MID CSP.LMC Software Documentation

Release 1.0.1

SKA Organization

Jul 11, 2022

PSS.LMC DESCRIPTION:

CHAPTER

ONE

PSS.LMC DESCRIPTION

The LMC schema emerging during the bridging phase will be progressively implemented inside the MVP prototype. It will comprise some emulation of main functions. It is foreseen also a minimal PSS pipeline functionality. At the moment Controller and Subarray devices are not implemented. We aim to collect the basic interaction modes with Cheetah in order to be able to control it by a simple device driver to be implemented in PI8. Basic information on Cheetah control has been reported in the AT4-362 documentation. The Cheetah pipeline is a single executable which contains either the CPU and the CUDA version of PSS and SPS pipeline. It is a CLI program and can be controlled either by a configuration program and by command line switches. The command line switch overrides the configuration file ones. The proposed design interact with Cheetah by means of command line and, in future, by OS level signals. We will implement the basic Cheetah pipeline (To became the CTRL module as defined in PSS.DDD) inheriting from the CspSubElementObsDevice d evice of Base classes.

1.1 PssPipeline device - CTRL

The PssPipeline device (CTRL) inherits from base classes CspSubElementObsDevice device In the following are not reported the properties and attributes inherited.

1.1.1 Properties

Properties are set in the Tango Database.

Attribute Name	Type	Note/Description
NodeIP	*Type*: 'DevString'	The IP address of the PSS node where the cheetah pipeline will be running
PipelineName	*Type*: 'DevString'	The pipeline name
CheetahOutputFile	*Type*: 'DevString'	The filename where cheetah stdout adn stderr will be stored
CheetahConfigFile	*Type*: 'DevString'	The filename where cheetah input configuration data wil be stored
CheetahExecutable	*Type*: 'DevString'	A string containing the command script to execute Cheetah.
CheetahUserPasswd	*Type*: ('DevString',)	A Tuple containing the username and password for the remote connection to Cheetah

1.1.2 Attributes

Attribute Name	Attribute Type	Note/Description
lastScanConfiguration	*Type*: 'DevString' *Access*: READ	The last valid scan configuration.
pipelineProgress	*Type*: 'DevUShort' *Access*: READ	The cheetah pipeline progress percentage
cheetahVersion	*Type*: 'DevString' *Access*: READ	The cheetah pipeline version
cheetahPid	*Type*: 'DevLong' *Access*: RE	The filename where cheetah input configuration data wil be stored
cheetahLogLine	*Type*: 'DevString' *Access*: RE	Cheetah pipeline log line
isCommunicating	*Type*: 'DevBoolean' *Access*: READ	A Tuple containing the username and password for the remote connection to Cheetah
NodeIP	*Type*: 'DevString' *Access*: READ	The PSS node IP address NOTE: it return the corresponding property
PipelineName	*Type*: 'DevString' *Access*: READ	The pipeline name NOTE: it return the corresponding property

1.1.3 Commands

Command Name	Input/output args	Note
On	Input: None *Output*: (resultCode, resultMsg)	PSS pipeline: enable (?)
Off	Input: None *Output*: (resultCode, resultMsg)	PSS pipeline: disable (?)
ConfigureScan	Input: DevString JSON formatted with scan configuration *Output*: (resultCode, resultMsg)	
GoToIdle	Input: None *Output*: (resultCode, resultMsg)	PssPipeline transits to IDLE obsState
Scan	*Input*: scan ID (integer) *Output*: (resultCode, resultMsg)	Start a Scan
EndScan	Input: None *Output*: (resultCode, resultMsg)	End a Scan
Abort	Input: None *Output*: (resultCode, resultMsg)	End a Scan Signal abort (see below)
ObsReset	Input: None *Output*: (resultCode, resultMsg)	Reset Pipeline from FAULT/ABORTED to IDLE obsState.

