns None

validate_obs_state()

versionInfo
Used by autodoc_mock_imports.

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activity message.

write_delayModel(*value*)
Internal construct of TANGO. Sets in to the delay model.

tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.csp_subarray_leaf_node.**main**(*args=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 None

class InitCommand(**args*, ***kwargs*)
A class for the SDP master's init_device() method"

do()
Initializes the attributes and properties of the SdpMasterLeafNode.

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

The message is for information purpose only.

Return type (ReturnCode, str)

Raises

class OffCommand(**args*, ***kwargs*)
A class for SDP master's Off() command.

do()
Sets the OperatingState to Off.

Parameters **argin** – None.

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

The message is for information purpose only.

Return type (ResultCode, str)

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

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

Type CmdDoneEvent object It has the following members:

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

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

Returns none

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

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

Parameters **argin** – None.

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

The message is for information purpose only.

Return type (ResultCode, str)

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

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

Type CmdDoneEvent object It has the following members:

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

Returns none

ProcessingBlockList
Used by autodoc_mock_imports.

SdpMasterFQDN
Used by autodoc_mock_imports.

Standby()
Invokes Standby command .

Parameters **argin** – None

Returns None

class StandbyCommand(**args*, ***kwargs*)
A class for SDP Master's Standby() command.

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

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

rtype boolean

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

do()
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 None

is_Standby_allowed()
Checks Whether this command is allowed to be run in current device state.

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

Return type boolean

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

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

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

Type CmdDoneEvent object It has the following members:

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

Returns none

activityMessage
Used by autodoc_mock_imports.

always_executed_hook()
Internal construct of TANGO.

delete_device()
Internal construct of TANGO.

init_command_objects()
Initialises the command handlers for commands supported by this device.

is_Disable_allowed()
Checks Whether this command is allowed to be run in current device state.

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

Return type boolean

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

read_ProcessingBlockList()
Internal construct of TANGO. :return:

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

read_versionInfo()
Internal construct of TANGO. Version information of TANGO device.

sdpHealthState
Used by autodoc_mock_imports.

versionInfo
Used by autodoc_mock_imports.

write_activityMessage(*value*)
Internal construct of TANGO. Sets the activity message.

tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.sdp_master_leaf_node.**main**(*args=None*, ***kwargs*)

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]

}

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(*argin*)
Invokes ReleaseResources command on MccsMasterLeafNode

class ReleaseResourcesCommand(**args*, ***kwargs*)
A class for MccsMasterLeafNode's ReleaseResources() command.

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

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

Return type boolean

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

do(*argin*)

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

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

Example:

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

}

Returns None.

Raises DevFailed if the command execution is not successful

releaseresources_cmd_ended_cb(*event*)

Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

activityMessage

Used by autodoc_mock_imports.

always_executed_hook()

Internal construct of TANGO.

delete_device()

Internal construct of TANGO.

init_command_objects()

Initialises the command handlers for commands supported by this device.

is_AssignResources_allowed()

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

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

Return type boolean

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

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

Return type boolean

mccsHealthState
Used by autodoc_mock_imports.

read_activityMessage()

write_activityMessage(*value*)

tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.mccs_master_leaf_node.**main**(*args=None*, ***kwargs*)
Runs the MccsMasterLeafNode.

Parameters

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

Returns An object of CompletedProcess class returned by the subprocess.

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},{"station_id":2}],"station_beam_pointings":[{"station_beam_id":1,"target":{

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

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

Return type (ReturnCode, str)

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

End()

Invokes End command on MccsSubarrayLeafNode.

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

A class for MccsSubarrayLeafNode's End() command.

check_allowed()

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

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

Return type boolean

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

do()

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

Returns None

Return type Void

Raises DevFailed if the command execution is not successful

end_cmd_ended_cb(*event*)

Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

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

Returns none

EndScan()
Invokes EndScan command on MccsSubarray.

class EndScanCommand(**args*, ***kwargs*)
A class for MccsSubarrayLeafNode's EndScan() command.

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

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

Return type boolean

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

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

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

endscan_cmd_ended_cb(*event*)
Callback function immediately executed when the asynchronous invoked command returns.

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

Type CmdDoneEvent object It has the following members:

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

Returns none

class InitCommand(**args*, ***kwargs*)
A class for the MccsSubarrayLeafNode's init_device() method"

do()
Initializes the attributes and properties of the MccsSubarrayLeafNode.

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

for information purpose only.

Return type (ReturnCode, str)

Raises DevFailed if error occurs in creating proxy for MCCSSubarray.

MccsSubarrayFQDN
Used by autodoc_mock_imports.

Scan(*argin*)
Invokes Scan command on 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

mccsSubarrayHealthState
Used by autodoc_mock_imports.

mccsSubarrayObsState
Used by autodoc_mock_imports.

read_activityMessage()

write_activityMessage(*value*)

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

CHAPTER 13

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

A

D

E

G

H

I

M

N

O

P

R

S

T

U

V

W