Subarray Node Documentation

Release 1.0

NCRA India

Jun 29, 2022

GETTING STARTED

This project is developing the SubarrayNode (Mid and Low) component of the Telescope Monitoring and Control (TMC) prototype, for the Square Kilometre Array.

CHAPTER

ONE

GETTING STARTED

This page contains instructions for software developers who want to get started with usage and development of the SubarrayNode.

1.1 Background

Detailed information on how the SKA Software development community works is available at the SKA software developer portal. There you will find guidelines, policies, standards and a range of other documentation.

1.2 Set up your development environment

This project is structured to use k8s for development and testing so that the build environment, test environment and test results are all completely reproducible and are independent of host environment. It uses `make` to provide a consistent UI (run `make help` for targets documentation).

1.2.1 Install minikube

You will need to install *minikube* or equivalent k8s installation in order to set up your test environment. You can follow the instruction here: :: git clone git@gitlab.com:ska-telescope/sdi/deploy-minikube.git cd deploy-minikube make all eval $(minikube docker-env)

Please note that the command `eval $(minikube docker-env)` will point your local docker client at the docker-in-docker for minikube. Use this only for building the docker image and another shell for other work.

1.2.2 How to Use

Clone this repo: :: git clone https://gitlab.com/ska-telescope/ska-tmc-subarraynode.git cd ska-tmc-subarraynode

Install dependencies :: apt update apt install -y curl git build-essential libboost-python-dev libtango-dev curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py | python3 - source $HOME/.poetry/env

Please note that:

- the *libtango-dev* will install an old version of the TANGO-controls framework (9.2.5);
- the best way to get the framework is compiling it (instructions can be found here);
- the above script has been tested with Ubuntu 20.04.

During this step, `libtango-dev` instalation can ask for the Tango Server IP:PORT. Just accept the default proposed value.

Install python requirements for linting and unit testing: :: $ poetry install

Activate the poetry environment: :: $ source $(poetry env info –path)/bin/activate

Alternate way to install and activate poetry

Follow the steps till installation of dependencies. Then run below command: :: $ virtualenv cn_venv $ source cn_venv/bin/activate $ make requirements

Run python-test: :: $ make python-test PyTango 9.3.3 (9, 3, 3) PyTango compiled with: Python : 3.8.5 Numpy : 0.0.0 ## output generated from a WSL windows machine Tango : 9.2.5 Boost : 1.71.0

PyTango runtime is: Python : 3.8.5 Numpy : None Tango : 9.2.5

PyTango running on: uname_result(system='Linux', node='LAPTOP-5LBGJH83', release='4.19.128-microsoft-standard', version='#1 SMP Tue Jun 23 12:58:10 UTC 2020', machine='x86_64', processor='x86_64')

============================= test session starts ============================== platform linux – Python 3.8.5, pytest-5.4.3, py-1.10.0, pluggy-0.13.1 – /home/ [....]

———————————— JSON report ————————————- JSON report written to: build/reports/report.json (165946 bytes)

———— coverage: platform linux, python 3.8.5-final-0 ———— Coverage HTML written to dir build/htmlcov Coverage XML written to file build/reports/code-coverage.xml

======================== 48 passed, 5 deselected in 42.42s =======================

Formatting the code: :: $ make python-format [...] ——————————————————————————— Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

Python linting: :: $ make python-lint [...] ——————————————————————————— Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

CHAPTER

TWO

SUBARRAYNODE CODE QUALITY GUIDELINES

2.1 Code formatting / style

2.1.1 Black

SubarrayNode uses the `black` code formatter to format its code. Formatting can be checked using the command `make python-format`.

The CI pipeline does check that if code has been formatted using black or not.

2.1.2 Linting

SubarrayNode uses below libraries/utilities for linting. Linting can be checked using command `make python-lint`.

- **`isort` - It provides a command line utility, Python library and** plugins for various editors to quickly sort all your imports.
- `black` - It is used to check if the code has been blacked.
- **`flake8` - It is used to check code base against coding style (PEP8),** programming errors (like "library imported but unused" and "Undefined name"),etc.
- **`pylint` - It is looks for programming errors, helps enforcing a coding standard,** sniffs for code smells and offers simple refactoring suggestions.

2.2 Test coverage

SubarrayNode uses pytest to test its code, with the pytest-cov plugin for measuring coverage.

