TMC Prototype Documentation

Release 1.0

NCRA India

Feb 19, 2021

Contents:

CHAPTER 1

Central Node

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

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

Device Properties

CentralAlarmHandler: Device name of CentralAlarmHandler

TMAlarmHandler: Device name of TMAlarmHandler

TMLowSubarrayNodes: List of TM Low Subarray Node devices

MCCSMasterLeafNodeFQDN: FQDN of Mccs Master Leaf Node.

Device Attributes

telescopeHealthState: Health state of Telescope

subarray1HealthState: Health state of SubarrayNode1

activityMessage: String providing information about the current activity in Central Node.

class tmcprototype.centralnodelow.src.centralnodelow.assign_resources_command.**AssignResourc**

A class for CentralNode's AssignResources() command.

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

do(*argin*)

Method to invoke AssignResources command on Subarray.

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: {“mccs”:{“subarray_id”:1,”station_ids”:[1,2],”channels”:[[0,8,1,1],[8,8,2,1],[24,16,2,1]],”station_beam_ids”:[1]

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

return: None

raises: KeyError if input argument json string contains invalid key

ValueError if input argument json string contains invalid value

AssertionError if Mccs On command is not completed.

class tmcprototype.centralnodelow.src.centralnodelow.release_resources_command.**ReleaseResou**

A class for CentralNodeLow’s ReleaseResources() command.

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.

do(*argin*)

Method to invoke ReleaseResources command on Subarray Node.

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: {“mccs”:{“subarray_id”:1,”release_all”:true}}

Note: From Jive, enter input as: {“mccs”:{“subarray_id”:1,”release_all”:true}} without any space.

return: 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 or command invocation on SubarrayNode is not successful

class tmcprototype.centralnodelow.src.centralnodelow.standby_telescope_command.**StandByTeles**

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

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

do()

Method to invoke StandBy command.

param argin: None.

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

rtype: (ResultCode, str)

raises: AssertionError if Mccs On command is not completed.

class tmcprototype.centralnodelow.src.centralnodelow.startup_telescope_command.**StartUpTeles**

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

StartUpTelescope command on Central Node Low enables the telescope to perform further operations and observations. It Invokes On command on lower level devices.

do ( )

Method to invoke ON command On lower level devices.

param argin: None.

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

rtype: (ResultCode, str)

raises: AssertionError if f Mccs Off command is not completed.

CHAPTER 2

Subarray Node Low

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.

Device Properties

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

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

Device Attributes

scanID: ID of ongoing SCAN

activityMessage: String providing information about the current activity in SubarrayNode.

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

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

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

do()

Method to invoke On command.

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

rtype: (ResultCode, str)

raises: DevFailed if the command execution is not successful

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

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

This command invokes Off Command on MCCS Subarray through Mccs Subarray leaf node. This comamnd changes Subaray device state from ON to OFF.

do()
Method to invoke Off command.

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

rtype: (ResultCode, str)

raises: DevFailed if the command execution is not successful

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

A class for SubarrayNodelow's AssignResources() command.

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

do(*argin*)
Method to invoke AssignResources command.

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":[[0,8,1,1],[8,8,2,1],[24,16,2,1]],"station_beam_ids":[1]}

return: A tuple containing ResultCode and string.

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

A class for SKASubarrayLow's ReleaseAllResources() command.

It invokes ReleaseAllResources command on Subarray Node Low.

do()
Method to invoke ReleaseAllResources command.

return: 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.

rtype: (ResultCode, str)

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

A class for SubarrayNodeLow's Configure() command.

Configures the resources assigned to the Mccs Subarray Leaf Node.

do(*argin*)
Method to invoke Configure command.

Parameters **argin** – DevString.

JSON string example is:

{“mccs”:{“stations”:[{“station_id”:1},{“station_id”:2}],”subarray_beams”:[{“subarray_id”:1, “subarray_beam_id”:1,”target”:{“system”:”HORIZON”,”name”:”DriftScan”,”Az”:180.0,”El”:45.0}, “update_rate”:0.0,”channels”:[[0,8,1,1],[8,8,2,1],[24,16,2,1]]}]},”tmc”:{“scanDuration”:10.0}}

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

rtype: (ReturnCode, str)

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

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

A class for SubarrayNodeLow’s Scan() command.