1.2 Cheetah command line interface

Cheetah program can be started with a command line of the form

./cheetah/pipeline/cheetah_pipeline –config cheetah.xml -p Dedispersion –log-level log

The different pipelines which can be selected are:

- Empty
- Dedispersion
- RfiDetectionPipeline
- SinglePulseHandler

For the input stream (the data source) there are the possibility to create synthetic internally generated data or external sources. We chose to use a standard file generated by sigproc fake (*ska.dat* hardcoded in the standard config file). The output of cheetah is on the standard output and it has a standard syslog format:

- [log][tid=140474636963584][/home/baffa/src/ska/cheetah/cheetah/../cheetah/pipeline/detail/BeamLauncher.cpp:148][160076742
  Beams....
- [warn][tid=140474636963584][/home/baffa/src/ska/cheetah/cheetah/../cheetah/tdas/detail/Tdas.cpp:76][1600767420]No
  Time Domain Accelerated Search algorithm has been specified
- [log][tid=140474636963584][/home/baffa/src/ska/cheetah/cheetah/../cheetah/pipeline/detail/BeamLauncher.cpp:171][160076742
  creating pipelines
- [log][tid=140474636963584][/home/baffa/src/ska/cheetah/cheetah/../cheetah/pipeline/detail/BeamLauncher.cpp:223][160076742
  Beam: identifier to distinguish between other similar type blocks
- [log][tid=140474527926016][/home/baffa/src/ska/cheetah/cheetah/../cheetah/sps/detail/Sps.cpp:110][1600767420]setting
  dedispersion buffer size to 2048 spectra
- [log][tid=140474527926016][/home/baffa/src/ska/cheetah/cheetah/../cheetah/sps/detail/Sps.cpp:113][1600767420]setting
  buffer overlap to 1514 spectra
- [log][tid=140474527926016][/home/baffa/src/ska/cheetah/cheetah/sigproc/src/SigProcFileStream.cpp:334][1600767424]resizing
  to 1024
- [log][tid=140474527926016][/usr/local/include/panda/detail/Pipeline.cpp:63][1600767424]End of stream

Cheetah pipeline is a stand alone CLI program with its main output on stdout. It runs on an different server of the Tango Device Server. We have implemented a class, called PipelineCommunicationManager, that have the skeleton of connection. It has to be specialized with the protocol to use with connection. At the present the connection is made via SSH.

This command line approach makes Cheetah program completely deaf while running: it is not foreseen a communication channel from CONTROL to the pipeline. The only possible channel is the use of Posix signals. We list here few useful ones with default behaviour:

1. *SIGKILL 9 Kill signal → Terminate*
2. SIGTERM 15 Termination signal → Terminate
3. SIGUSR1 30,10,16 User-defined → Terminate
4. SIGSTOP 17,19,23 Stop process → Terminate

The use of *SIGKILL*, *SIGTERM* and *SIGSTOP* are clear and related to Abort (first and second) and to Stop (last). We have the opportunity to implement a custom channel by the use of *SIGUSR1*.

NOTE: At the present implementation both Abort and EndScan are sending a Kill Signal

CHAPTER

TWO

PROJECT'S API

2.1 PSS.LMC Ctrl Pipeline API

2.1.1 PSS Ctrl Pipeline Tango Device

class ska_pss_lmc.pipeline.pipeline_ctrl_device.**PipelineCtrlDevice**(**args*, ***kwargs*)

Bases: CspSubElementObsDevice

PSS Pipeline Control Tango device.

Device Properties:

NodeIP

- The IP address of the PSS node where the cheetah pipeline will be running
- Type:'DevString

PipelineName

- The pipeline name
- Type:'DevString'

CheetahOutputFile

- The filename where cheetah stdout adn stderr will be stored
- Type:'DevString'

CheetahConfigFile

- The filename where cheetah input configuration data will be stored
- Type:'DevString'

CheetahExecutable

- A string containing the command script to execute Cheetah
- Type:'DevString'

CheetahUserPasswd

- A Tuple containing the username and password for the remote connection to Cheetah
- Type: ('DevString',)

_init_state_model()

Override base method.

Configure some device attributes to push events from the code.

Return type
None

create_component_manager()

Create and return the Pss Pipeline Control Component Manager.

Return type
PipelineCtrlComponentManager

_component_state_changed(*fault=False*, *power=None*, *configured=None*, *scanning=None*, *obsfault=None*, ***kwargs*)

Update the state of the controlled component.