CHAPTER

THREE

API

3.1 ska_tmc_subarraynode package

3.1.1 Subpackages

ska_tmc_subarraynode.commands package

Submodules

ska_tmc_subarraynode.commands.abstract_command module

class ska_tmc_subarraynode.commands.abstract_command.**AbstractOnOff**(**args: Any*, ***kwargs: Any*)

Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name

check_allowed_low()

Checks whether this command is allowed It checks that the device is in a state to perform this command and that all the component needed for the operation are not unresponsive

Returns True if this command is allowed

Return type boolean

check_allowed_mid()

Checks whether this command is allowed It checks that the device is in a state to perform this command and that all the component needed for the operation are not unresponsive

Returns True if this command is allowed

Return type boolean

init_adapters_low()

init_adapters_mid()

class ska_tmc_subarraynode.commands.abstract_command.**SubarrayNodeCommand**(**args: Any*, ***kwargs: Any*)

Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name

check_allowed()

do(*argin=None*)

init_adapters()

is_allowed(*raise_if_disallowed=False*)

is_allowed_low(*raise_if_disallowed*)

is_allowed_mid(*raise_if_disallowed*)

ska_tmc_subarraynode.commands.assign_resources_command module

AssignResourcesCommand class for SubarrayNode.

class ska_tmc_subarraynode.commands.assign_resources_command.**AssignResources**(**args: Any*, ***kwargs: Any*)

Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.AssignResourcesCommand._name

A class for SubarrayNode's AssignResources() command.

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

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

assign_csp_resources(*receptor_ids*)

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

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

Example: ['0001', '0002']

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

assign_sdp_resources(*argin*)

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

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

Returns

List of strings.

Example: ['PB1', 'PB2']

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

completed()

do(*argin=None*)

do_low(*argin*)

Method to invoke AssignResources command on Subarraynode low.

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

mccs: subarray_beam_ids: list of integers

station_ids: list of integers

channel_blocks: list of integers

Example:

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

Returns A tuple containing ResultCode and string.

do_mid(*argin*)

Method to invoke AssignResources command on subarraynode mid.

Parameters **argin** – DevString.

Example:

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

Returns A tuple containing a return code and string of Resources added to the Subarray. Example of string of Resources : [“0001”,”0002”] as argout if allocation successful.

rtype: (ResultCode, str)

Raises

- **ValueError if input argument json string contains invalid value** –
- **Exception if the command execution is not successful** –

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)

Method to check whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)
Method to check whether this command is allowed in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

set_up_dish_data(*receptor_ids*)
Adds the receptors in dish leaf node group. The healthState and pointintgState attributes of all all the dishes are subscribed.

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

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

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

ska_tmc_subarraynode.commands.configure_command module

Configure Command class for SubarrayNode.

class ska_tmc_subarraynode.commands.configure_command.**Configure**(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.ConfigureCommand._name

A class for SubarrayNode’s Configure() command.

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

check_only_dish_config(*scan_configuration*)

completed()

do(*argin=None*)

do_low(*argin*)
Method to invoke Configure command on the Mccs Subarray Leaf Node.

Parameters **argin** – DevString.

JSON string example is:

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

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

rtype: (ReturnCode, str)

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

Parameters **argin** – DevString.

JSON string that includes pointing parameters of Dish - Azimuth and Elevation Angle, CSP Configuration and SDP Configuration parameters. JSON string example is: {"interface":"https://schema.skao.int/ska-tmc-configure/2.0","transaction_id":"txn-....-00001", "pointing":{"target":{"reference_frame":"ICRS","target_name":"Polaris Australis","ra":"21:08:47.92", "dec":"-88:57:22.9"}},"dish":{"receiver_band":"1"},"csp":{"interface": "https://schema.skao.int/ska-csp-configure/2.0","subarray":{"subarray_name":"science period 23"}, "common":{"config_id":"sbi-mvp01-20200325-00001-science_A","frequency_band":"1","subarray_id":1}, "cbf":{"fsp":[{"fsp_id":1,"function_mode":"CORR","frequency_slice_id":1,"integration_factor":1, "zoom_factor":0,"channel_averaging_map":[[0,2],[744,0]],"channel_offset":0,"output_link_map": [[0,0],[200,1]]},{"fsp_id":2,"function_mode":"CORR","frequency_slice_id":2,"integration_factor":1, "zoom_factor":1,"channel_averaging_map":[[0,2],[744,0]],"channel_offset":744,"output_link_map": [[0,4],[200,5]],"zoom_window_tuning":650000}],"vlbi":{}},"pss":{},"pst":{}},"sdp":{"interface": "https://schema.skao.int/ska-sdp-configure/0.3","scan_type":"science_A"},"tmc":{"scan_duration":10.0}}

Note: While invoking this command from JIVE, provide above JSON string without any space.

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

rtype: (ReturnCode, str)

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)

Whether this command is allowed to run in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)

Method to check whether this command is allowed in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

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

Bases: object

static build_up_csp_cmd_data(*scan_config*, *delay_model_subscription*, *receive_addresses_map*, *component_manager*)

Here the input data for CSP is build which is used in configuration of CSP. Below is the csp_config_schema variable value generated by using ska_telmodel library.