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.

do(*argin*)

Method to invoke Scan command.

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

JSON string example as follows:

{“mccs”:{“id”:1,”scan_time”:0.0}} Note: Above JSON string can be used as an input argument while invoking this command from JIVE.

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

rtype: (ReturnCode, str)

raises: DevFailed if the command execution is not successful

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

A class for SubarrayNodeLow’s EndScan() command.

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

do()

Method to invoke EndScan command.

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

rtype: (ReturnCode, str)

raises: DevFailed if the command execution is not successful.

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

A class for SubarrayNodeLow’s End() command.

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

do()

Method to invoke End command.

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

rtype: (ResultCode, str)

raises: DevFailed if the command execution is not successful.

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

A class for SubarrayNode's Abort() command.

This command on Subarray Node Low invokes Abort command on MCCS Subarray Leaf Node and aborts ongoing activity.

do()

Method to invoke Abort command.

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

rtype: (ResultCode, str)

raises: DevFailed if error occurs in invoking command on MCCS Subarrayleaf node.

class tmcprototype.subarraynodelow.src.subarraynodelow.obsreset_command.**ObsReset**(**args*, ***kwargs*)

A class for Low SubarrayNode's ObsReset() command.

This command invokes ObsReset command on Mccs Subarray Leaf Node.

do()

Method to invoke ObsReset command.

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

rtype: (ResultCode, str)

raises: DevFailed if error occurs while invoking command on MccsSubarrayLeafNode.

CHAPTER 3

Dish Leaf Node

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

A Leaf control node for DishMaster.

Device Properties

DishMasterFQDN: FQDN of Dish Master Device

Device Attributes

activityMessage: String providing information about the current activity in DishLeaf Node.

dishHealthState: Forwarded attribute to provide Dish Master Health State

dishPointingState: Forwarded attribute to provide Dish Master Pointing State

class tmcprototype.dishleafnode.src.dishleafnode.setoperatemode_command.**SetOperateMode**(**args*, ***kw*

A class for DishLeafNode's SetOperateMode() command.

Invokes SetOperateMode command on DishMaster.

do()

Method to invoke SetOperateMode command on DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking SetOperateMode command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.setstandbyfpmode_command.**SetStandbyFPMode**

A class for DishLeafNode's SetStandbyFPMode() command.

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

do()

Method to Invoke SetStandbyFPMode on DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking SetStandbyFPMode command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.setstandbylpmode_command.**SetStandbyLPMode**

A class for DishLeafNode's SetStandbyLPMode() command.

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

do()

Method to invoke SetStandbyLPMode command on DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking SetStandbyLPMode command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.setstowmode_command.**SetStowMode**(**args*, ***kwargs*)

A class for DishLeafNode's SetStowMode() command.

do()

Invokes SetStowMode command on DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking SetStowMode command on DishMaster.

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

A class for DishLeafNode's Configure() command.

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.

do(*argin*)

Method to invoke Configure command on dish.

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"}}

return: None

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

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

A class for DishLeafNode's Scan() command.

do(*argin*)

Invokes Scan command on DishMaster.

param argin: timestamp

return: None

raises: DevFailed If error occurs while invoking Scan command on DishMaster.

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

A class for DishLeafNode's EndScan() command.

do(*argin*)

Invokes Endscan command on DishMaster.

param argin: timestamp

raises: DevFailed If error occurs while invoking StopCapture command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.slew_command.**Slew**(**args*, ***kwargs*)

A class for DishLeafNode's SlewCommand() command.

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

do(*argin*)

Method to invoke Slew command on Dish Master.

