TMC CSP Leaf Nodes Documentation

Release 1.0

NCRA India

Feb 03, 2023

GETTING STARTED

1 Background 3

2 Set up your development environment 5

3 TMC CSP Leaf Nodes code quality guidelines 7

4 ska_tmc_cspmasterleafnode package 9

5 ska_tmc_cspsubarrayleafnode package 15

6 Indices and tables 19

Python Module Index 21

Index 23

This project is developing the TMC CSP Leaf Nodes component of the Telescope Monitoring and Control (TMC) prototype, for the Square Kilometre Array.

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

CHAPTER

ONE

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.

CHAPTER

TWO

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).

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.

2.2 How to Use

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

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

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

THREE

TMC CSP LEAF NODES CODE QUALITY GUIDELINES

3.1 Code formatting / style

3.1.1 Black

TMC CSP Leaf Nodes 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.

3.1.2 Linting

TMC CSP Leaf Nodes 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.

3.2 Test coverage

TMC CSP Leaf Nodes uses pytest to test its code, with the pytest-cov plugin for measuring coverage.

CHAPTER

FOUR

SKA_TMC_CSPMASTERLEAFNODE PACKAGE

4.1 Subpackages

4.1.1 ska_tmc_cspmasterleafnode.commands package

Submodules

ska_tmc_cspmasterleafnode.commands.abstract_command module

Abstract command class for Csp Master Leaf Node

class ska_tmc_cspmasterleafnode.commands.abstract_command.**CspMLNCommand**(**args: Any*, ***kwargs: Any*)

Bases: TmcLeafNodeCommand

Abstract command class for all CspMasterLeafNode

check_unresponsive() → None

Checks if the device is unresponsive

Returns
None

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

Creates adapter for underlying csp master device

ska_tmc_cspmasterleafnode.commands.on_command module

On command class for CSPMasterLeafNode.

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

Bases: *CspMLNCommand*

A class for CspMasterLeafNode's On() command.

On command on CspmasterLeafNode enables the telescope to perform further operations and observations. It Invokes On command on Csp Master device.

do(*argin=None*)

Method to invoke On command on Csp Master.

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

A method to invoke the On command. It sets the task_callback status according to command progress.

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

ska_tmc_cspmasterleafnode.commands.standby_command module

Standby command class for Csp Master Leaf Node

class ska_tmc_cspmasterleafnode.commands.standby_command.**Standby**(**args: Any*, ***kwargs: Any*)

Bases: *CspMLNCommand*

A class for CspMasterLeafNode's Standby() command.

Standby command on CspMasterLeafNode invokes Standby command on Csp Master device.

do(*argin=None*)

Method to invoke Standby command on Csp Master.

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

A method to invoke the Standby command. It sets the task_callback status according to command progress.

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

Module contents

Init file for Csp Master Leaf Node Commands

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

Bases: *CspMLNCommand*

A class for CspMasterLeafNode's Off() command.

Off command on CspMasterLeafNode invokes Off command on Csp Master device.

do(*argin=None*)

Method to invoke Off command on Csp Master.

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

A method to invoke the Off command. It sets the task_callback status according to command progress.

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

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

Bases: *CspMLNCommand*

A class for CspMasterLeafNode's On() command.

On command on CspmasterLeafNode enables the telescope to perform further operations and observations. It Invokes On command on Csp Master device.

do(*argin=None*)

Method to invoke On command on Csp Master.

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

A method to invoke the On command. It sets the task_callback status according to command progress.

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

class ska_tmc_cspmasterleafnode.commands.**Standby**(**args: Any*, ***kwargs: Any*)

Bases: *CspMLNCommand*

A class for CspMasterLeafNode's Standby() command.

Standby command on CspMasterLeafNode invokes Standby command on Csp Master device.

do(*argin=None*)

Method to invoke Standby command on Csp Master.

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

A method to invoke the Standby command. It sets the task_callback status according to command progress.

