Subarray Node Documentation

Release 1.0

NCRA India

Jul 28, 2023

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

Abstract command class for SubarrayNode, a subclass of TMCCommand. This class provides methods to initialize adapters and executing commands on subarray devices.

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

Bases: TMCCommand

This code defines the SubarrayNodeCommand class, which is a subclass of TMCCommand. This class provides methods for initializing adapters and executing commands on subarray devices.

adapter_error_message_result(*dev_name: str*, *exception*) → Tuple[ska_tango_base.commands.ResultCode, str]

Returns ResultCode.FAILED with failure message as a tuple

do(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

get_adapter_by_device_name(*device_name: str*) → ska_tmc_common.adapters.AdapterFactory

The get_adapter_by_device_name method takes a device_name as
input and searches for an adapter object in the adapter_factory object's adapters attribute that matches the input device_name. If a matching adapter object is found, it is returned. If no matching adapter object is found, None is returned.

Args: device_name (str): The name of the device to search for.

Returns: An adapter object if a matching device is found, otherwise None.

get_dish_adapter_by_device_name(*device_name_list: List[str]*) → List[ska_tmc_common.adapters.AdapterFactory]

The get_dish_adapter_by_device_name method takes a list of

device_name_list as input and searches for adapter objects in

the adapter_factory object's adapters attribute that match any of the device names in the input list. It returns a list of matching adapter objects.

Args: device_name_list (list): A list of device names to search for.

Returns: A list of adapter objects that match the device names in the input list. If no matching adapter object is found, an empty list is returned.

init_adapters() → Tuple[ska_tango_base.commands.ResultCode, str]

init_adapters_low() → Tuple[ska_tango_base.commands.ResultCode, str]

init_adapters_mid() → Tuple[ska_tango_base.commands.ResultCode, str]

reject_command(*message: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to return task status as TaskStatus.REJECTED along with the error message

update_command_in_progress(*command_name: str*) → None

The update_command_in_progress method updates the command_in_progress
and command_in_progress_id attributes of the component_manager object with the given command_name. It also logs information about the updated command.

Args: command_name (str): The name of the command being updated.

Returns: None

update_task_status(*result: ska_tango_base.commands.ResultCode*, *message: str = ''*) → None

Method to update task status with result code and exception message if any.

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: *SubarrayNodeCommand*

A class for SubarrayNode's AssignResources() command.

Assigns resources to CSP, SDP via respective subarray leaf nodes. It also sets AssignedResources attribute on SubarrayNode.

assign_csp_resources(*json_argument: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

This function accepts the AssignResources input JSON and invokes the assign resources command on the CSP Subarray Leaf Node.

Parameters
json_argument – AssignResources input JSON string without SDP block

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

assign_low_csp_resources(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

This function accepts the CSP Resources as input and assigns CSP resources to CSP Subarray through CSP Subarray Leaf Node.

Parameters
argin – JSON string including CSP resources.

Returns
A tuple containing ResultCode and string.

assign_sdp_resources(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

This function accepts SDP block from input AssignResources JSON and assigns SDP resources to SDP Subarray through SDP Subarray Leaf Node.

Parameters
argin – List of strings JSON string containing only SDP resources.

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

clear_resources()

Method for clearing resources in case of failure

do_low(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke AssignResources command on subarraynode low.

Parameters
argin – DevString.

Example:

{“interface”: “https://schema.skao.int/ska-tmc-assignresources/2.1”, “transaction_id”:”txn-….-00001”,”subarray_id”: 1,”mccs”:{“subarray_beam_ids”:

[1],”station_ids”: [[1,2]],”channel_blocks”: [3]},”sdp”:{“interface”:

