CSP.LMC Common Software Documentation

Release 0.17.2

SKA Organization

Jul 06, 2023

CSP.LMC COMMON PACKAGE:

CHAPTER

ONE

CSP.LMC COMMON PACKAGE DESCRIPTION

General requirements for the monitor and control functionality are the same in both telescopes. In addition two of three other CSP Sub-elements, namely PSS and PST, have the same functionality and use the same design for both the telescopes.

Functionality common to Low and Mid CSP.LMC includes: communication framework, logging, archiving, alarm generation, subarraying, some of the functionality realated to handling observing mode changes, Pulsar Search and Pulsar Timing, and to some extent Very Long Baseline Interferometry (VLBI).

The difference between LOW and MID CSP.LMC is mostly due to the different receivers (dishes vs stations) and different CBF functionality and design. More than the 50% of the CSP.LMC functionality is common for both telescopes.

The CSP.LMC Common Package comprises all the software components and functionality common to LOW and MID CSP.LMC and is used as a base for development of the Low CSP.LMC and Mid CSP.LMC software.

The *CSP.LMC Common Package* is delivered as a part of each CSP.LMC release, via a Python package that can be used as required for maintenance and upgrade.

CSP.LMC implements a high level interface (API) that Telescope Manager (TM), or other authorized client, can use to monitor and control CSP as a single instrument.

At the same time, CSP.LMC provides high level commands that the TM can use to sub-divide the array into up to 16 sub-arrays, i.e. to assign station/receptors to sub-arrays, and to operate each sub-array independently and concurrently with all other sub-arrays.

The top-level software components provided by CSP.LMC API are:

- *Csp Controller*
- *Csp Subarray*
- CSP Alarm Handler (TBD)
- CSP Logger (TBD)
- CSP TANGO Facility Database (TBD)
- Input processor Capability (receptors/stations) (TBD)
- *Search Beam Capability* (TBD)
- *Timing Beam Capability* (TBD)
- *VLBI Beam Capability* (TBD)

Components listed above are implemented as TANGO devices, i.e. classes that implement standard TANGO API. The CSP.LMC TANGO devices are based on the standard SKA1 TANGO Element Devices provided via the *SKA Base Classes package*.

1.1 CSP.LMC Controller

The CSP controller provides API for monitor and control the CSP sub-system. CSP Controller is the primary point of access for CSP Monitor and Control.

CSP Controller maintains the pool of schedulable resources, and it can relies on the CSP CapabilityMonitor devices, as needed. The CSP Controller implements CSP sub-system-level status indicators, configuration parameters, housekeeping commands.

1.2 CSP.LMC Subarray

The core CSP functionality, configuration and execution of signal processing, is configured, controlled and monitored via subarrays.

CSP Subarray makes provision to TM to configure a subarray, select Processing Mode and related parameters, specify when to start/stop signal processing and/or generation of output products. TM accesses directly a CSP Subarray to:

- Assign resources
- Configure a scan
- Control and monitor states/operations

1.2.1 Resources assignment

The assignment of Capabilities to a subarray (*subarray composition*) is performed in advance of a scan configuration. Assignable Capabilities for CSP Subarrays are:

- receptors (MID) or stations (LOW)
- tied-array beams: Search Beams, Timing Beams and Vlbi Beams.

In general resource assignment to a subarray is exclusive, but in some cases the same Capability instance may be used in shared manner by more then one subarray.

1.2.2 Inherent Capabilities

Each CSP subarray has also a set of permanently assigned *inherent Capabilities*: the number and type is different for LOW and MID instance.

Only the Inherent Capabilities related to the Processing Mode are common to both instances.

These are:

- Correlation
- PSS
- PST
- VLBI

An inherent Capability can be enabled or disabled, but cannot assigned or removed to/from a subarray.