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

4.1.2 ska_tmc_cspmasterleafnode.manager package

Submodules

ska_tmc_cspmasterleafnode.manager.component_manager module

This module implements ComponentManager class for the Csp Master Leaf Node.

class ska_tmc_cspmasterleafnode.manager.component_manager.**CspMLNComponentManager**(**args: Any*, ***kwargs: Any*)

Bases: TmcLeafNodeComponentManager

A component manager for The CSP Master Leaf Node component.

It supports in controlling the behaviour of CSP Master.

property csp_master_device_name: str

Returns device name for the Csp Master Device.

is_command_allowed(*command_name: str*) → bool

Checks whether this command is allowed. It checks that the device is in the right state to execute this command and that all the components needed for the operation are not unresponsive.

Returns
True if this command is allowed

Return type
boolean

off_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the Off command for execution.

Return type
tuple

on_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the On command for execution.

Return type
tuple

standby_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the Standby command for execution.

Return type
tuple

Module contents

Init file for Csp Master Leaf Node Manager

class ska_tmc_cspmasterleafnode.manager.**CspMLNComponentManager**(**args: Any*, ***kwargs: Any*)

Bases: TmcLeafNodeComponentManager

A component manager for The CSP Master Leaf Node component.

It supports in controlling the behaviour of CSP Master.

property csp_master_device_name: str

Returns device name for the Csp Master Device.

is_command_allowed(*command_name: str*) → bool

Checks whether this command is allowed. It checks that the device is in the right state to execute this command and that all the components needed for the operation are not unresponsive.

Returns
True if this command is allowed

Return type
boolean

off_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the Off command for execution.

Return type
tuple

on_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the On command for execution.

Return type
tuple

standby_command(*task_callback=None*) → Tuple[ska_tango_base.executor.TaskStatus, str]

Submits the Standby command for execution.

Return type
tuple

4.2 Submodules

4.3 ska_tmc_cspmasterleafnode._csp_master_leaf_node module

4.4 Module contents

CspMasterLeafNode

CHAPTER

FIVE

SKA_TMC_CSPSUBARRAYLEAFNODE PACKAGE

5.1 Subpackages

5.1.1 ska_tmc_cspsubarrayleafnode.commands package

Submodules

ska_tmc_cspsubarrayleafnode.commands.abstract_command module

Abstract Command Class for Csp Subarray Leaf Node

class ska_tmc_cspsubarrayleafnode.commands.abstract_command.**CspSLNCommand**(**args: Any*, ***kwargs: Any*)

Bases: TmcLeafNodeCommand

Abstract command class for all CspSubarrayLeafNode

do(*argin=None*)

This method will determine if telescope device types are for low or mid and accordingly invoke required behaviour

init_adapter() → tuple

ska_tmc_cspsubarrayleafnode.commands.on_command module

ska_tmc_cspsubarrayleafnode.commands.off_command module

ska_tmc_cspsubarrayleafnode.commands.assign_resources_command module

AssignResources command class for CspSubarrayLeafNode.

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

Bases: *CspSLNCommand*

A class for CspSubarayLeafNode's AssignResources() command.

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

assign_resources(*argin: str*, *logger=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 CspSubarray.

Parameters
argin – Input JSON string

:type argin : str :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

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

Method to invoke AssignResources command on CSP low Subarray. :param argin: DevString. The string in JSON format. The JSON contains following values: subarray_id: integer Example: {"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"}]}}

Note: Enter the json string without spaces as an input. :return: (ResultCode, message) :raises Exception: if the command execution is not successful

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

Method to invoke AssignResources command on CSP Subarray. :param argin: DevString. The string in JSON format. The JSON contains following values: subarray_id: integer dish: Mandatory JSON object consisting of receptor_ids: DevVarString The individual string should contain dish numbers in string format with preceding zeroes upto 3 digits. E.g. 0001, 0002. Example: { "interface": "https://schema.skao.int/ska-mid-csp-assignresources/2.0", "subarray_id": 1, "dish": { "receptor_ids": ["0001", "0002"] } } Note: Enter the json string without spaces as an input. :return: (ResultCode, message)

Raises
Exception – if the command execution is not successful

validate_json_argument(*input_argin: str*) → tuple

Validates the json argument

validate_json_argument_low(*input_argin: str*) → tuple

Validates the json argument

ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command module

ReleaseAllResources command class for CSPSubarrayLeafNode.