“https://schema.skao.int/ska-sdp-assignres/0.4”,”execution_block”: {“eb_id”: “eb-mvp01-20200325-00001”,”max_length”: 100,”context”:{},”beams”:[{“beam_id”: “vis0”, “function”:”visibilities”},{“beam_id”: “pss1”,”search_beam_id”: 1,”function”: “pulsar search”}, {“beam_id”: “pss2”,”search_beam_id”: 2,”function”:”pulsar search”},{“beam_id”: “pst1”, “timing_beam_id”: 1,”function”: “pulsar timing”},{“beam_id”:”pst2”,”timing_beam_id”:2, “function”: “pulsar timing”},{“beam_id”: “vlbi1”,”vlbi_beam_id”:1,”function”: “vlbi”}], “channels”: [{“channels_id”:”vis_channels”,”spectral_windows”:[{“spectral_window_id”: “fsp_1_channels”,”count”: 744,”start”: 0,”stride”: 2,”freq_min”: 350000000,”freq_max”: 368000000,”link_map”: [[0,0],[200,1],[744,2],[944,3]]},{“spectral_window_id”:”fsp_2_channels”, “count”: 744,”start”: 2000,”stride”: 1,”freq_min”: 360000000,”freq_max”:368000000, “link_map”: [[2000,4],[2200,5]]},{“spectral_window_id”: “zoom_window_1”,”count”: 744,”start”: 4000,”stride”: 1,”freq_min”: 360000000,”freq_max”: 361000000,”link_map”: [[4000,6],[4200,7]]}]}, {“channels_id”:”pulsar_channels”,”spectral_windows”:[{“spectral_window_id”: “pulsar_fsp_channels”,”count”: 744,”start”: 0,”freq_min”: 350000000,”freq_max”: 368000000}]}], “polarisations”: [{“polarisations_id”: “all”,”corr_type”:[“XX”,”XY”,”YY”,”YX”]}],”fields”: [{“field_id”: “field_a”,”phase_dir”:{“ra”:[123,0.1],”dec”:[123,0.1],”reference_time”: “…”, “reference_frame”: “ICRF3”},”pointing_fqdn”: “low-tmc/telstate/0/pointing”}]}, “processing_blocks”:[{“pb_id”: “pb-mvp01-20200325-00001”,”sbi_ids”:[“sbi-mvp01-20200325-00001”], “script”:{},”parameters”:{},”dependencies”:{}},{“pb_id”: “pb-mvp01-20200325-00002”,”sbi_ids”: [“sbi-mvp01-20200325-00002”],”script”:{},”parameters”:{},”dependencies”:{}},{“pb_id”: “pb-mvp01-20200325-00003”,”sbi_ids”:[“sbi-mvp01-20200325-00001”,”sbi-mvp01-20200325-00002”], “script”:{},”parameters”: {},”dependencies”:{}}],”resources”:{“csp_links”:[1,2,3,4], “receptors”:[“FS4”,”FS8”],”receive_nodes”:10}}”csp”:{“interface”:”https://schema.skao.int/ska-low-csp-assignresources/2.0”,”common”:{“subarray_id”:1},”lowcbf”:{“resources”: [{“device”:”fsp_01”,”shared”:true,”fw_image”:”pst”,”fw_mode”:”unused”},{“device”:”p4_01”, “shared”:true,”fw_image”:”p4.bin”,”fw_mode”:”p4”}]}}}

Returns
A tuple containing ResultCode and string.

rtype:
(ResultCode, str)

Raises

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

do_mid(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke AssignResources command on subarraynode mid.

Parameters
argin – DevString.

Example:

{“interface”: “https://schema.skao.int/ska-tmc-assignresources/2.1”,”transaction_id”: “txn-....-00001”,”subarray_id”: 1,”dish”: {“receptor_ids”: [“SKA001”]},”sdp”:{“interface”: “https://schema.skao.int/ska-sdp-assignres/0.4”,”execution_block”: {“eb_id”: “eb-mvp01-20200325-00001”,”max_length”: 100,”context”:{},”beams”:[{“beam_id”: “vis0”, “function”:”visibilities”},{“beam_id”: “pss1”,”search_beam_id”: 1,”function”: “pulsar search”}, {“beam_id”: “pss2”,”search_beam_id”: 2,”function”:”pulsar search”},{“beam_id”: “pst1”, “timing_beam_id”: 1,”function”: “pulsar timing”},{“beam_id”:”pst2”,”timing_beam_id”:2, “function”: “pulsar timing”},{“beam_id”: “vlbi1”,”vlbi_beam_id”:1,”function”: “vlbi”}], “channels”: [{“channels_id”:”vis_channels”,”spectral_windows”:[{“spectral_window_id”: “fsp_1_channels”,”count”: 744,”start”: 0,”stride”: 2,”freq_min”: 350000000,”freq_max”: 368000000,”link_map”: [[0,0],[200,1],[744,2],[944,3]]},{“spectral_window_id”:”fsp_2_channels”, “count”: 744,”start”: 2000,”stride”: 1,”freq_min”: 360000000,”freq_max”:368000000, “link_map”: [[2000,4],[2200,5]]},{“spectral_window_id”: “zoom_window_1”,”count”: 744,”start”: 4000,”stride”: 1,”freq_min”: 360000000,”freq_max”: 361000000,”link_map”: [[4000,6],[4200,7]]}]}, {“channels_id”:”pulsar_channels”,”spectral_windows”:[{“spectral_window_id”: “pulsar_fsp_channels”,”count”: 744,”start”: 0,”freq_min”: 350000000,”freq_max”: 368000000}]}], “polarisations”: [{“polarisations_id”: “all”,”corr_type”:[“XX”,”XY”,”YY”,”YX”]}],”fields”: [{“field_id”: “field_a”,”phase_dir”:{“ra”:[123,0.1],”dec”:[123,0.1],”reference_time”: “...”, “reference_frame”: “ICRF3”},”pointing_fqdn”: “low-tmc/telstate/0/pointing”}]}, “processing_blocks”:[{“pb_id”: “pb-mvp01-20200325-00001”,”sbi_ids”:[“sbi-mvp01-20200325-00001”], “script”:{},”parameters”:{},”dependencies”:{}},{“pb_id”: “pb-mvp01-20200325-00002”,”sbi_ids”: [“sbi-mvp01-20200325-00002”],”script”:{},”parameters”:{},”dependencies”:{}},{“pb_id”: “pb-mvp01-20200325-00003”,”sbi_ids”:[“sbi-mvp01-20200325-00001”,”sbi-mvp01-20200325-00002”], “script”:{},”parameters”: {},”dependencies”:{}}],”resources”:{“csp_links”:[1,2,3,4], “receptors”:[“FS4”,”FS8”],”receive_nodes”:10}}}

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

rtype:
(ResultCode, str)

Raises

- **KeyError if JSON parsing failed** –
- **Exception if the command execution is not successful** –

invoke_assign_resources(*argin: Optional[str] = None*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*) → None

This is a long running method for AssignResources command, it executes do hook, invokes AssignResources command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **logger** (*logging.Logger*) – logger
- **logger** – argin
- **argin** (*logging.Logger*) – JSON string consisting of the resources to be assigned
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

set_low_assigned_resources(*argin*)

Set assigned resources to low telescope

set_up_dish_data(*receptor_ids: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

Creates dish leaf node and dish device FQDNs using input receptor ids. The healthState and pointintgState attributes of all all the dishes are subscribed.

Parameters

receptor_ids – List of receptor IDs to be allocated to subarray. Example: [‘SKA001’, ‘SKA002’]

Returns

List of Resources added to the Subarray. Example: [‘SKA001’, ‘SKA002’]

update_task_status(*result: ska_tango_base.commands.ResultCode*, *message: str = ''*) → None

Method to update task status with result code and exception message if any.

validate_low_json(*argin: <module 'json' from '/home/docs/.pyenv/versions/3.7.9/lib/python3.7/json/__init__.py'>*)

Method to validate low input jsons for AssignResources command

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: *SubarrayNodeCommand*

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: dict*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to check only dish configuration

configure_low_csp(*scan_config: dict*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to configure low CSP

do_low(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

This method executes the Configure workflow of the Subarray Node Low. It will invoke Configure command on the CSP Subarray Leaf Node, SDP Subarray Leaf Node

Parameters

argin – DevString.