1.2.3 Scan configuration

TM provides a complete scan configuration to a subarray via an ASCII JSON encoded string. Parameters specified via a JSON string are implemented as TANGO Device attributes and can be accessed and modified directly using the buil-in TANGO method *write_attribute*. When a complete and coherent scan configuration is received and the subarray configuration (or re-configuration) completed, the subarray it's ready to observe.

1.2.4 Control and Monitoring

Each CSP Subarray maintains and report the status and state transitions for the CSP subarray as a whole and for individual assigned resources.

In addition to pre-configured status reporting, a CSP subarray makes provision for the TM and any authorized client, to obtain the value of any subarray attribute.

1.3 CSP.LMC Capabilities

Capabilities represent the CSP schedulable resources and provide API that can be used to configure, monitor and control resources that implement signal processing functionality. During normal operations, TM uses the sub-array API to assign capabilities to the sub-array, configure sub-array Processing Mode, start and stop scan.

The CSP.LMC Common Package implements the capabilities that are shared between LOW and MID instances.

These are:

- *CSP Search Beam Capability*
- *CSP Timing Beam Capability*
- *CSP VLBI Beam Capability*

1.3.1 CSP.LMC Search Beam Capability

(To be implemented)

The Search Beam Capability exposes the attributes and commands to monitor and control beam-forming and PSS processing in a single beam.

The mapping between an instance of the CSP Search Beam and the internal CSP Sub-element components performing beam-forming and search is established at initialization and is permanent.

CSP.LMC SearchBeamCapability API Documentation

(To be implemented)

1.3.2 CSP.LMC Timing Beam Capability

(To be implemented)

The Timing Beam Capability exposes the attributes and commands to monitor and control beam-forming and PST processing in a single beam.

The mapping between an instance of the CSP Search Beam and the internal CSP Sub-element components performing beam-forming and search is established at initialization and is permanent.

CSP.LMC TimingBeamCapability API Documentation

(To be implemented)

1.3.3 CSP.LMC VLBI Beam Capability

(To be implemented)

The VLBI Beam Capability exposes the attributes and commands to monitor and control beamforming and VLBI processing in a single beam.

CSP.LMC VlbiBeamCapability API Documentation

1.3.4 CSP.LMC CapabilityMonitor

(To be implemented)

CSP.LMC CapabilityMonitor API Documentation

(To be implemented)