{"interface":"https://schema.skao.int/ska-csp-configure/2.0","subarray":{"subarray_name": "science period 23"},"common":{"config_id":"sbi-mvp01-20200325-00001-science_A", "frequency_band":"1","subarray_id":1},"cbf":{"fsp":[{"fsp_id":1,"function_mode":"CORR", "frequency_slice_id":1,"integration_factor":1,"zoom_factor":0,"channel_averaging_map": [[0,2],[744,0]],"channel_offset":0,"output_link_map":[[0,0],[200,1]],"output_host": [[0,"192.168.0.1"],[400,"192.168.0.2"]],"output_mac":[[0,"06-00-00-00-00-00"]], "output_port":[[0,9000,1],[400,9000,1]]},{"fsp_id":2,"function_mode":"CORR", "frequency_slice_id":2,"integration_factor":1,"zoom_factor":1,"channel_averaging_map": [[0,2],[744,0]],"channel_offset":744,"output_link_map":[[0,4],[200,5]],"zoom_window_tuning": 650000,"output_host":[[0,"192.168.0.3"],[400,"192.168.0.4"]],"output_mac": [[0,"06-00-00-00-00-01"]],"output_port":[[0,9000,1],[400,9000,1]]}],"vlbi":{}},"pss":{},"pst":{}}

Returns csp confiuration schema

static build_up_dsh_cmd_data(*scan_config*, *component_manager*)

static build_up_sdp_cmd_data(*scan_config*, *component_manager*)

ska_tmc_subarraynode.commands.end_command module

A class for TMC SubarrayNode's End() command

class ska_tmc_subarraynode.commands.end_command.**End**(**args: Any*, ***kwargs: Any*)

Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.EndCommand._name

A class for SubarrayNode's End() command.

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

do(*argin=None*)

do_low()

Method to invoke End command on MCCS Subarray Leaf Node.

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

rtype: (ResultCode, str)

do_mid()

Method to invoke End command on CSP Subarray Leaf Node, SDP Subarray Leaf Node and Dish Leaf Nodes.

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

rtype: (ResultCode, str)

end_csp()

End command on CSP Subarray Leaf Node

end_sdp()

End command on SDP Subarray Leaf Node

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)

Whether this command is allowed to run in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)

Whether this command is allowed to run in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

stop_dish_tracking()

ska_tmc_subarraynode.commands.end_scan_command module

A class for TMC SubarrayNode's EndScan() command.

class ska_tmc_subarraynode.commands.end_scan_command.**EndScan**(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.EndScanCommand._name

A class for SubarrayNode's EndScan() command.

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

do(*argin=None*)

do_low()
Method to invoke Endscan command.

Returns None

Raises DevFailed if the command execution is not successful. –

do_mid()
Method to invoke Endscan command.

Returns None

Raises DevFailed if the command execution is not successful. –

end_scan()

end_scan_low()

end_scan_mccs()
set up mccs devices

end_scan_mid()

endscan_csp()
set up csp devices

endscan_sdp()
set up sdp devices

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

ska_tmc_subarraynode.commands.obsreset_command module

ObsReset Command for SubarrayNode.

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

A class for SubarrayNode's ObsReset() command.

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

completed()

do(*argin=None*)

do_low()
Method to invoke ObsReset command on MCCS Subarray Leaf Node.

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

rtype: (ResultCode, str)

do_mid()
Method to invoke ObsReset command on CSP Subarray Leaf Node, SDP Subarray Leaf Node and Dish Leaf Nodes.

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

rtype: (ResultCode, str)

get_csp_subarray_obsstate()

get_sdp_subarray_obsstate()

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

obsreset_csp()
Invoke ObsReset command on CSP Subarray Leaf Node.

obsreset_sdp()
Invoke ObsReset command on SDP Subarray Leaf Node.

obsrest_dish()
Invoke ObsReset command on Dish Leaf Node.

ska_tmc_subarraynode.commands.release_all_resources_command module

ReleaseAllResources Command for SubarrayNode

class ska_tmc_subarraynode.commands.release_all_resources_command.**ReleaseAllResources**(**args: Any*, ***kwargs: Any*)

Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.ReleaseAllResourcesCommand._name

A class for SKASubarray's ReleaseAllResources() command.

It checks whether all resources are already released. If yes then it returns code FAILED. If not it Releases all the resources from the subarray i.e. Releases resources from TMC Subarray Node, CSP Subarray and SDP Subarray. Upon successful execution, all the resources of a given subarray get released and empty array is returned. Selective release is not yet supported.

clean_up_resources()
Clears the AssignedResources attribute.