JSON string example is:

{“interface”:”https://schema.skao.int/ska-low-tmc-configure/3.0”,”transaction_id”: “txn-....-00001”,”mccs”:{“stations”:[{“station_id”:1}, {“station_id”:2}],”subarray_beams”: [{“subarray_beam_id”:1,”station_ids”:[1,2],”update_rate”:0.0,”channels”:[[0,8,1,1], [8,8,2,1],[24,16,2,1]],”antenna_weights”:[1.0,1.0,1.0],”phase_centre”:[0.0,0.0],”target”: {“reference_frame”:”HORIZON”,”target_name”:”DriftScan”,”az”:180.0,”el”:45.0}}]},”sdp”: {“interface”:”https://schema.skao.int/ska-sdp-configure/0.4”,”scan_type”:”science_A”},”csp”: {“interface”:”https://schema.skao.int/ska-csp-configure/2.0”,”subarray”:{“subarray_name”: “science period23”},”common”:{“config_id”:”sbi-mvp01-20200325-00001-science_A” },”lowcbf”:{“stations”:{“stns”:[[1,0],[2,0],[3,0],[4,0]],”stn_beams”: [{“beam_id”:1,”freq_ids”:[64,65,66,67,68,68,70,71],”boresight_dly_poly”:”url”}]}, “timing_beams”:{“beams”:[{“pst_beam_id”:13,”stn_beam_id”:1,”offset_dly_poly”:”url”, “stn_weights”:[0.9,1.0,1.0,0.9],”jones”:”url”,”dest_ip”:[“10.22.0.1:2345”,”10.22.0.3:3456”] ,”dest_chans”:[128,256],”rfi_enable”:[true,true,true],”rfi_static_chans”:[1,206,997], “rfi_dynamic_chans”:[242,1342],”rfi_weighted”:0.87}]}}} ,”tmc”:{“scan_duration”:10.0}}

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: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

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)

invoke_configure(*argin: Optional[str] = None*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*) → None

This is a long running method for Configure command, it executes do hook, invokes Configure command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **argin** (*Json string, defaults to None*) – JSON string consisting of the resources to be configured
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

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

Bases: object

A class representing data for an element device.

static build_up_csp_cmd_data(*scan_config: dict*, *delay_model_subscription: Tuple*, *receive_addresses_map: str*, *component_manager*) → Tuple[ska_tango_base.commands.ResultCode, str]

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: dict*, *component_manager*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to build up dish command data

static build_up_sdp_cmd_data(*scan_config: dict*, *component_manager*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to build up sdp command data

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: *SubarrayNodeCommand*

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_low(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

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(*argin=None*) → Tuple[ska_tango_base.commands.ResultCode, str]

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() → Tuple[ska_tango_base.commands.ResultCode, str]

End command on CSP Subarray Leaf Node

end_sdp() → Tuple[ska_tango_base.commands.ResultCode, str]

End command on SDP Subarray Leaf Node

invoke_end(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*) → None

This is a long running method for End command, it executes do hook, invokes End Command SdpSubarrayleafnode.

Parameters

- **logger** (*logging.Logger*) – logger
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

stop_dish_tracking() → Tuple[ska_tango_base.commands.ResultCode, str]

Method to stop dish tracking

update_task_status() → None

updating the task status

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: *SubarrayNodeCommand*

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_low(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

This method executes the End Scan workflow of the Subarray Node Low. It will invoke End Scan command on the CSP Subarray Leaf Node, SDP Subarray Leaf Node.

Returns
None

Raises
DevFailed if the command execution is not successful. –

do_mid(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke Endscan command.