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

return: None

raises: DevFailed If error occurs while invoking Slew command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.startcapture_command.**StartCapture**(**args*, ***kwargs*)

A class for DishLeafNode's StartCapture() command.

do(*argin*)

Invokes StartCapture command on DishMaster on the set configured band.

param argin: timestamp

return: None

raises: DevFailed If error occurs while invoking StartCapture command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.stopcapture_command.**StopCapture**(**args*, ***kwargs*)

A class for DishLeafNode's StopCapture() command.

do(*argin*)

Invokes StopCapture command on DishMaster on the set configured band.

param argin: timestamp

return: None

raises: DevFailed If error occurs while invoking StopCapture command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.stoptrack_command.**StopTrack**(**args*, ***kwargs*)

A class for DishLeafNode's StopTrack() command.

do()

Invokes TrackStop command on the DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking TrackStop command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.track_command.**Track**(**args*, ***kwargs*)

A class for DishLeafNode's Track() command.

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"}}

return: None

raises: DevFailed If error occurs while invoking Track command on DishMaster.

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

A class for DishLeafNode's Abort command.

do()

Invokes TrackStop command on the DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking TrackStop command on DishMaster.

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

A class for DishLeafNode's Restart command.

Invokes Restart command on the DishMaster.

do()

Method to invoke Restart command on the DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking StopCapture command on DishMaster.

class tmcprototype.dishleafnode.src.dishleafnode.obsreset_command.**ObsReset**(**args*, ***kwargs*)

A class for DishLeafNode's ObsReset command.

do()

Invokes ObsReset command on the DishMaster.

param argin: None

return: None

raises: DevFailed If error occurs while invoking StopCapture command on DishMaster.

CHAPTER 4

Dish Master

override class with command handlers for dsh-lmc.

class tmcprototype.dishmaster.src.dishmaster.dish_master_behaviour.**AzEl**(*azim*, *elev*)

azim
Alias for field number 0

elev
Alias for field number 1

class tmcprototype.dishmaster.src.dishmaster.dish_master_behaviour.**OverrideDish**

AZIM_DRIVE_MAX_RATE = 3.0

AZIM_IDX = 1

ELEV_DRIVE_MAX_RATE = 1.0

ELEV_IDX = 2

FAILED = 2

MAINT_AZIM = 90.0

MAX_DESIRED_AZIM = 270.0

MAX_DESIRED_ELEV = 90.0

MAX_SAMPLE_HISTORY = 2400

MIN_DESIRED_AZIM = -270.0

MIN_DESIRED_ELEV = 15.0

OK = 0

TS_IDX = 0

action_configureband1 (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 1. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband2 (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 2. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband3 (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 3. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband4 (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 4. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband5a (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 5a. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model

- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband5b (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 5b. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_configureband5c (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the CONFIGURE Dish Element Mode, and returns to the caller. To configure the Dish to operate in frequency band 5c. On completion of the band configuration, Dish will automatically revert to the previous Dish mode (OPERATE or STANDBY_FP).

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP, OPERATE, STOW).

action_lowpower (*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the LOW power state. All subsystems go into a low power state to power only the essential equipment. Specifically the Helium compressor will be set to a low power consumption, and the drives will be disabled. When issued a STOW command while in LOW power, the DS controller should be able to turn the drives on, stow the dish and turn the drives off again. The purpose of this mode is to enable the observatory to perform power management (load curtailment), and also to conserve energy for non-operating dishes.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STOW, MAINTENANCE).

action_resettracktable (*model*, *tango_dev=None*, *data_input=None*)
Resets the coordinates in the queue. Clear ACU's table (should show number of coordinates drops to zero)

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

action_resettracktablebuffer (*model*, *tango_dev=None*, *data_input=None*)
Resets the Dish LMC's buffer. (In our case it's desired_pointings)

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

action_scan(*model*, *tango_dev=None*, *data_input=None*)
The Dish is tracking the commanded pointing positions within the specified SCAN pointing accuracy.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OPERATE).

action_setmaintenancemode(*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the MAINTENANCE Dish Element Mode, and returns to the caller. To go into a state that is safe to approach the Dish by a maintainer, and to enable the Engineering interface to allow direct access to low level control and monitoring by engineers and maintainers. This mode will also enable engineers and maintainers to upgrade SW and FW. Dish also enters this mode when an emergency stop button is pressed.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_LP, STANDBY_FP).