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

Bases: *CspSLNCommand*

A class for CspSubarayLeafNode's ReleaseAllResources() command.

Releases all the resources of given CSP Subarray Leaf Node.

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

Method to invoke ReleaseAllResources command on CSP Subarray.

Parameters
argin – DevString. The string in JSON format. Defaults to None.

Returns
(ResultCode, message)

Raises
Exception – if the command execution is not successful

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

Method to invoke ReleaseAllResources command on CSP Subarray.

Parameters
argin – DevString. The string in JSON format. Defaults to None.

Returns
(ResultCode, message)

Raises
Exception – if the command execution is not successful

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

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

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

Module contents

5.1.2 ska_tmc_cspsubarrayleafnode.manager package

Submodules

ska_tmc_cspsubarrayleafnode.manager.component_manager module

ska_tmc_cspsubarrayleafnode.manager.event_receiver module

Module contents

5.2 Submodules

5.3 ska_tmc_cspsubarrayleafnode._csp_subarray_leaf_node module

5.4 Module contents

CspSubarrayLeafNode

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search
- search

PYTHON MODULE INDEX

S

ska_tmc_cspmasterleafnode, 13
ska_tmc_cspmasterleafnode.commands, 10
ska_tmc_cspmasterleafnode.commands.abstract_command, 9
ska_tmc_cspmasterleafnode.commands.on_command, 9
ska_tmc_cspmasterleafnode.commands.standby_command, 10
ska_tmc_cspmasterleafnode.manager, 12
ska_tmc_cspmasterleafnode.manager.component_manager, 11
ska_tmc_cspsubarrayleafnode, 17
ska_tmc_cspsubarrayleafnode.commands, 17
ska_tmc_cspsubarrayleafnode.commands.abstract_command, 15
ska_tmc_cspsubarrayleafnode.commands.assign_resources_command, 15
ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command, 16

INDEX

A

assign_resources() (*ska_tmc_cspsubarrayleafnode.commands.assign_resources_command.AssignResources method*), 15
AssignResources (*class in ska_tmc_cspsubarrayleafnode.commands.assign_resources_command*), 15

C

check_unresponsive() (*ska_tmc_cspmasterleafnode.commands.abstract_command.CspMLNCommand method*), 9
csp_master_device_name (*ska_tmc_cspmasterleafnode.manager.component_manager.CspMLNComponentManager property*), 11
csp_master_device_name (*ska_tmc_cspmasterleafnode.manager.CspMLNComponentManager property*), 12
CspMLNCommand (*class in ska_tmc_cspmasterleafnode.commands.abstract_command*), 9
CspMLNComponentManager (*class in ska_tmc_cspmasterleafnode.manager*), 12
CspMLNComponentManager (*class in ska_tmc_cspmasterleafnode.manager.component_manager*), 11
CspSLNCommand (*class in ska_tmc_cspsubarrayleafnode.commands.abstract_command*), 15

D

do() (*ska_tmc_cspmasterleafnode.commands.Off method*), 10
do() (*ska_tmc_cspmasterleafnode.commands.On method*), 11
do() (*ska_tmc_cspmasterleafnode.commands.on_command.On method*), 9
do() (*ska_tmc_cspmasterleafnode.commands.Standby method*), 11
do() (*ska_tmc_cspmasterleafnode.commands.standby_command.Standby method*), 10
do() (*ska_tmc_cspsubarrayleafnode.commands.abstract_command.CspSLNCommand method*), 15
do_low() (*ska_tmc_cspsubarrayleafnode.commands.assign_resources_command.AssignResources method*), 16
do_low() (*ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command.ReleaseAllResources method*), 16
do_mid() (*ska_tmc_cspsubarrayleafnode.commands.assign_resources_command.AssignResources method*), 16
do_mid() (*ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command.ReleaseAllResources method*), 17