Cleans dictionaries of the resources across the subarraynode.

Note: Currently there are only receptors allocated so only the receptor ids details are stored.

Parameters **argin** – None

Returns None

completed()

do()

do_low()
Method to invoke ReleaseAllResources command.

Returns A tuple containing a return code STARTED on successful release all resources and message.

rtype: (ResultCode, str)

do_mid()
Method to invoke ReleaseAllResources command.

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

rtype: (ResultCode, str)

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters **raise_if_disallowed** – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

`is_allowed_mid`(*raise_if_disallowed*)
Method to check whether this command is allowed in the current state of the state model.

Parameters `raise_if_disallowed` – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

`release_csp_resources`()
This function invokes releaseAllResources command on CSP Subarray via CSP Subarray Leaf Node.

Parameters `argin` – DevVoid

Returns DevVoid

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

Parameters `argin` – DevVoid

Returns DevVoid

ska_tmc_subarraynode.commands.reset_command module

Reset Command for SubarrayNode.

`class` ska_tmc_subarraynode.commands.reset_command.`Reset`(**args: Any*, ***kwargs: Any*)
Bases: `ska_tango_base.SKABaseDevice.ska_tango_base.SKABaseDevice.ResetCommand._name`

A class for SubarrayNode's Reset() command.

`do`(*argin=None*)

`do_low`(*argin=None*)
" Method to invoke Reset command on TMC Low Devices.

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

rtype: (ResultCode, str)

Raises `DevFailed.` –

`do_mid`(*argin=None*)
" Method to invoke Reset command on SubarrayNode.

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

rtype: (ResultCode, str)

Raises `Exception.` –

`is_allowed`(*raise_if_disallowed=True*)

`is_allowed_low`(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters `raise_if_disallowed` – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

`is_allowed_mid`(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters `raise_if_disallowed` – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

ska_tmc_subarraynode.commands.restart_command module

Restart Command for SubarrayNode.

`class` ska_tmc_subarraynode.commands.restart_command.**Restart**(**args: Any*, ***kwargs: Any*)
Bases: `ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.RestartCommand._name`

A class for SubarrayNode's Restart() command.

`clean_up_configuration`()
Removes group of dishes from tango group client.

Unsubscribes events for dish health state and dish pointing state.

Cleans dictionaries of the resources across the subarraynode.

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

Parameters `argin` – None

Returns None

`completed`()

`do`()

`do_low`()
This command on Subarray Node Low invokes Restart command on MCCS Subarray Leaf Node and restarts the ongoing activity.

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

rtype: (ResultCode, str)

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

`do_mid`()
This method invokes Restart command on CSPSubarrayLeafNode, SDpSubarrayLeafNode and DishLeafNode.

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

rtype: (ResultCode, str)

Raises

- **Exception if error occurs while invoking command on CSPSubarrayLeafNode, SDpSubarrayLeafNode or** –
- **DishLeafNode.** –

get_csp_subarray_obstate()

get_mccs_subarray_obstate()

get_sdp_subarray_obstate()

get_subarray_obstate(*dev_name*)

is_allowed(*raise_if_disallowed=True*)

is_allowed_low(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

is_allowed_mid(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model. :param raise_if_disallowed: whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

restart_csp()
set up csp devices

restart_dishes()

restart_sdp()
set up sdp devices

ska_tmc_subarraynode.commands.scan_command module

A class for TMC SubarrayNode's Scan() command

class ska_tmc_subarraynode.commands.scan_command.**Scan**(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.ScanCommand._name

A class for SubarrayNode's Scan() command.

The command accepts Scan id as an input and executes a scan on the subarray. Scan command is invoked on respective CSP and SDP subarray node for the provided interval of time. It checks whether the scan is already in progress. If yes it throws error showing duplication of command.

do(*argin=None*)

do_low(*argin*)
Method to invoke Scan command.

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

JSON string example as follows:

{“interface”:”https://schema.skao.int/ska-low-tmc-scan/2.0”,”transaction_id”:”txn-….-00001”,”scan_id”:1} Note: Above JSON string can be used as an input argument while invoking this command from JIVE.

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

rtype: (ReturnCode, str)

Raises `DevFailed if the command execution is not successful` –

`do_mid`(*argin*)
Method to invoke Scan command.

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

Example

{ “interface”: “https://schema.skao.intg/ska-tmc-scan/2.0”, “transaction_id”: “txn-….-00001”, “scan_id”: 1 }

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

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

rtype: (ReturnCode, str)

`is_allowed`(*raise_if_disallowed=True*)

`is_allowed_low`(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters `raise_if_disallowed` – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

`is_allowed_mid`(*raise_if_disallowed*)
Whether this command is allowed to run in the current state of the state model.