action_setoperatemode(*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the OPERATE Dish Element Mode, and returns to the caller. This mode fulfils the main purpose of the Dish, which is to point to designated directions while capturing data and transmitting it to CSP.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_FP).

action_setstandbyfpmode(*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the STANDBY_FP Dish Element Mode, and returns to the caller. To prepare all subsystems for active observation, once a command is received by TM to go to the FULL_POWER mode.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (STANDBY_LP, STOW, OPERATE, MAINTENANCE).

action_setstandbylpmode(*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the STANDBY_LP Dish Element Mode, and returns to the caller. Standby_LP is the default mode when the Dish is configured for low power consumption, and is the mode wherein Dish ends after a start up procedure.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OFF, STARTUP, SHUTDOWN, STANDBY_FP, MAINTENANCE, STOW, CONFIG, OPERATE).

action_setstowmode(*model*, *tango_dev=None*, *data_input=None*)
This command triggers the Dish to transition to the STOW Dish Element Mode, and returns to the caller. To point the dish in a direction that minimises the wind loads on the structure, for survival in strong wind conditions. The Dish is able to observe in the stow position, for the purpose of transient detection.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OFF, STARTUP, SHUTDOWN, STANDBY_LP, STANDBY_FP, MAINTENANCE, CONFIG, OPERATE).

action_slew(*model*, *tango_dev=None*, *data_input=None*)
The Dish moves to the commanded pointing angle at the maximum speed, as defined by the specified slew rate.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – list [0]: Azimuth [1]: Elevation

Raises **DevFailed** – dishMode is not in any of the allowed modes (OPERATE).

action_startcapture(*model*, *tango_dev=None*, *data_input=None*)
Triggers the dish to start capturing the data on the configured band.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OPERATE) or configuredBand is (NONE, UNKNOWN, ERROR, UNDEFINED).

action_stopcapture(*model*, *tango_dev=None*, *data_input=None*)
Triggers the dish to stop capturing the data on the configured band.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

action_track(*model*, *tango_dev=None*, *data_input=None*)
The Dish is tracking the commanded pointing positions within the specified TRACK pointing accuracy.

Parameters

- **model** – tango_simlib.model.Model
- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OPERATE).

action_trackstop(*model*, *tango_dev=None*, *data_input=None*)
The Dish will stop tracking but will not apply brakes. Stops movement, but doesn't clear tables/queues.

Parameters

- **model** – tango_simlib.model.Model

- **data_input** – None

Raises **DevFailed** – dishMode is not in any of the allowed modes (OPERATE).

actual_position = AzEl(azim=0.0, elev=30.0)

desired_pointings = []

ensure_within_mechanical_limits(*next_pos*)

find_next_position(*desired_pointings*, *sim_time*)
Return the latest desiredPointing not in the future, or last requested.

get_new_unverified_pointings(*model*)
Return the latest list of coordinates

Parameters **model** – Model The device Model

Returns list - Empty if no updates have occured since the last time - 1 entry of desiredPointing if it is the latest - All the entries of programTrackTable if it is the latest (7 in testing)

get_new_valid_pointings(*model*)

get_rate_limited_position(*current_pos*, *next_pos*, *dt*)

static is_movement_allowed(*model*)

is_on_target()

last_coordinate_update_timestamp = 0.0

move_towards_target(*sim_time*, *dt*)

pre_update(*model*, *sim_time*, *dt*)

requested_position = AzEl(azim=0.0, elev=30.0)

set_achieved_pointing_attribute(*model*, *sim_time*, *position*)

static set_lock_attribute(*model*, *target_reached*)

update_desired_pointing_history(*model*)

update_movement_attributes(*model*, *sim_time*)

tmcprototype.dishmaster.src.dishmaster.dish_master_behaviour.**get_direction_sign**(*here*, *there*)
Return sign (+1 or -1) required to move from here to there.

tmcprototype.dishmaster.src.dishmaster.dish_master_behaviour.**get_enum_str**(*quantity*)
Returns the enum label of an enumerated data type