The device reports only a sub-set of the possible states of the controlled component (cheetah pipeline). These are:

- DISABLE: the device is not trying to connect to the component (default state at device startup)
- UNKNOWN: the devices trying to connect to the PSS node where the software component will be running but the connection is not established
- ON: the device is connected with the PSS node; the software component might or not be in running.
- FAULT: the controlled component is experiencing a fault condition

Return type
None

_force_transition_to_obs_state(*obs_state*)

Force the observing state machine to a state.

Use the state machine auto transitions to force the transition to a not allowed state.

Parameters
obs_state (ObsState) – the desired observing state

Return type
None

_communication_state_changed(*communication_state*)

Update the PipelineCtrlDevice communication status.

Parameters
communication_state (CommunicationStatus) – the status of communication with the controlled component (cheetah pipeline)

Return type
None

update_attribute(*attribute_name*, *attribute_value*)

General method invoked to push an event on a device attribute.

Parameters

- **attribute_name** (str) – the TANGO attribute name
- **attribute_value** (Any) – the attribute value

Return type
None

class InitCommand(**args*, ***kwargs*)

Bases: InitCommand

Class that implements device initialisation for the device.

do()

Initialise the attributes and properties of the device.

Return type
Tuple[ResultCode, str]

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

class CheckLongRunningCommandStatusCommand(**args: Any*, ***kwargs: Any*)

Bases: FastCommand

The command class for the CheckLongRunningCommandStatus command.

do(*argin*)

Determine the status of the command ID passed in, if any.

- Check *command_result* to see if it's finished.
- Check *command_status* to see if it's in progress
- Check *command_ids_in_queue* to see if it's queued

Note: this method has been overridden to fix an issue in the BC implementation.

Parameters
argin (*str*) – The command ID

Returns
The resultcode for this command and the string of the TaskStatus

Return type
tuple (ResultCode.OK, str)

pipelineProgress()

Return the cheetah pipeline progress.

Return type
int

cheetahVersion()

Return the cheetah version.

Return type
int

cheetahPid()

Return the PID of the cheetah pipeline process.

The attribute is stored into the TANGO DB as memorized attribute" so that on device restart it's possible to recover the connection " to the cheetah pipeline process

Return type
int

cheetahLogLine()

Return the cheetah log, updated line by line

Return type
str

isCommunicating()

Return the device communicating status.

Return a boolean flag indicating whether the TANGO device is communicating with the controlled component.

Return type
bool

nodeIP()

Return the pss node IP.

Return type
str

pipelineName()

Return the cheetah pipeline name.

Return type
str

read_lastScanConfiguration()

Return the last programmed configuration.

2.1.2 PSS Ctrl Pipeline Component Manager

class ska_pss_lmc.pipeline.pipeline_component_manager.**PipelineCtrlComponentManager**(*max_workers*, *properties*, *communication_status_changed_callb* *component_state_changed_callba* *update_device_attribute_cbk* *logger=None*)

Bases: TaskExecutorComponentManager

A base component manager for PSS Pipeline Control Device.

property pipeline_configured: bool

Return a flag that states if the pipeline is already configured. 3

It is used in case of re-configuration by the _configure_scan method to properly trigger the state machine.

Return type
bool

property is_communicating: bool

Return whether communication with the component is established.

Return type
bool

Returns
whether there is currently a connection to the component

property log_line: *str*

Stores the last line of pipeline log

Return type
str

property pipeline_progress: *int*

Stores the last line of percentage progress of the pipeline

Return type
int

property cheetah_version: *str*

Return the version of Cheetah pipeline in use.

NOTE: to be implemented

Return type
str

property cheetah_pid: *int*

Return the pid of the cheetah process

Return type
int

property scan_configuration: *str*

Return the JSON script used to configure the pipeline.

Return type
str

create_communication_manager(*logger*)

Instantiate the communication manager to access Cheetah.

Parameters
logger (Logger) – a logger for this instance to use.

Return type
SshAccess

update_component_state(***kwargs*)

Handle a change in component state.

Override the BaseComponentManager *_update_component_state()* method. It invokes the device _component_state_changed method that performs action on the state machine, as required.

