

# DSH.LMC-TM Interface Design

S.Riggi - DSH.LMC, INAF OACT

*SKA LMC Harmonization Meeting  
11-13 Apr 2014, Madrid*



## ■ **TM-Dish Interface Overview**

- Current Status
- Functionalities (*see A. Marassi's presentation*)
- Common & specific commands
- Common & specific monitoring points

## ■ **Interface Design**

- Decisions & Assumptions
- Architectural view & constraints
- Guiding principles

## ■ **A case study: Scheduling commands**

- Modelling
  - ✓ Architecture view
  - ✓ Interaction view
- Implementation aspects
- Demo

## ■ TM-Dish Interface definition crucial for LMC design advances

- Interface requirements spread among LIG, LMC Scope & Resp, ICD, Tango LIG

## ■ ICD Rev 2 released in Feb. 2016

- No significant changes wrt to Rev 1
- Less detailed wrt LMC Internal ICDs and Tango LIG
  - ✓ *No moni points/commands defined*
  - ✓ *Comm protocols & architectural view left TBD*
  - ✓ *Logging/monitoring/archiving strategies are TBD*
- No advances possible wrt LMC PDR. . .

## ■ Tango LIG was very welcome!

- Tango established as the M&C framework, Tango standards & patterns under discussion
- Preliminary common commands & monitoring points given (*see Tango LIG Appendix*)
- Alignment of ICD to Tango LIG definitely needed for ICD Rev 3
- LIG & Tango LIG to be aligned as well

## ■ For DDR design we made assumptions using:

- Tango LIG
- past LMC Harmonization meetings
- ongoing discussions within the (unofficial) LMC ANT team and mailing lists

**Command argument format (see Tango LIG)**

- Input Args: Request JSON string
- Output Arg: Response JSON string

Type	Cmd	Add. Argin	Add. Argout
Self Control	Shutdown	Restart Abort Reason Comment	-
	ShutdownSubElement	subElementName Restart Abort Reason Comment	-
	StartSubElement	subElementName	-
	PowerDown	-	-
Scheduling	Revoke	revokeCmdID	-
	FlushCommandQueue	-	-
Configuration	LMCLastKnownGoodConfig	downloadURL	-
	ConfigureLogging	logConfig	-
	ProvideSelfDescription	-	SDD
Alarm	GetActiveAlarms	-	-
	SuppressNotification	skaEventName	-
	GetSuppressedAlarms	-	suppressedAlarmsList

**Command argument format (see Tango LIG)**

- Input Args: Request JSON string
- Output Arg: Response JSON string

Type	Cmd	Add. Argin	Add. Argout
Capability	AllocateXCapability	subArrayId (NA) numInstance (NA) BandId	-
	IsCapabilityAchievable	achievability	capabilityName
	SetOperatingMode	mode	-
Life-Cycle	StartUpgrade	downloadURL	-
	GetVersionInfo	-	<integrantName>_ VersionInfo
	ReportSerialNumbers	-	<HWComponent>_ SerialNumber
Maint	EnableEngInterfaces	subEl	-

**Command argument format (see Tango LIG)**

- Input Args: Request JSON string
- Output Arg: Response JSON string

Type	Cmd	Add. Argin	Add. Argout
Pointing	TrackFromAzEl	Az El timestamp	-
	TrackFromPolynomial	polynom. coeff	-
Configuration	ConfigureNoiseDiode	params	-
Power	SetPowerLevel	level	-
Safety	Stow	-	-

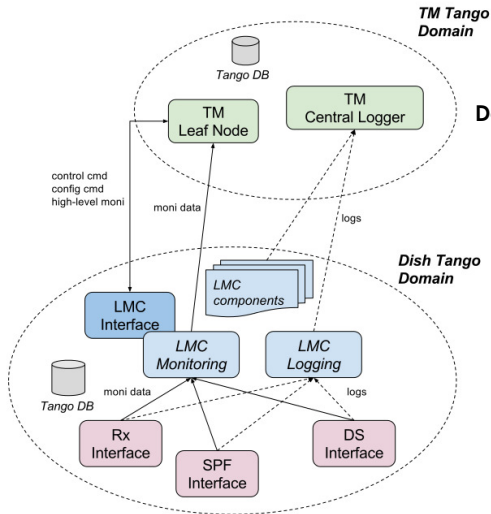
Type	Name	Data Type
Self M&C	startupProgress	DevShort
	rxStartupProgress, spfStartupProgress, dsStartupProgress	DevShort
Life-Cycle	upgradeProgress	DevShort
Usage Mode/Status	elementType	DevEnum (REAL/SIM)
	controlMode	DevShort (CENTRAL/LOCAL)
	usageStatus	DevShort (IDLE/ACTIVE/BUSY)
Mode	operatingMode	DevEnum (ENABLED/MAINTENANCE/SAFE/...)
State	operatingState	DevEnum (INITIALIZING/READY/SHUTTING-DOWN/...)
	powerState	DevEnum (UPS/LOW-POWER/FULL-POWER)
	pointingState	DevEnum (READY/TRACK/SLEW/SCAN)
Health/Capability	healthStatus	DevEnum (NORMAL/DEGRADED/FAILED)
	bandCapability(x5)	DevEnum (UNAVAILABLE/CONFIGURING/ OPERATE/...)
	rx(spf)BandCapability (x5)	DevEnum (UNAVAILABLE/STANDBY/OPERATE/...)
LRU Health	rxHealthState,rx123HealthState, rxs45HealthState,rxpuHealthState	DevEnum (NORMAL/DEGRADED/FAILED)
	spfHealthState (Va/He/Band(x5)/Ctrl)	DevEnum (NORMAL/DEGRADED/FAILED)