Parameters `raise_if_disallowed` – whether to raise an error or simply return False if the command is disallowed

Returns whether this command is allowed to run

Return type boolean

`scan_csp`(*argin*)
set up csp devices

`scan_mccs`(*argin*)
set up mccs devices

`scan_sdp`(*argin*)
set up sdp devices

`start_scan_timer`(*scan_duration*)

`update_mccs_json`(*input_scan*)
This Scan command input string is updated to send to MCCS SubarrayLeafNode.

ska_tmc_subarraynode.commands.off_command module

class ska_tmc_subarraynode.commands.off_command.**Off**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name

A class for Subarraynode's Off() command.

do_low(*argin=None*)
Method to invoke off command on the MCCS Subarray Leaf Node. param: None

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

rtype: (ResultCode, str)

do_mid(*argin=None*)
Method to invoke Off command on CSP and SDP Subarray Leaf Nodes. param: None

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

rtype: (ResultCode, str)

get_csp_subarray_obstate()

get_sdp_subarray_obstate()

get_subarray_obstate(*dev_name*)

ska_tmc_subarraynode.commands.on_command module

class ska_tmc_subarraynode.commands.on_command.**On**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name

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

do_low(*argin=None*)
Method to invoke On command on MccsSubarrayLeafNode

Raises Exception if the command execution is not successful –

do_mid(*argin=None*)
Method to invoke On command on CspSubarrayLeafNode and SdpSubarrayLeafNode.

Raises DevFailed if the command execution is not successful –

ska_tmc_subarraynode.commands.standby_command module

class ska_tmc_subarraynode.commands.standby_command.**Standby**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TMCCommand._name

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

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

do_mid(*argin=None*)
Method to invoke Standby command on CSPSubarrayLeafNode and SDPSubarrayLeafNode.

Raises Exception if the command execution is not successful –

get_command_object(*command_name*)

Module contents

ska_tmc_subarraynode.manager package

Submodules

ska_tmc_subarraynode.manager.aggregators module

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

Bases: ska_tmc_common.aggregators.ska_tmc_common.aggregators.Aggregator._name

aggregate()

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

Bases: ska_tmc_common.aggregators.ska_tmc_common.aggregators.Aggregator._name

aggregate()

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

Bases: ska_tmc_common.aggregators.ska_tmc_common.aggregators.Aggregator._name

aggregate()
Calculates aggregated observation state of Subarray.

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

Bases: ska_tmc_common.aggregators.ska_tmc_common.aggregators.Aggregator._name

aggregate()
Calculates aggregated observation state of Subarray.

subarray_node_obstate_not_aggregated()

ska_tmc_subarraynode.manager.event_receiver module

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

Bases: ska_tmc_common.event_receiver.ska_tmc_common.event_receiver.EventReceiver._name

The SubarrayNodeEventReceiver class has the responsibility to receive events from the sub devices managed by the Subarray node.

The ComponentManager uses the handle events methods for the attribute of interest. For each of them a callback is defined.

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

handle_pointing_state_event(*evt*)

handle_receive_addresses_event(*evt*)

subscribe_events(*dev_info*)

unscubscribe_dish_health(*dish_proxy*, *evt_id*)

unscubscribe_dish_leaf_health(*dish_leaf_proxy*, *evt_id*)

unscubscribe_dish_leaf_state(*dish_leaf_proxy*, *evt_id*)

unscubscribe_dish_pointing(*dish_proxy*, *evt_id*)

unscubscribe_dish_state(*dish_proxy*, *evt_id*)

unsubscribe_dish_events()

unsubscribe_dish_leaf_events()

ska_tmc_subarraynode.manager.subarray_node_component_manager module

This module provided a reference implementation of a BaseComponentManager.

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

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

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

A component manager for The Subarray Node component.

It supports:

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

adapter_error_message_result(*dev_name*, *e*)

add_device(*dev_name*)
Add device to the monitoring loop

Parameters **dev_name** (*str*) – device name

add_multiple_devices(*device_list*)
Add multiple devices to the monitoring loop

Parameters **device_list** – list of device names

property assigned_resources
Return the resources assigned to the component.