I

init_adapter() (*ska_tmc_cspmasterleafnode.commands.abstract_command.CspMLNCommand method*), 9
init_adapter() (*ska_tmc_cspsubarrayleafnode.commands.abstract_command.CspSLNCommand method*), 15
is_command_allowed() (*ska_tmc_cspmasterleafnode.manager.component_manager.CspMLNComponentManager method*), 12
is_command_allowed() (*ska_tmc_cspmasterleafnode.manager.CspMLNComponentManager method*), 12

M

module
ska_tmc_cspmasterleafnode, 13
ska_tmc_cspmasterleafnode.commands, 10
ska_tmc_cspmasterleafnode.commands.abstract_command, 9
ska_tmc_cspmasterleafnode.commands.on_command, 9
ska_tmc_cspmasterleafnode.commands.standby_command, 10
ska_tmc_cspmasterleafnode.manager, 12
ska_tmc_cspmasterleafnode.manager.component_manager, 11
ska_tmc_cspsubarrayleafnode, 17
ska_tmc_cspsubarrayleafnode.commands, 17
ska_tmc_cspsubarrayleafnode.commands.abstract_command, 15
ska_tmc_cspsubarrayleafnode.commands.assign_resources_command, 15
ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command, 16

O

Off (*class in ska_tmc_cspmasterleafnode.commands*), 10
off() (*ska_tmc_cspmasterleafnode.commands.Off method*), 10
off_command() (*ska_tmc_cspmasterleafnode.manager.component_manager.CspMLNComponentManager method*), 12
off_command() (*ska_tmc_cspmasterleafnode.manager.CspMLNComponentManager method*), 12
On (*class in ska_tmc_cspmasterleafnode.commands*), 10
On (*class in ska_tmc_cspmasterleafnode.commands.on_command*), 9
on() (*ska_tmc_cspmasterleafnode.commands.On method*), 11
on() (*ska_tmc_cspmasterleafnode.commands.on_command.On method*), 9
on_command() (*ska_tmc_cspmasterleafnode.manager.component_manager.CspMLNComponentManager method*), 12
on_command() (*ska_tmc_cspmasterleafnode.manager.CspMLNComponentManager method*), 12

R

release_all_resources() (*ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command.ReleaseAllResources method*), 17
ReleaseAllResources (*class in ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command*), 16

S

ska_tmc_cspmasterleafnode
    module, 13
ska_tmc_cspmasterleafnode.commands
    module, 10
ska_tmc_cspmasterleafnode.commands.abstract_command
    module, 9
ska_tmc_cspmasterleafnode.commands.on_command
    module, 9
ska_tmc_cspmasterleafnode.commands.standby_command
    module, 10
ska_tmc_cspmasterleafnode.manager
    module, 12
ska_tmc_cspmasterleafnode.manager.component_manager
    module, 11
ska_tmc_cspsubarrayleafnode
    module, 17
ska_tmc_cspsubarrayleafnode.commands
    module, 17
ska_tmc_cspsubarrayleafnode.commands.abstract_command
    module, 15
ska_tmc_cspsubarrayleafnode.commands.assign_resources_command
    module, 15
ska_tmc_cspsubarrayleafnode.commands.release_all_resources_command
    module, 16
Standby (*class in ska_tmc_cspmasterleafnode.commands*), 11
Standby (*class in ska_tmc_cspmasterleafnode.commands.standby_comman*), 10
standby() (*ska_tmc_cspmasterleafnode.commands.Standby method*), 11
standby() (*ska_tmc_cspmasterleafnode.commands.standby_command.Sta*), 10
standby_command() (*ska_tmc_cspmasterleafnode.manager.component_m*), 12
standby_command() (*ska_tmc_cspmasterleafnode.manager.CspMLNComp*), 13

V

validate_json_argument() (*ska_tmc_cspsubarrayleafnode.commands.assign_resources_com*), 16
validate_json_argument_low() (*ska_tmc_cspsubarrayleafnode.commands.assign_resources_com*), 16