Return type
None

_push_component_state_update(***kwargs*)

start_communicating()

Try to open a connection with the PSS Node.

Open a channel with this instance of the component manager and the software component (cheetah pipeline). Submit the _start_communicating method Initial communication status is DISABLED.

Return type
None

_start_communicating_callback(*status*, *communication_status=None*, *exception=None*, *message=None*)

Task callback passed when submitting the task of _start_communicating.

Param
status: the status of the task

Param
communicationStatus: the status of communication to be reported to the Pipeline device

Param
exception: possible exception raised when submitting the task

Param
message: possible message received when submitting the task

Return type
None

_start_communicating(*task_callback=None*, *task_abort_event=None*)

Task submitted by start_communicating.

On failure the communication status is reported as NOT_ESTABLISHED and the adminMode as ONLINE/MAINTENANCE. The state of the device is set to UNKNOWN. Need to re-try the connection.

Param
task_callback: Task callback passed during the submission of the task

Param
task_abort_event: the abort event (still not implemented)

Return type
None

_monitor_scan()

Method invoked at connection when the pipeline process is running.

stop_communicating()

Close connection with the software component.

Close the ssh channel between this instance of the component manager and the controlled software component (cheetah). The communication state variable is set to DISCONNECTED and the adminMode to OFFLINE.

Return type
None

end_scan(*task_callback=None*)

Shutdown pipeline.

Return type
Tuple[TaskStatus, str]

_endscan(*task_callback=None*, *task_abort_event=None*)

abort(*task_callback=None*)

Terminate pipeline.

If the cheetah pipeline is running, it is stopped. If the command is invoked when no observation is in running, the device transition to ABORTED.

Return type
Tuple[TaskStatus, str]

_abort(*task_callback=None*, *task_abort_event=None*)

Task submitted by the task executor on abort request.

Check if the pipeline is running and in this case, it sets the abort event and invokes the termination of the process.

Return type
None

configure_scan(*json_configuration*, *task_callback=None*)

Store the configuration file on PSS Node.

The received configuration file is translated in XML format and stored on the PSS node to be used at pipeline startup.

The original configuration in JSON format is also stored into an instance property.

Param
json_configuration: a string containing the Json configuration

Param
task_callback: the callback invoked when command ends

Return type
Tuple[TaskStatus, str]

_config_scan(*json_configuration*, *task_callback=None*, *task_abort_event=None*)

scan(*scan_id*, *task_callback=None*)

Start cheetah pipeline on the PSS Node.

Run Cheetah pipeline This method is meant to run in its own worker thread, as it is a long running command.

Param
scan_id: an integer the identify the scan

Param
task_callback: the callback invoked when command ends

Return type
None

_scan(*cmd_string*, *task_callback=None*, *task_abort_event=None*)

Task submitted by the active thread of the pool.

Parameters

- **cmd_string** (str) – the command line to run cheetah
- **task_callback** (Optional[Callable]) – the CommandTracker callback invoked when the command complete
- **task_abort_event** (Optional[Event]) – the abort event

Return type
None

_parse_logs(*task_callback=None*)

Parse and forward logs from the cheetah application.

Parameters
task_callback (Optional[Callable]) – CommandTracker method invoked on command completion.

reset_to_idle(*task_callback=None*)

Submit a task to reset the observing state to IDLE.

This method is invoked both by the ObsReset and GoToIdle commands.

Parameters
task_callback (Optional[Callable]) – CommandTRacker method invoked on command completion.

Return type
Tuple[str, str]

Returns
a tuple with the task status and a message

obsreset(*task_callback=None*)

Reset the device from a FAULT/ABORT condition.

Parameters
task_callback (Optional[Callable]) – CommandTRacker method invoked on command completion.

Return type
Tuple[str, str]

Returns
a tuple with the task status and a message

deconfigure(*task_callback=None*)

Transition the device from READY to IDLE.

Return type
Tuple[str, str]

Returns
a tuple with the task status and a message

_reset_to_idle(*task_callback=None*, *task_abort_event=None*)

Task submitted by the reset_to_idle method.

Parameters

- **task_callback** (Optional[Callable]) – CommandTRacker method invoked on command completion.
- **task_abort_event** (Optional[Event]) – the shared event to signal an abort request.