Parameters **quantity** – object The quantity object of a DevEnum attribute

Returns str Current string value of a DevEnum attribute

tmcprototype.dishmaster.src.dishmaster.dish_master_behaviour.**set_enum**(*quantity*, *label*, *timestamp*)
Sets the quantity last_val attribute to index of label

Parameters

- **quantity** – object The quantity object from model
- **label** – str The desired label from enum list
- **timestamp** – float The time now

CHAPTER 5

CSP Master Leaf Node

class tmcprototype.cspmasterleafnode.src.cspmasterleafnode.csp_master_leaf_node.**CspMasterLe**

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

Device Properties

CspMasterFQDN: Property to provide FQDN of CSP Master Device

Device Attributes

cspHealthState: Forwarded attribute to provide CSP Master Health State

activityMessage: Attribute to provide activity message

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

A class for CspMasterLeafNode's Off() command. Off command is inherited from SKABaseDevice.

It Sets the State to Off.

do()

Method to invoke Off command on CSP Element.

param argin: None.

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

rtype: (ResultCode, str)

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

A class for CspMasterLeafNode's On() command. On command is inherited from SKABaseDevice.

It Sets the State to On.

do()

Method to invoke On command on CSP Element.

param argin: None

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

rtype: (ResultCode, str)

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

class tmcprototype.cspmasterleafnode.src.cspmasterleafnode.standby_command.**Standby**(**args*, ***kwargs*)

A class for CspMasterLeafNode's Standby() command. Standby command is inherited from BaseCommand.

It Sets the OpState to Standby.

do(*argin*)

Method to invoke Standby command on CSP Element.

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.

return: None

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

CHAPTER 6

SDP Subarray Leaf Node

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.sdp_subarray_leaf_node.**SdpSu**

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

Device Properties

SdpSubarrayFQDN: FQDN of the SDP Subarray Tango Device Server.

Device Attributes

receiveAddresses: This attribute is used for testing purposes. In the unit test cases it is used to provide FQDN of receiveAddresses attribute from SDP.

activityMessage: String providing information about the current activity in SDP Subarray Leaf Node.

activeProcessingBlocks: This is a attribute from SDP Subarray which depicts the active Processing Blocks in the SDP Subarray.

sdpSubarrayHealthState: Attribute to provide SDP Subarray Health State.

sdpSubarrayObsState: Attribute to show ObsState of Tango Device.

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

A class for sdpSubarrayLeafNode's Abort() command.

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

do()

Method to invoke Abort command on SDP Subarray.

return: None

raises: DevFailed if error occurs while invoking command on SDP Subarray.

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.assign_resources_command.**Ass**

A class for SdpSubarayLeafNode's AssignResources() command.

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

do(*argin*)

Method to invoke AssignResources command on SDP Subarray.

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

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

SBI ID : String

max_length: Float

Scan types: Consist of Scan type id name

scan_type: DevVarStringArray

Processing blocks: Mandatory JSON object consisting of

processing_blocks: DevVarStringArray

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

Note: Enter input without spaces

return: None

raises: ValueError if input argument json string contains invalid value.

DevFailed if the command execution is not successful.

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

A class for SdpSubarrayLeafNode's Configure() command.

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

do(*argin*)

Method to invoke Configure command on SDP Subarray.

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

Example:

{ "scan_type": "science_A" }

return: 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

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

A class for SdpSubarrayLeafNode's End command.

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

do()

Method to invoke End command on SDP Subarray.

return: None

raises: DevFailed if the command execution is not successful.

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.endscan_command.**EndScan**(**args*, ***kw*

A class for SdpSubarrayLeafNode's EndScan() command.

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

do()

Method to invoke EndScan command on SDP Subarray.

Parameters **argin** – None

return: None

raises: DevFailed if the command execution is not successful.

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.obsreset_command.**ObsReset**(**a*, ***

A class for SdpSubarrayLeafNode's ObsResetCommand() command.

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

do()

Method to invoke ObsReset command on SDP Subarray.