Returns the resources assigned to the component

Return type list of str

check_command_not_allowed_exception(*op_state*, *states_not_allowed*, *cmd_name*)

check_device_unresponsive_exception(*dev_name*)

check_if_no_dishes_available()

property checked_devices
Return the list of the checked monitored devices

Returns list of the checked monitored devices

property command_executed

property command_executor

property command_in_progress

device_failed(*device_info*, *exception*)
Set a device to failed and call the relative callback if available

Parameters

- **device_info** (*DeviceInfo*) – a device info
- **exception** – an exception

Type Exception

property devices
Return the list of the monitored devices

Returns list of the monitored devices

generate_command_result(*result_code*, *message*)

get_assigned_resources()

get_csp_subarray_dev_name()
Return Csp Subarray device name

get_dev_info(*dev_name*)

get_device(*dev_name*)
Return the device info our of the monitoring loop with name dev_name

Parameters **dev_name** (*str*) – name of the device

Returns a device info

Return type DeviceInfo

get_dish_dev_names()
Return the names of the dishes assigned to Subarray

get_dish_leaf_prefix()
Return the Dish Leaf Node prefix

get_mccs_subarray_dev_name()
Returns Mccs Subarray device name

get_sb_id()

get_scan_duration()

get_scan_id()

get_sdp_subarray_dev_name()
Return Sdp Subarray device name

get_subarray_healthstate()
Returns value of subarray's health state.

get_tm_leaf_csp_subarray_dev_name()
Return Csp Subarray Leaf Node device name

get_tm_leaf_dish_dev_names()
Return the names of the dish leaf nodes assigned to Subarray

init_adapters(*adapter_factory*)

init_adapters_low(*adapter_factory*)

init_adapters_mid(*adapter_factory*)

property input_parameter
Return the input parameter

Returns input parameter

Return type *InputParameter*

is_scan_timer_running()

property last_command_executed

remove_dish_devices()

reset()

reset_sb_id()

reset_scan_duration()

set_assigned_resources(*resources=[]*)
For SubarrayNode Mid, set assigned_resources with the list of dishes in argin. :param dev_name: name of the dish devices :type list: list[str]

set_dish_dev_names(*dish_dev_names*)
Update the names of the dishes assigned to Subarray

Parameters **dev_name** – name of the dish devices

set_sb_id(*sb_id*)

set_scan_duration(*scan_duration*)

set_scan_id(*scan_id*)

set_tmc_leaf_dish_dev_names(*tm_leaf_dish_dev_names*)
Update the names of the dish leaf nodes assigned to Subarray

Parameters **dev_name** – name of the dish devices

stop()

stop_scan_timer()

unsubscribe_dish_events()

update_assigned_resources_low()

update_device_health_state(*dev_name*, *health_state*)
Update a monitored device health state aggregate the health states available

Parameters

- **dev_name** (*str*) – name of the device
- **health_state** (*HealthState*) – health state of the device

update_device_info(*device_info*)
Update a device with correct monitoring information and call the relative callback if available

Parameters **device_info** (*DeviceInfo*) – a device info

update_device_obs_state(*dev_name*, *obs_state*)
Update a monitored device obs state, and call the relative callbacks if available

Parameters

- **dev_name** (*str*) – name of the device

- **obs_state** (*ObsState*) – obs state of the device

update_device_pointing_state(*dev_name*, *pointing_state*)
Update a monitored device pointing state aggregate the Subarray obs states and Dish pointing states

Parameters

- **dev_name** (*str*) – name of the device
- **pointing_state** (PointingState) – pointing state of the device

update_device_state(*dev_name*, *state*)
Update a monitored device state, aggregate the states available and call the relative callbacks if available

Parameters

- **dev_name** (*str*) – name of the device
- **state** (*DevState*) – state of the device

update_event_failure(*dev_name*)

update_input_parameter()

update_receive_addresses(*dev_name*, *receive_addresses*)
Update recesiveAddresses for a monitored device

Parameters

- **dev_name** (*str*) – name of the device
- **receive_addresses** (*str*) – recesiveAddresses

ska_tmc_subarraynode.manager.monitoring_loop module

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

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

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

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

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

create_device_info(*dev_info*, *proxy*)

device_task(*dev_info*)

Module contents

ska_tmc_subarraynode.model package

Submodules

ska_tmc_subarraynode.model.component module

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

A component class for Subarray Node

It supports:

- Maintaining a connection to its component
- Monitoring its component

property **assigned_resources**
Return the resources assigned to the component.

Returns the resources assigned to the component

Return type list of str

property **devices**
Return the monitored devices.

Returns the monitored devices

Return type DeviceInfo[]

get_device(*dev_name*)
Return the monitored device info by name.