Returns
None

Raises
The command execution is not successful. –

end_scan() → None

Method for setting device for Subarray Mid

end_scan_low() → Tuple[ska_tango_base.commands.ResultCode, str]

setting up device for low sdp and csp

end_scan_mid() → Tuple[ska_tango_base.commands.ResultCode, str]

Method for setting up device for mid SDP and CSP

endscan_csp() → Tuple[ska_tango_base.commands.ResultCode, str]

set up csp devices

endscan_sdp() → Tuple[ska_tango_base.commands.ResultCode, str]

set up sdp devices

invoke_end_scan(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*)

This is a long running method for Scan command, it executes do hook, invokes EndScan command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **logger** (*logging.Logger*) – logger
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

update_task_status()

Method for implementing for updating task status

ska_tmc_subarraynode.commands.obsreset_command module

ObsReset Command for SubarrayNode.

class ska_tmc_subarraynode.commands.obsreset_command.**ObsReset**(**args: Any*, ***kwargs: Any*)

Bases: ObsResetCommand

A class for SubarrayNode's ObsReset() command.

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

completed() → None

Placeholder method that indicates a task has completed.

do(*argin: Optional[str] = None*) → Tuple

do method of obsreset command

do_low() → Tuple[ska_tango_base.commands.ResultCode, str]

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() → Tuple[ska_tango_base.commands.ResultCode, str]

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() → None

Returns csp subarray obsstate

get_sdp_subarray_obsstate() → None

Returns sdp subarray obsstate

is_allowed(*raise_if_disallowed: bool = True*) → bool

check if component manager is of instance of mid or low

is_allowed_low(*raise_if_disallowed: bool*) → bool

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: bool*) → bool

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() → Tuple[ska_tango_base.commands.ResultCode, str]

Invoke ObsReset command on CSP Subarray Leaf Node.

obsreset_sdp() → Tuple[ska_tango_base.commands.ResultCode, str]

Invoke ObsReset command on SDP Subarray Leaf Node.

obsrest_dish() → Tuple[ska_tango_base.commands.ResultCode, str]

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: *SubarrayNodeCommand*

A class for TMC SubarrayNode'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_low_resources() → Tuple[ska_tango_base.commands.ResultCode, str]

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

clean_up_mid_resources() → Tuple[ska_tango_base.commands.ResultCode, str]

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

do_low(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

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(*argin=None*)

Method to invoke ReleaseAllResources command.

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

rtype:
(ResultCode, str)

invoke_release_resources(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*)

This is a long running method for ReleaseAllResources command, it executes do hook, invokes ReleaseAllResources command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **logger** (*logging.Logger*) – logger
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

release_csp_resources() → Tuple[ska_tango_base.commands.ResultCode, str]

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

Parameters
argin – DevVoid

Returns
DevVoid

release_sdp_resources() → Tuple[ska_tango_base.commands.ResultCode, str]

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

Parameters
argin – DevVoid

Returns
DevVoid

update_task_status(*result: ska_tango_base.commands.ResultCode*, *message: str = ''*) → None

Method to update task status with result code and exception message if any.

ska_tmc_subarraynode.commands.reset_command module

Reset Command for SubarrayNode.

class ska_tmc_subarraynode.commands.reset_command.**Reset**(**args: Any*, ***kwargs: Any*)

Bases: ResetCommand

A class for SubarrayNode's Reset() command.

do(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

do method for reset command.

do_low(*argin=None*) → Tuple[ska_tango_base.commands.ResultCode, str]

" 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: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

" 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: bool = True*) → True

check if component manager is of instance of mid or low

is_allowed_low(*raise_if_disallowed: bool*) → bool

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: bool*) → bool

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: *SubarrayNodeCommand*

A class for SubarrayNode's Restart() command.

clean_up_configuration() → Tuple[ska_tango_base.commands.ResultCode, str]

Restart command is a like a hard reset. So, the transition should ensure the removal of all resources from the subarray.