Parameters **argin** – None

return: None

raises: DevFailed if error occurs while invoking command on SDP Subarray.

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

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

Invokes Off command on the SDP Subarray.

do()

Method to invoke Off command on SDP Subarray.

Parameters **argin** – None.

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

rtype: (ResultCode, str)

raises: DevFailed if error occurs while invoking command on SDPSubarray.

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

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

Invokes On command on the SDP Subarray.

do()

Method to invoke On command on SDP Subarray.

Parameters **argin** – None.

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

rtype: (ResultCode, str)

raises: DevFailed if error occurs while invoking command on SDPSubarray.

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.release_resources_command.**Re**

A class for SdpSubarayLeafNode's ReleaseAllResources() command.

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

do()

Method to invoke ReleaseResources command on SDP Subarray.

Parameters **argin** – None.

return: None

raises: DevFailed if the command execution is not successful.

class tmcprototype.sdpsubarrayleafnode.src.sdpsubarrayleafnode.restart_command.**Restart**(**args*, ***kw*

A class for sdpSubarrayLeafNode's Restart() command.

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

do()

Method to invoke Restart command on SDP Subarray.

return: None

raises: DevFailed if error occurs while invoking command on SDPSubarray.

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

A class for SdpSubarrayLeafNode's Scan() command.

Invoke Scan command to SDP Subarray.

do(*argin*)

Method to invoke Scan command on SDP Subarray.

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

Example: {"id":1}

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

return: None

raises: DevFailed if the command execution is not successful.

CHAPTER 7

CSP Subarray Leaf Node

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.

Device Properties

CspSubarrayFQDN: Property to provide FQDN of CSP Subarray Device

Device Attributes

cspsubarrayHealthState: Forwarded attribute to provide CSP Subarray Health State

cspSubarrayObsState: Forwarded attribute to provide CSP Subarray Observation State

activityMessage: String providing information about the current activity in CspSubarrayLeaf Node.

delayModel: Attribute to provide the delay model.

versionInfo: Provides Version information of TANGO device.

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

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

Invokes On command on the CSP Subarray.

do()

Method to invoke On command on CSP Subarray.

param argin: None

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

rtype: (ResultCode, str)

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

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

Invokes Off command on the CSP Subarray.

do()

Method to invoke Off command on CSP Subarray.

param argin: None

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

rtype: (ResultCode, str)

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.assign_resources_command.**Ass**

A class for CspSubarrayLeafNode's AssignResources() command.

It accepts subarrayID and receptor ids in JSON string format and invokes AssignResources command on CSP Subarray.

do(*argin*)

Method to invoke AssignResources command on CSP Subarray.

Parameters **The string in JSON format. The JSON contains following values** (*argin:DevString.*) – subarrayID: integer

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:

{ "subarrayID":1, "dish": { "receptorIDList": [

"0001", "0002"

}

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

return: 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

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.release_all_resources_comma

A class for CspSubarrayLeafNode's ReleaseAllResources() command. ReleaseAllResources command is inherited from BaseCommand.

It invokes ReleaseAllResources command on Csp Subarray and releases all the resources assigned to CSP Subarray.

do()

Method to invoke ReleaseAllResources command on CSP Subarray.

return: None

raises: DevFailed if the command execution is not successful

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.configure_command.**ConfigureC**

A class for CspSubarrayLeafNode's Configure() command. Configure command is inherited from BaseCommand.

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

do(*argin*)

Method to invoke Configure command on CSP Subarray.

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

Example: {"interface":"https://schema.skatelescope.org/ska-csp-configure/1.0","subarray":{"subarrayName":"science period 23"},"common":{"id":"sbi-mvp01-20200325-00001-science_A","frequencyBand":"1","subarrayID":"1"},
"cbf":{"fsp":[{"fspID":1,"functionMode":"CORR","frequencySliceID":1,"integrationTime":1400,"corrBandwidth":0,"chan
:[[0,"192.168.1.1"]],"outputPort":[[0,9000,1]]},{"fspID":2,"functionMode":"CORR","frequencySliceID":2,"integrationTim
"outputLinkMap":[[0,4],[200,5]],"outputHost":[[0,"192.168.1.1"]],"outputPort":[[0,9744,1]]}],"vlbi":{},"delayModelSubsc
"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.

return: None

raises: DevFailed if the command execution is not successful

ValueError if input argument json string contains invalid value

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.scan_command.**StartScanComma**

A class for CspSubarrayLeafNode's StartScan() command. StartScan command is inherited from BaseCommand.