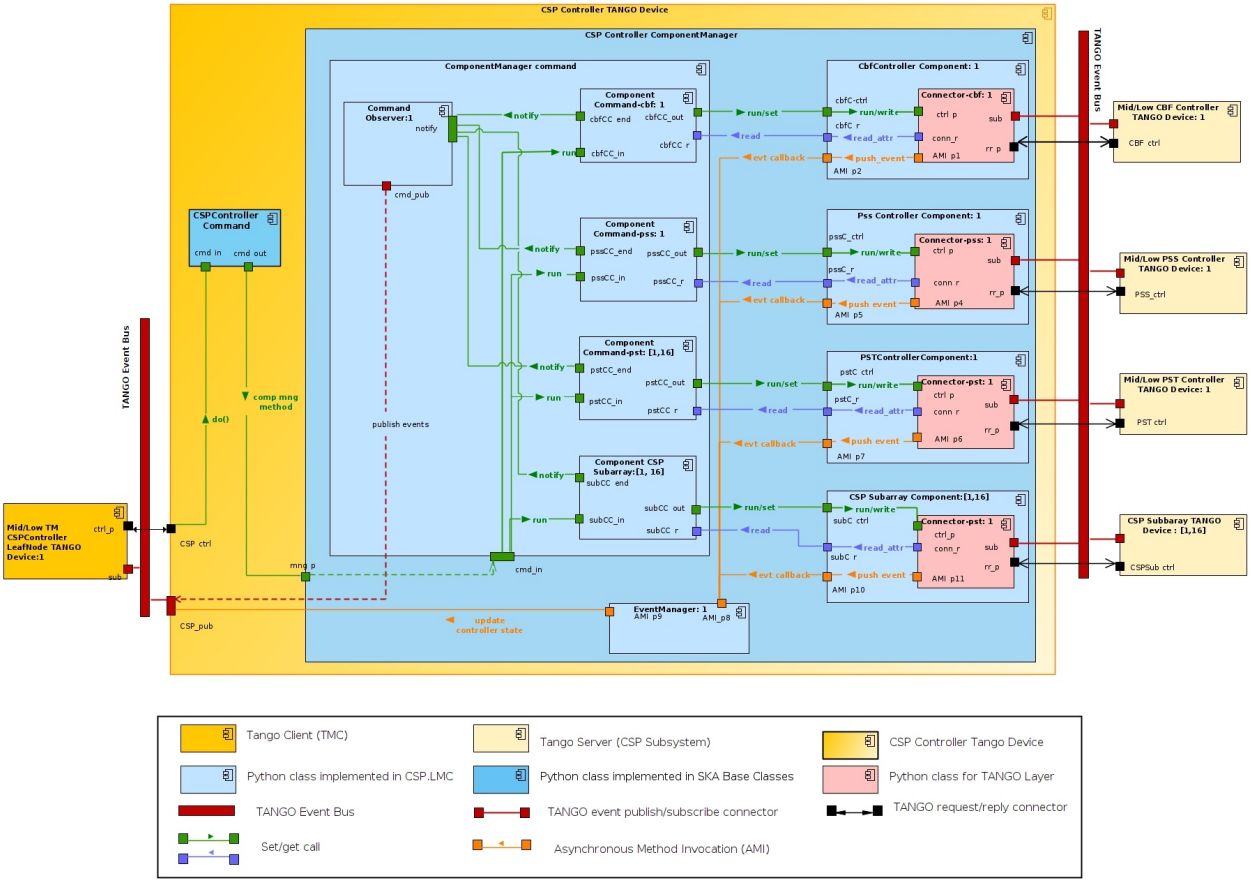
CHAPTER

TWO

ARCHITECTURE DESCRIPTION

The architecture of CSP.LMC is shared between the Controller and the Subarray. Both of them communicate with three sub-systems: CBF, PSS and PST. The Controller must also access the CSP.LMC Subarrays and the Capabilities device manager to report the information on the resources.

In the figure below, the C&C view of CSP.LMC controller is provided. The case of CSP.LMC subarray is identical, where Subsystem's Controller are substituted with correspondant Subsystem's subarrays and no other CSP.LMC subarrays are controlled. Further diagrams and a more comprehensive description of its component can be found at this page.



The main operations of CSP are carried out into the three sub-elements. The interaction between the CSP Controller and the subordinate sub-systems devices are mediated through a Python class that works as a proxy (Component Class). This approach has the advantage of abstraction.

Since version 0.11.0 the state machine of ska-tango-base is no longer used. The motivation of this choice is described

here. For this reason, custom state models are implemented for the Operational, Observing and Health State (*CSP State Models*).

2.1 Interface to subsystem TANGO devices

Specific operations on a sub-element can be done by specializing the proxy class for each sub-system and the corresponding functions are maintained in a specific part of the code.

2.1.1 Csp sub-system Component

The component class is a mediator between CSP.LMC and a Subsystem Device. It acts as an adapter and allows, when needed, to execute specific instructions on a subsystem before invoking the required command. In other words, its functionalities are:

- read and write of associated device's attributes;
- command execution;
- subscription of attributes on the corresponding Tango Device.

2.1.2 Connector

Connector Class is class working as interface to the TANGO system. It relies on TangoClient class of ska-tmc-common package developed by NCRA team, and it has the purpose to communicate with the device proxy of Sub System TANGO device for all the functionalities used by the Component classe.

One of the main advantage to have this class, it the possibility to be easily mocked during the tests.

