TMC CSP Leaf Nodes Documentation

Release 1.0

NCRA India

Apr 01, 2022

GETTING STARTED

This project is developing the TMC 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

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

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

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

Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TmcLeafNodeCommand._name

Abstract command class for all CspMasterLeafNode

check_allowed()

Checks whether this command is allowed It checks that the device is in the right state to execute 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_unresponsive()

init_adapter()

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: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TmcLeafNodeCommand._name

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.

ska_tmc_cspmasterleafnode.commands.standby_command module

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

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.

Module contents

class ska_tmc_cspmasterleafnode.commands.**On**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TmcLeafNodeCommand._name

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.

class ska_tmc_cspmasterleafnode.commands.**Standby**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_command.ska_tmc_common.tmc_command.TmcLeafNodeCommand._name

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.

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: ska_tmc_common.tmc_component_manager.ska_tmc_common.tmc_component_manager.TmcLeafNodeComponentManager._name

A component manager for The CSP Master Leaf Node component.

It supports in controlling the behaviour of CSP Master.

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

Parameters **exception** – an exception

Type Exception

update_device_info(*csp_master_dev_name*)

Module contents

class ska_tmc_cspmasterleafnode.manager.**CspMLNComponentManager**(**args: Any*, ***kwargs: Any*)
Bases: ska_tmc_common.tmc_component_manager.ska_tmc_common.tmc_component_manager.TmcLeafNodeComponentManager._name

A component manager for The CSP Master Leaf Node component.

It supports in controlling the behaviour of CSP Master.

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

Parameters **exception** – an exception

Type Exception

update_device_info(*csp_master_dev_name*)

4.2 Submodules

4.3 ska_tmc_cspmasterleafnode._csp_master_leaf_node module

CSP Master Leaf node acts as a CSP contact point for Master Node and also to monitor and issue commands to the CSP Master.

class ska_tmc_cspmasterleafnode.csp_master_leaf_node.**CspMasterLeafNode**(**args: Any*, ***kwargs: Any*)
Bases: ska_tango_base.ska_tango_base.SKABaseDevice._name

CSP Master Leaf node acts as a CSP contact point for Master Node and also to monitor and issue commands to the CSP Master.

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

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

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

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

rtype: (ResultCode, str)

Off()
This command invokes Off() command on Csp Master.

always_executed_hook()

create_component_manager()

delete_device()

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

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

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

Return type boolean

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

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

Return type boolean

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

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

Return type boolean

read_commandExecuted()
Return the commandExecuted attribute.

read_cspMasterDevName()
Return the cspmasterdevname attribute.

write_cspMasterDevName(*value*)
Set the cspmasterdevname attribute.

ska_tmc_cspmasterleafnode.csp_master_leaf_node.**main**(*args=None*, ***kwargs*)
Runs the CspMasterLeafNodeMid. :param args: Arguments internal to TANGO

Parameters **kwargs** – Arguments internal to TANGO

Returns CspMasterLeafNodeMid TANGO object.

4.4 Module contents

CspSubarrayLeafNode

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search
- search

PYTHON MODULE INDEX

S

INDEX