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

do(*argin*)

Method to invoke StartScan command on CSP Subarray.

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

Example: {"id":1}

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

return: None

raises: DevFailed if the command execution is not successful

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.end_scan_command.**EndScanComm**

A class for CspSubarrayLeafNode's EndScan() command. EndScan command is inherited from BaseCommand.

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

do()

Method to invoke Endscan command on CSP Subarray.

return: None

raises: DevFailed if the command execution is not successful

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.end_command.**GoToIdleCommand**

A class for CspSubarrayLeafNode's GoToIdle() command. GoToIdle command is inherited from BaseCommand.

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

do()

Method to invoke GoToIdle command on CSP Subarray.

return: None

raises: DevFailed if the command execution is not successful

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.abort_command.**AbortCommand**(

A class for CSPSubarrayLeafNode's Abort() command.

Command to abort the current operation being done on the CSP Subarray.

do()

This command invokes Abort command on CSP Subarray.

return: None

raises: DevFailed if error occurs while invoking command on CSP Subarray.

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.restart_command.**RestartComma**

A class for CSPSubarrayLeafNode's Restart() command. Restart command is inherited from BaseCommand.

This command invokes Restart command on CSP Subarray.

do()

Method to invoke Restart command on CSP Subarray.

return: None

raises: DevFailed if error occurs while invoking the command on CSP Subarray.

class tmcprototype.cspsubarrayleafnode.src.cspsubarrayleafnode.obsreset_command.**ObsResetCo**

A class for CSPSubarrayLeafNode's ObsReset() command. ObsReset command is inherited from BaseCommand.

Command to reset the Csp Subarray and bring it to its RESETTING state.

do()

Method to invoke ObsReset command on CSP Subarray.

param argin: None

return: None

raises: DevFailed if error occurs while invoking the command on CSP Subarray.

CHAPTER 8

SDP Master Leaf Node

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.

Device Properties

SdpMasterFQDN: Property to provide FQDN of SDP Master Device

Device Attributes

versionInfo: Provides Version information of TANGO device.

activityMessage: String providing information about the current activity in SDP Master Leaf Node.

ProcessingBlockList: List of Processing Block devices

sdpHealthState: Forwarded attribute to provide SDP Master Health State

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

A class for SDP master's Off() command. Off command is inherited from SKABaseDevice.

It Sets the State to Off.

do()

Method to invoke Off command on SDP Master.

Parameters **argin** – None.

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

rtype: (ResultCode, str)

class tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.disable_command.**Disable**(**args*, ***kwargs*)

A class for SDP master's Disable() command. Disable command is inherited from BaseCommand.

Sets the State to Disable.

do()
Method to invoke Disable command on SDP Master.

Parameters **argin** – None.

return: None

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

A class for SDP master's On() command. On command is inherited from SKABaseDevice.

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

do()
Method to invoke On command on SDP Master.

Parameters **argin** – None.

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

rtype: (ResultCode, str)

class tmcprototype.sdpmasterleafnode.src.sdpmasterleafnode.standby_command.**Standby**(**args*, ***kwargs*)

A class for SDP Master's Standby() command. Standby command is inherited from BaseCommand.

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.

do()
Method to invoke Standby command on SDP Master.

Parameters **argin** – None.

return: None

CHAPTER 9

MCCS Master Leaf Node

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

Device Properties

MccsMasterFQDN: Property to provide FQDN of MCCS Master Device

Device Attributes

mccsHealthState: Forwarded attribute to provide MCCS Master Health State

activityMessage: String providing information about the current activity in MccsMasterLeafNode.

class tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.assign_resources_command.**Assig**

A class for MccsMasterLeafNode's AssignResources() command.