Parameters **dev_name** – name of the device

Returns the monitored device info

Return type DeviceInfo

remove_device(*dev_name*)
Remove a device from the list

Parameters **dev_name** – name of the device

property **sb_id**
Return the Sb_id

Returns the Sb_id

Return type str

property **scan_duration**
Return the duration of scan

Returns the scan duration

Return type int

property **scan_id**
Return the Scan id

Returns the Scan id

Return type str

set_obs_callbacks(*_update_assigned_resources_callback=None*)

set_op_callbacks(*_update_device_callback=None*, *_update_subarray_health_state_callback=None*)

property subarray_health_state
Return the aggregated subarray health state

Returns the subarray health state

Return type HealthState

to_dict()

update_device(*dev_info*)
Update (or add if missing) Device Information into the list of the component.

Parameters **dev_info** – a DeviceInfo object

update_device_exception(*dev_info*, *exception*)
Update (or add if missing) Device Information into the list of the component.

Parameters **dev_info** – a DeviceInfo object

ska_tmc_subarraynode.model.enum module

class ska_tmc_subarraynode.model.enum.**PointingState**(*value*)
Bases: enum.IntEnum

An enumeration.

NONE = 0

READY = 1

SCAN = 4

SLEW = 2

TRACK = 3

UNKNOWN = 5

ska_tmc_subarraynode.model.input module

class ska_tmc_subarraynode.model.input.**InputParameter**(*changed_callback*)
Bases: object

update(*component_manager*)

class ska_tmc_subarraynode.model.input.**InputParameterLow**(*changed_callback*)
Bases: *ska_tmc_subarraynode.model.input.InputParameter*

property mccs_subarray_dev_name
Return the MCCS Subarray device name

Returns the MCCS Subarray device name

Return type str

property tm_leaf_mccs_subarray_dev_name
Return the TM Leaf MCCS Subarray device name

Returns the TM Leaf MCCS Subarray device name

Return type str

update(*component_manager*)

class ska_tmc_subarraynode.model.input.**InputParameterMid**(*changed_callback*)
Bases: *ska_tmc_subarraynode.model.input.InputParameter*

property csp_subarray_dev_name
Return the CSP Subarray device name

Returns the CSP Subarray device name

Return type str

property dish_dev_names
Return the dish device names

Returns the TM dish device names

Return type list

property sdp_subarray_dev_name
Returns the SDP Subarray device name

Returns the SDP Subarray device name

Return type str

property tm_leaf_csp_subarray_dev_name
Return the CSP Subarray Leaf Node device names

Returns the CSP Subarray Leaf Node device names

Return type str

property tm_leaf_dish_dev_names
Return the TM dish device names

Returns the TM dish device names

Return type list

property tm_leaf_sdp_subarray_dev_name
Return the SDP Subarray Leaf Node device names

Returns the SDP Subarray Leaf Node device names

Return type str

update(*component_manager*)

Module contents

3.1.2 Submodules

3.1.3 ska_tmc_subarraynode._subarraynode_node module

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

class ska_tmc_subarraynode.subarray_node.**AbstractSubarrayNode**(**args: Any*, ***kwargs: Any*)

Bases: ska_tango_base.ska_tango_base.SKASubarray._name, ska_tmc_common.tmc_base_device.ska_tmc_common.tmc_base_device.TMCBaseDevice._name

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

AbstractSubarrayNode class is inherited from SKASubarray class and TMCBaseDevice class. TMCBaseDevice class is further inherited from SKABaseDevice class.

Common attributes and device_properties within TMC nodes are getting inherited from TMCBaseDevice.

Device Attributes

scanID: ID of ongoing SCAN

sbID: ID of ongoing Scheduling Block

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

Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.InitCommand._name

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

do()

Initializes the attributes and properties of the Subarray Node.

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

rtype: (ReturnCode, str)

Raises DevFailed if the error while subscribing the tango attribute –

Off()

Invokes Off command on SubarrayNode

On()

Invokes On command on SubarrayNode

Standby()

Invokes Standby command on SubarrayNode

always_executed_hook()

Internal construct of TANGO.

create_component_manager()

delete_device()

enqueue_command(*command_name*, *args=None*)

init_command_objects()

Initialises the command handlers for commands supported by this device.

internalModel_read()

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

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

Return type boolean

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

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

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

Return type boolean

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

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

Returns True if the command is allowed

Return type boolean

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

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

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

Returns True if the command is allowed

Return type boolean

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

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

Return type boolean

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

read_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.

transformedInternalModel_read()

update_assigned_resources_callback(*assigned_resources*)

update_command_in_progress_callback(*command_in_progress*)

update_device_callback(*dev_info*)

update_subarray_health_state_callback(*subarray_health_state*)

3.1.4 ska_tmc_subarraynode._subarraynode_node_low module

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

class ska_tmc_subarraynode.subarray_node_low.**SubarrayNodeLow**(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.ska_tango_base.SKASubarray._name, ska_tmc_common.tmc_base_device.ska_tmc_common.tmc_base_device.TMCBaseDevice._name