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
Tuple[ResultCode,str]

clean_up_dishes() → Tuple[ska_tango_base.commands.ResultCode, str]

Removes group of dishes from tango group client

do_low(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

This command on Subarray Node Low invokes Restart command on CSP Subarray Leaf Node and SDP 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)

do_mid(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

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_sln_subarray_obstate() → ska_tango_base.control_model.ObsState
Return obstate of csp subarray obstate

get_mccs_subarray_obstate() → ska_tango_base.control_model.ObsState
Return obstate of mccs subarray obstate

get_sdp_sln_subarray_obstate()
Return obstate of sdp subarray obstate

get_subarray_leaf_node_obstate(*dev_name: str*) → ska_tango_base.control_model.ObsState
Return obstate of subarray obstate

invoke_restart(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*) → None
This is a long running method for Restart command, it executes do hook, invokes Restart Command

Parameters

- **logger** (*logging.Logger*) – logger
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

restart_csp() → Tuple[ska_tango_base.commands.ResultCode, str]
set up csp devices

restart_dishes() → Tuple[ska_tango_base.commands.ResultCode, str]
Restarts all dish leaf nodes in the component manager.

restart_sdp() → Tuple[ska_tango_base.commands.ResultCode, str]
set up sdp devices

update_task_status()
Method for implementing for updating task status

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: *SubarrayNodeCommand*

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_low(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]
Method to invoke Scan command.

Parameters
argin – DevString. JSON string containing id.

JSON string example as follows: TODO : Update once json schema is confirmed {"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: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

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 WEBJIVE.

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

rtype: (ReturnCode, str)

invoke_scan(*argin: Optional[str] = None*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*) → None

This is a long running method for Scan command, it executes do hook, invokes Scan command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **argin** (*Json string, defaults to None*) – JSON string consisting of the resources to be Scan
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

scan_csp(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

set up csp devices

scan_csp_low(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

set up csp devices

scan_sdp(*argin: str*) → Tuple[ska_tango_base.commands.ResultCode, str]

set up sdp devices

start_scan_timer(*scan_duration: int*) → None

Method for starting scan timer

update_task_status()

Method for implementing for updating task status

ska_tmc_subarraynode.commands.off_command module

Off Command for SubarrayNode

class ska_tmc_subarraynode.commands.off_command.**Off**(**args: Any*, ***kwargs: Any*)

Bases: *SubarrayNodeCommand*

A class for Subarraynode's Off() command.

do_low(*argin=None*)

Method to invoke off command on the MCCS Subarray Leaf Node.

Parameters
argin – Input json for Command, defaults to None

:type None

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

rtype: (ResultCode, str)

do_mid(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke Off command on SDP Subarray Leaf Nodes. :param argin: Input json for Command, defaults to None :type None

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

rtype: (ResultCode, str)

get_csp_subarray_obstate()

Return obstate of csp subarray obstate

get_sdp_subarray_obstate()

Return obstate of sdp subarray obstate

get_subarray_obstate(*dev_name*)

Return obstate of subarray obstate

subarray_off(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*)

"This is a long running method for Off command, it executes do hook, invokes Off command on SdpSubarrayleafnode. :param logger: logger :type logger: logging.Logger :param task_callback: Update task state, defaults to None :type task_callback: Callable, optional :param task_abort_event: Check for abort, defaults to None :type task_abort_event: Event, optional

update_task_status()

Method for implementing for updating task status

ska_tmc_subarraynode.commands.on_command module

On Command for SubarrayNode

class ska_tmc_subarraynode.commands.on_command.**On**(**args: Any*, ***kwargs: Any*)

Bases: *SubarrayNodeCommand*

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

do_low(*argin=None*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke On command on MccsSubarrayLeafNode, Low CspSubarrayLeafNode and Low SdpSubarrayLeafNode.

Parameters
argin – Input json for Command, defaults to None

:type None

return: A tuple containing a return code and a string message indicating status. rtype: (ResultCode, str)

Raises
Exception if the command execution is not successful –

do_mid(*argin: Optional[str] = None*) → Tuple[ska_tango_base.commands.ResultCode, str]

Method to invoke On command on Mid CspSubarrayLeafNode and SdpSubarrayLeafNode.