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

do(*argin*)

Method to invoke AssignResources command on MCCS Master.

Parameters **argin** – StringType. The string in JSON format.

Example:

{ "subarray_id": 1, "station_ids":[1,2], "channels": [[[0,8,1,1],[8,8,2,1],[24,16,2,1]], "station_beam_ids": [1]

}

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

return: 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

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

A class for MccsMasterLeafNode's Off() command. Off command is inherited from SKABaseDevice.

It Sets the State to Off.

do()

Method to invoke Off command on the MCCS.

param argin: None.

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

rtype: (ResultCode, str)

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

A class for MccsMasterLeafNode's On() command. On command is inherited from SKABaseDevice.

It Sets the State to On.

do()

Method to invoke On command on the MCCS.

param argin: None

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

rtype: (ResultCode, str)

class tmcprototype.mccsmasterleafnode.src.mccsmasterleafnode.release_resources_command.**Rele**

A class for MccsMasterLeafNode's ReleaseResources() command.

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

do(*argin*)

Method to invoke ReleaseResources command on Mccs Master.

Parameters **argin** – StringType. The string in JSON format.

Example:

{

"subarray_id": 1, "release_all": true

}

return: None.

raises: DevFailed if the command execution is not successful

ValueError if invalid json string.

CHAPTER 10

MCCS Subarray Leaf Node

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.

Device Properties

MccsSubarrayFQDN: FQDN of MCCS Subarray.

Device Attributes

mccsSubarrayHealthState: Forwarded attribute to provide MCCS Subarray Health State.

mccsSubarrayObsState: Attribute to provide MCCS Subarray Observation State.

activityMessage: String providing information about the current activity in MCCS Subarray Leaf Node.

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.abort_command.**Abort**(**args*, ***kwarg*

A class for MccsSubarrayLeafNode's Abort() command.

Command to abort the current operation being done on the MCCS Subarray.

do()

Method to invoke Abort command on MCCS Subarray.

Parameters **argin** – None

return: None

raises: DevFailed if the command execution is not successful

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.configure_command.**Configu**

A class for MccsSubarrayLeafNode's Configure() command.

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

do(*argin*)
Method to invoke Configure command on MCCS Subarray.

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

Example: {“mccs”:{“stations”:[{“station_id”:1},{“station_id”:2}],”subarray_beams”:[{“subarray_id”:1,”subarray_beam_id
“target”:{“system”:”HORIZON”,”name”:”DriftScan”,”Az”:180.0,”El”:45.0},”update_rate”:0.0,”channels”:[[0,8,1,1],
[8,8,2,1],[24,16,2,1]]}]}}

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

return: None

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

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.end_command.**End**(**args*, ***kwargs*)
A class for MccsSubarrayLeafNode’s End() command.

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

do()
Method to invoke End command on MCCS Subarray.

return: None

raises: DevFailed if the command execution is not successful.

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.end_scan_command.**EndScan**(*

A class for MccsSubarrayLeafNode’s EndScan() command.

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

do()
Method to invoke EndScan command on MCCS Subarray.

raises: DevFailed if the command execution is not successful.

AssertionError if MccsSubarray is not in SCANNING obsState.

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.obsreset_command.**ObsReset**

A class for MccsSubarrayLeafNode’s ObsReset() command.

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

do()
Method to invoke ObsReset command on MCCS Subarray.

Parameters **argin** – None

return: None

raises: DevFailed if error occurs while invoking the command on MccsSubarray.

class tmcprototype.mccssubarrayleafnode.src.mccssubarrayleafnode.scan_command.**Scan**(**args*, ***kwargs*)
A class for MccsSubarrayLeafNode’s Scan() command.

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

do(*argin*)

Method to invoke Scan command on MCCS Subarray.

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

Example: {“id”:1, “scan_time”:0.0}

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

return: None

raises: DevFailed if the command execution is not successful

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

A

C

D

E

F

G

I

L

M

O

P

R

S

T

U