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

Device Properties

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

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

Device Attributes

class InitCommand(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.InitCommand._name

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

do()
Initializes the attributes and properties of the Subarray Node Mid.
Returns A tuple containing a return code and a string message indicating status. The message is for information purpose only.

rtype: (ReturnCode, str)

Raises DevFailed if the error while subscribing the tango attribute –

create_component_manager()

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

read_mccsSubarrayDevName()
Return the mccsSubarrayDevName attribute.

read_tmcLeafMccsSubarrayDevName()
Return the tmcLeafMccsSubarrayDevName attribute.

write_mccsSubarrayDevName(*value*)
Set the mccsSubarrayDevName attribute.

write_tmcLeafMccsSubarrayDevName(*value*)
Set the tmcLeafMccsSubarrayDevName attribute.

ska_tmc_subarraynode.subarray_node_low.main(*args=None*, ***kwargs*)
Runs the SubarrayNodeLow. :param args: Arguments internal to TANGO :param kwargs: Arguments internal to TANGO :return: SubarrayNodeLow TANGO object.

3.1.5 ska_tmc_subarraynode._subarraynode_node_mid module

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

class ska_tmc_subarraynode.subarray_node_mid.SubarrayNodeMid(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.ska_tango_base.SKASubarray._name, ska_tmc_common.tmc_base_device.ska_tmc_common.tmc_base_device.TMCBaseDevice._name

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

Device Properties

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

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

DishLeafNodePrefix: Device name prefix for the Dish Leaf Node.

CspSubarrayFQDN: FQDN of the CSP Subarray Tango Device Server.

SdpSubarrayFQDN: FQDN of the CSP Subarray Tango Device Server.

Device Attributes

class InitCommand(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.SKASubarray.ska_tango_base.SKASubarray.InitCommand._name

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

do()
Initializes the attributes and properties of the Subarray Node Mid.
Returns A tuple containing a return code and a string message indicating status. The message is for information purpose only.

rtype: (ReturnCode, str)

Raises DevFailed if the error while subscribing the tango attribute –

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

read_cspSubarrayDevName()
Return the cspsubarraydevname attribute.

read_dishDevNames()
Return the dishdevnames attribute.

read_sdpSubarrayDevName()
Return the sdpsubarraydevname attribute.

read_tmcLeafCspSubarrayDevName()
Return the tmcLeafCspSubarrayDevName attribute.

read_tmcLeafDishDevNames()
Return the tmcLeafDishDevNames attribute.

read_tmcLeafSdpSubarrayDevName()
Return the tmcLeafSdpSubarrayDevName attribute.

write_cspSubarrayDevName(*value*)
Set the cspsubarraydevname attribute.

write_dishDevNames(*value*)
Set the dishdevnames attribute.

write_sdpSubarrayDevName(*value*)
Set the sdpsubarraydevname attribute.

write_tmcLeafCspSubarrayDevName(*value*)
Set the tmcLeafCspSubarrayDevName attribute.

write_tmcLeafDishDevNames(*value*)
Set the tmcLeafDishDevNames attribute.

write_tmcLeafSdpSubarrayDevName(*value*)
Set the tmcLeafSdpSubarrayDevName attribute.

ska_tmc_subarraynode.subarray_node_mid.main(*args=None*, ***kwargs*)
Runs the SubarrayNode. :param args: Arguments internal to TANGO :param kwargs: Arguments internal to TANGO :return: SubarrayNode TANGO object.

3.1.6 ska_tmc_subarraynode.exceptions module

exception ska_tmc_subarraynode.exceptions.**CommandNotAllowed**
Bases: Exception

Raised when a command is not allowed.

exception ska_tmc_subarraynode.exceptions.**DeviceUnresponsive**
Bases: Exception

Raised when a device is not responsive.

exception ska_tmc_subarraynode.exceptions.**InvalidObsStateError**
Bases: ValueError

Raised when subarray is not in required obsState.

3.1.7 ska_tmc_subarraynode.release module

Release information for Python Package

3.1.8 ska_tmc_subarraynode.transaction_id module

ska_tmc_subarraynode.transaction_id.**identify_with_id**(*name: str*, *arg_name: str*)

ska_tmc_subarraynode.transaction_id.**inject_id**(*obj*, *data: Dict*) → Dict

ska_tmc_subarraynode.transaction_id.**inject_with_id**(*arg_position: int*, *arg_name: str*)

ska_tmc_subarraynode.transaction_id.**update_with_id**(*obj*, *parameters: Any*) → Union[Dict, str]

3.1.9 Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

D

E

G

H

I

L

M

N

O

P

R

S

T

U

W