2.2 PSS.LMC modules API

2.2.1 Manager subpackage

Pipeline Communication Manager

class ska_pss_lmc.manager.communication_manager.**PipelineCommunicationManager**(*logger=None*)

Bases: ABC

Communicate with Cheetah

_pid = None

property get_pid: int

Return process id

Return type
int

abstract connect()

Establish connection

Return type
bool

Returns
successful of connection

abstract disconnect()

Close connection

abstract start(*cmd*)

Start Cheetah pipeline

Parameters
cmd (str) – command to be executed

abstract is_running()

Return if cheetah is currently running

Return type
bool

Returns
if cheetah process is running

abstract kill()

Kill Cheetah pipeline process

shutdown()

Gracefully shutdown cheetah, not yet supported in cheetah

abstract write_config(*config*)

Write cheetah configuration

Parameters
config (*xml.etree.ElementTree*) – cheetah xml config

reload_config()

Reload configuration while cheetah is running. Mentioned by chris as future potential addition to cheetahs signals Unclear if this actually can be mapped to the obs state machine Maybe we would just shut down cheetah, and reload with the new config

delete_config()

Delete the configuration file from the PSS node. To be invoked by ObsReset

abstract get_logs()

Get logs from Cheetah process

Return type
Iterator[str]

Returns
iterator over lines of log entry

`_abc_impl = <_abc_data object>`

class ska_pss_lmc.manager.communication_manager.**SshAccess**(*host*, *user*, *password=None*, *private_key_path=None*, *logger=None*)

Bases: `PipelineCommunicationManager`

Access Cheetah over ssh

NOTE: there are several exceptions that paramiko library might throw when trying to connect to the ssh server, executing commands, etc. They shall be handled at component manager level.

`log_input = None`

`client = None`

`host = None`

`user = None`

`password = None`

`private_key_path = None`

`connect()`
Establish connection

Return type
`bool`

`disconnect()`
Close connection

`start(cmd)`
Start Cheetah pipeline

`fetch_pid()`
Fetch pid stored on cheetah host

`is_running()`
Check if the pipeline is running on the server

return: a flag that states whether the pipeline is running

Return type
`bool`

`kill()`
Kill Cheetah pipeline process

`get_logs()`
Get logs from Cheetah process

Returns
iterator over lines of log entry

`_close_logs()`
Send Ctrl+c to the tail process

Close the log reading process

write_config(*config*)

Write cheetah configuration file.

Parameters
config (*xml.etree.ElementTree*) – cheetah xml config

delete_config()

Delete cheetah configuration file. TODO: to be implemented

_abc_impl = <_abc_data object>

class ska_pss_lmc.manager.communication_manager.**SubprocessAccess**(*pid=None*, *logger=None*)

Bases: PipelineCommunicationManager

Access Cheetah via a subprocess

process = None

connect()

Connect

_abc_impl = <_abc_data object>

disconnect()

Disconnect

start(*cmd*)

Start cheetah process

kill()

kill cheetah process

write_config(*config*)

write cheetah config

delete_config()

Delete cheetah configuration file. TODO: to be implemented

Pipeline Component Manager Configuration

class ska_pss_lmc.manager.manager_configuration.**ComponentManagerConfiguration**(*dev_name*, *logger=None*)

Bases: object

Class to store the device properties of the controlling TANGO Device to pass to the ComponentManager.

get_device_properties()

Retrieve the list of the Tango properties of the device registered within the TANGO DB.

Format the information as a dictionary where each entry is the property name and the value is the property value (as a string).

Return type
Dict[str, str]

Returns
A dictionary with the property name and the associated value.

add_attributes()

Add the device properties as attribute of the class.

Return type
None

2.2.2 Model subpackage

Health State Model

class ska_pss_lmc.model.health_state_model.**HealthStateModel**(*init_state*, *health_changed_callback*, *logger=None*)

Bases: object

A simple health model the supports.

- HealthState.OK – when the component is fully operative.
- **HealthState.UNKNOWN – when communication with the component is not** established.
- HealthState.FAILED – when the component has faulted

PYTHON MODULE INDEX

S

INDEX

M

S