TMC integration Documentation

Release 1.0

NCRA India

Aug 25, 2023

GETTING STARTED

This project is integration of the TMC components for the Square Kilometre Array.

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

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-integration.git cd ska-tmc-integration

To deploy the pods: :: make k8s-install-chart

To test the integration test cases: :: make k8s-test

To uninstall the pods: :: make k8s-uninstall-chart

To watch the pods, services status: :: make k8s-watch

CHAPTER

THREE

TMC INTEGRATION CODE QUALITY GUIDELINES

3.1 Code formatting / style

3.1.1 Black

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

CHAPTER

FOUR

TMC (TELESCOPE MONITORING AND CONTROL)

The Telescope Monitor and Control (TMC) is the software module identified to perform the telescope management, and data management functions of the Telescope Manager. Main responsibilities identified for TMC are:

- Operational monitoring and control of the telescope
- Support execution of astronomical observations
- Manage telescope hardware and software subsystems in order to perform astronomical observations
- Manage the data to support operators, maintainers, engineers and science users to achieve their goals
- Determine telescope state.

To support these responsibilities, the TMC performs high-level functions such as Observation Execution, Monitoring and Control of Telescope, Resource Management, Configuration Management, Alarm and Fault Management, and Telescope Data Management (Historical data and Real time data). These high level functions are again divided into lower level functions to perform the specific functionalities.

The TMC has a hierarchy of control nodes for Mid and Low- Central Node, Subarray Node, SDP Leaf Nodes, CSP Leaf Nodes, MCCS Leaf Nodes, Dish Leaf Nodes.

The components(CentralNode, SubarrayNode, Leaf Nodes) of the TMC system are integrated in the TMC integration repository, which contains the Helm chart to deploy the TMC. More details on the design of the TMC and how to run it locally or in the integration environment can be found in the Documentation

4.1 Observation Execution APIs

The observation execution can be done by following a sequence of APIs as follows:

- Resource allocation
- Configure a scan
- Perform scan
- End a scan
- Resource deallocation

Before performing any observation related operation it is necessary that the telescope is in ON state.

4.2 Operational Monitoring and Control APIs

TMC provides APIs for controlling the telescope as follows:

- TelescopeOn
- TelescopeOff
- Standby

To monitor the telescope, the Central Node exposes following attributes:

- `telescopeState`
- `healthState`

4.3 Indices and tables

- genindex
- modindex
- search
- search