2.2 Commands execution

A command issued on the CSP Controller or CSP Subarray (controller command) by a TANGO client or the TM, breaks up, nearly always, into several commands (>=3), one for each CSP sub-system. These commands (sub-commands or component commands) are forwarded to the connected sub-sub-system.

The CSP Controller or Subarray TANGO device has to be able to invoke the command on a sub-element and monitors its execution, detecting its progress and its final status (success/failure).

The sequence of operation to be performed are the following:

- check the initial device state to determine if the command is allowed;
- wait for the final status (the one expected after the end of successful execution) and detect possible conditions of failures;
- implement support for timeout;
- report the end of the command.

The execution of a command is reported by the attribute CommandResult, which is a Tuple with the name of the latest command invoked and a the resultCode ENUM (from ska-tango-base) that report the state of the command (SUCCEEDED:0, STARTED:1, FAILED:3)

2.2.1 Command Observer

A specific Python class (CommandObserver Class), using the Observer Pattern Design, is used to detect the controller command completion. Each component command is registered within the observer and notifies it when it has completed

A CSP Subarray command is considered completed when all the forwarded commands have ended. This component monitors the execution of a CSP Subarray command, keeping track of the commands running on the CSP sub-systems.

When the execution of a command ends on a sub-system, the Component sub-system notifies this condition to the CommandObserver invoking the notify method provided by this component.

This component implements also a set of attributes to report information about the status of each monitored sub-system command, as for example the running and progress status.

At the end of the command, the Command Observer report the status of the command to the commandResult attribute.

2.2.2 Sub-system Command (Component Command)

The Component Command models a command acting on a sub-system Component instance. It implements the logic to manage and control the command issued on a single component. The ComponentCommand class, when instantiated for a specific command (On, Off , etc) contains all the information about the request such as:

- the input parameters (if any)
- the Component to act on
- success, failure and timeout conditions

When the CSP Controller invokes the run method, each Component Command will run one (or more actions) on the associated Component object. When the Command ends, it reports to the Command Observer the success or the failure.

2.3 Event Manager

Management of the events is delegated to a specific class (Event Manager Class). On initialization completion (when the connection with the sub-system devices has been established) CSP.LMC devices (Controller and Subarray) select which events are to be monitored on the sub-systems and delegate the subscription to the EventManager. The aim of this class is to aggregate and report to TM the collective states and modes of the CSP (State, ObsState, HealthState, ecc…).

In other words, Event Manager works on the behalf of the CSP.LMC to:

- subscribes the events for the main state and modes subsystem's attributes (registering callbacks to the Component's classes);
- retrieve the value or errors reported by the callback registered with the events
- carry out particular policies of aggregation on attributes, reducing the load of information traveling to the subarray;

This object does not subscribe directly to a sub-system TANGO devices, but relies on the corresponding Component objects to perform such work. The events received from each sub-system are pushed back to the CSP Subarray via callbacks registered at subscription time.

CHAPTER

THREE

PROJECT'S API

3.1 CSP.LMC Common Devices API

3.1.1 CspController

3.1.2 CspSubarray

3.2 CSP.LMC modules API

3.2.1 Manager subpackage

Controller Component Manager

Subarray Component Manager

Event Manager

Component Manager Configurator

3.2.2 CSP Sub-system Component

Component

Observing Component

CBF Controller Component

PSS Controller Component

PST Controller Component

CBF Subarray Component

PSS Subarray Component

PST Beam Component

3.2.3 Command subpackage

Component Command

Base Component Commands

Observing Component Commands

This module includes the ComponentCommand specialized classes.

AssignResources

ReleaseResources

ReleaseAllResources

Configure

Scan

EndScan

Abort

ObsReset

Restart

Macro Component Commands

3.2.4 CSP State Models

Operational State Model

Health State Model

Observing State Model

Controller Operational State Model

Controller Health State Model

Subarray Operational State Model

Subarray Health State Model

Subarray Observing State Model

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search