Parameters
argin – Input json for Command, defaults to None

:type None

return: A tuple containing a return code and a string message indicating status. rtype: (ResultCode, str)

Raises
DevFailed if the command execution is not successful –

on_leaf_nodes(*logger*, *task_callback: Optional[Callable] = None*, *task_abort_event: Optional[Event] = None*)

This is a long running method for On command, it executes do hook, invokes On command on CspSubarrayleafnode and SdpSubarrayleafnode.

Parameters

- **logger** (*logging.Logger*) – logger
- **task_callback** (*Callable, optional*) – Update task state, defaults to None
- **task_abort_event** (*Event, optional*) – Check for abort, defaults to None

update_task_status() → None

Method for implementing for updating task status

ska_tmc_subarraynode.commands.standby_command module

Module contents

ska_tmc_subarraynode.manager package

Submodules

ska_tmc_subarraynode.manager.aggregators module

This Module is for Aggregating the HealthState and
ObsStates of Devices

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

Bases: Aggregator

The HealthStateAggregatorLow class is an implementation of an aggregator that is responsible for aggregating the health states of various devices in the low telescope

aggregate() → IntEnum

This method aggregating HealthState in Low telescope

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

Bases: Aggregator

The HealthStateAggregatorMid class is an implementation of an aggregator that is responsible for aggregating the health states of various devices in the mid telescope

aggregate() → IntEnum

This method aggregating HealthState in Mid telescope

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

Bases: Aggregator

The ObsStateAggregatorLow class is a subclass of the Aggregator class and is responsible for aggregating and managing the observation states of the subarrays

aggregate()

Calculates aggregated observation state of Subarray.

subarray_node_obstate_not_aggregated()

the subarray's obsState is not yet aggregated due to missing events.

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

Bases: Aggregator

The ObsStateAggregatorMid class is a subclass of the Aggregator class and is responsible for aggregating and managing the observation states of the subarrays

aggregate()

Calculates aggregated observation state of Subarray.

subarray_node_obstate_not_aggregated()

the subarray's obsState is not yet aggregated due to missing events.

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

Bases: Aggregator

class to aggregate tmc low subarray device availability depending on tmc low leaf nodes

aggregate()

Aggregates the subarray availability

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

Bases: Aggregator

class to aggregate tmc mid subarray device availability depending on tmc mid leaf nodes

aggregate()

Aggregates the subarray availability

ska_tmc_subarraynode.manager.event_receiver module

This Module is used to Receive events from devices

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

Bases: EventReceiver

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?

check_event_error(*evt*)

Method for checking event error.

handle_assigned_resources_event(*evt*)

Method for handling assigned resources events.

handle_csp_leaf_obs_state_event(*evt*)

Handle the CSP Leaf Node obsState event.

Parameters
evt (*tango.ChangeEvent*) – Event data

handle_lrcr_event(*event*)

Method for handling longRunningCommandResult events.

handle_pointing_state_event(*evt*)

Method for handling pointing state event.

handle_receive_addresses_event(*evt*)

Method for handling and receiving addresses events.

handle_sdp_leaf_obs_state_event(*evt*)

Handle the SDP Leaf Node obsState event.

Parameters
evt (*tango.ChangeEvent*) – Event data

handle_subsystem_availability_event(*event_data*)

Method for handling isSubarrayAvailable attribute events.

subscribe_events(*dev_info*) → None
Method for subscribing events

unscubscribe_dish_health(*dish_proxy*, *evt_id*)
Method for unsubscribing dish health

unscubscribe_dish_leaf_health(*dish_leaf_proxy*, *evt_id*)
Method for unsubscribing dish leaf health

unscubscribe_dish_leaf_state(*dish_leaf_proxy*, *evt_id*)
Method for unsubscribing dish leaf health.

unscubscribe_dish_pointing(*dish_proxy*, *evt_id*)
Method for unsubscribing dish pointing

unscubscribe_dish_state(*dish_proxy*, *evt_id*)
Method for unsubscribing dish state

unsubscribe_dish_events() → None
Method for unsubscribing dish events

unsubscribe_dish_leaf_events()
Method for unsubscribing dish leaf events.

ska_tmc_subarraynode.manager.subarray_node_component_manager module

Module contents

ska_tmc_subarraynode.model package

Submodules

ska_tmc_subarraynode.model.component module

This module is used for maintaining and monitoring the components of subarray device.

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

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: List[ska_tmc_common.device_info.DeviceInfo]

Return the monitored devices.

Returns
the monitored devices

Return type
DeviceInfo[]

get_device(*device_name*)

Return the monitored device info by name.

Parameters
device_name – name of the device

Returns
the monitored device info

Return type
DeviceInfo

invoke_device_callback(*dev_info*)

This method invoke device callback

remove_device(*device_name*)

Remove a device from the list

Parameters
device_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*)

This method sets obs callback

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

this method update device callback and subarray health state

property subarray_availability

Return the aggregated status for subarray node availability

Returns
the subarray availability

Return type
bool

property subarray_health_state

Return the aggregated subarray health state

Returns
the subarray health state

Return type
HealthState

property subarray_id

Return the subarray_id

Returns
the subarray_id

Return type
str

to_dict()

Return result HealthState in dictionary

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

Enum Constant Values for Subarray Node

class ska_tmc_subarraynode.model.enum.**PointingState**(*value*)

Bases: IntEnum

An enumeration class representing the different pointing states of a device.

NONE = 0

READY = 1

SCAN = 4

SLEW = 2

TRACK = 3

UNKNOWN = 5

ska_tmc_subarraynode.model.input module

This module provide Provides input parameters to both the telescopes

class ska_tmc_subarraynode.model.input.**InputParameter**(*changed_callback*)

Bases: object

Method to Check CSP Subarray Device

property csp_subarray_dev_name: str

Return the CSP Subarray device name

Returns
the CSP Subarray device name

Return type
str

property sdp_subarray_dev_name: None

Returns the SDP Subarray device name

Returns
the SDP Subarray device name

Return type
str

property tmc_csp_sln_device_name: str

Return the CSP Subarray Leaf Node device names

Returns
the CSP Subarray Leaf Node device names

Return type
str

property tmc_sdp_sln_device_name: str

Return the SDP Subarray Leaf Node device names

Returns
the SDP Subarray Leaf Node device names

Return type
str

update(*component_manager*) → List[str]

update the devices in liveliness probe

class ska_tmc_subarraynode.model.input.**InputParameterLow**(*changed_callback*)

Bases: *InputParameter*

Method to Check CSP subarray Low Device

class ska_tmc_subarraynode.model.input.**InputParameterMid**(*changed_callback*)

Bases: *InputParameter*

Method to Check CSP Subarray Mid Device

property dish_dev_names: List[str]

Return the dish device names

Returns
the TM dish device names

Return type
list

property tmc_dish_ln_device_names: List[str]

Return the TM dish device names

Returns
the TM dish device names

Return type
list

update(*component_manager*) → None

Update input parameter for Input Parameter

Module contents

3.1.2 Submodules

3.1.3 ska_tmc_subarraynode._subarraynode_node module

3.1.4 ska_tmc_subarraynode._subarraynode_node_low module

3.1.5 ska_tmc_subarraynode._subarraynode_node_mid module

3.1.6 ska_tmc_subarraynode.exceptions module

This module has custom exception for repository ska_tmc_subarraynode

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

This module is for identifing and changing the transaction ids

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

This method decorator that identifies a transaction with a unique ID and adds it to the wrapped function's object.

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

This method injecting a transaction id

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

For this method A decorator that injects an ID field into a dictionary-like argument of a function.

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

Updates the given dictionary-like *obj* with an ID field and the values in *parameters*.

3.1.9 Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

E

G

H

I

M

N

O

P

R

S

T

U

V