## Summary (see A. Ingallinera's presentation)

- SPF: About 170 physical moni points defined (He & Vacuum system, LNA voltage/current/temperature, ...)
- Rx: About 40 moni points defined (clock, controller voltage/current/temp, adc, ...)
- DS: TBD

Type	Name	Data Type
Rx	b1_samplingClock, ... rxs123_supplyVoltage,..., attenuation, ...	DevFloat/DevBool/DevLong
SPF	b1_lna_h_drainVoltage, ... b1_cs_Current,...	DevFloat/DevBool/DevLong
DS	Synch Local time, Circuit breakers Surge Protection Devices Hatches/doors open Shielded enclosure door open Limit switch(es) & Emergency stop status DS equipment temperatures Shielded enclosure internal air temperature/humidity Equipment running hours DSH Power supply/consumption/voltage/inbalance/... UPS status Time stamped estimated Az/El position Sensors used to apply local corrections	TBD



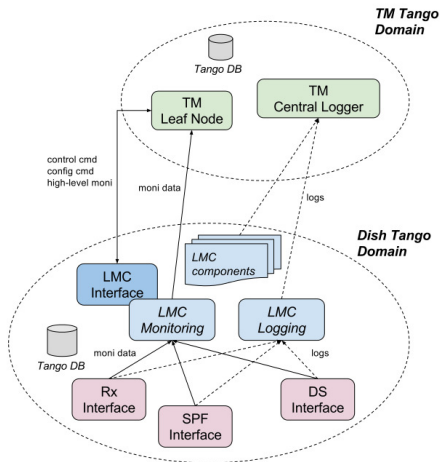


## Decisions made

- TANGO adopted for interfacing TM-DSH.LMC and DSH.LMC-Dish SE and for SKA M&C prototype development
- TM-LMC M&C interface realized by a single (or multiple) Tango Device Servers
- TM shall not directly access Sub-Elements in normal operations (allowed in Englnt mode)

**Domain:** 1 Tango DB domain for each Dish

- Sub-Elements (SE) devices hosted (TBD)
- A&A not provided by LMC
- Security: network+Access Control



## Assumptions made

- **Dynamic features (add/remove points/cmd):** None
- **Control/Cfg:** single control/cfg point for TM (LMC Interface Tango Device)  
*The LMC consists of a commercial off the shelf controller that serves as a single point of entry for all control and monitoring messages to the outside. (from L4 Req)*
  - Access to internal devices possible and ruled by access policies
- **Monitoring**
  - Summary/rolled-up moni data forwarded @ interface device from internal components
  - Drill-down or low-level moni data defined in internal LMC devices and accessible by TM
- **Logging**
  - LMC devices logging to Central & Local Logger + file
  - SE logging to Local Logger
  - Targets/Levels configurable from the LMC interface
- **Archiving/GUI/SFW Update:** TBD

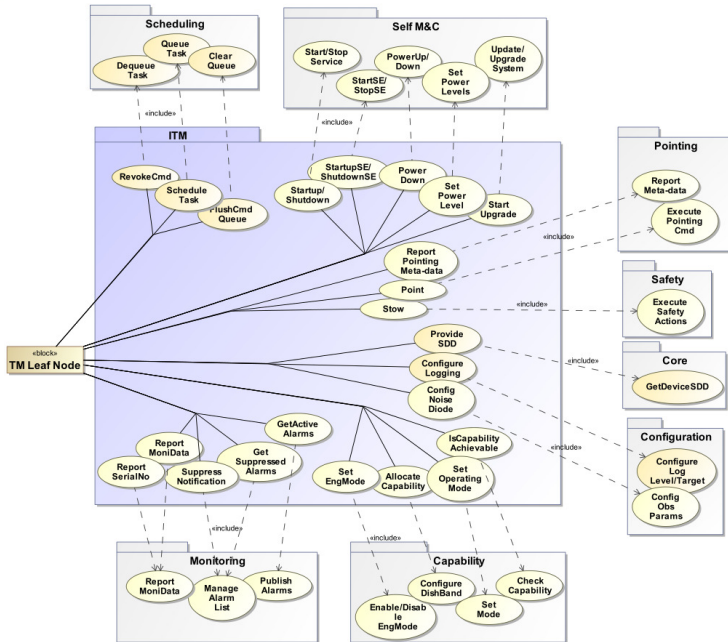
## ■ Minimize interface device complexity

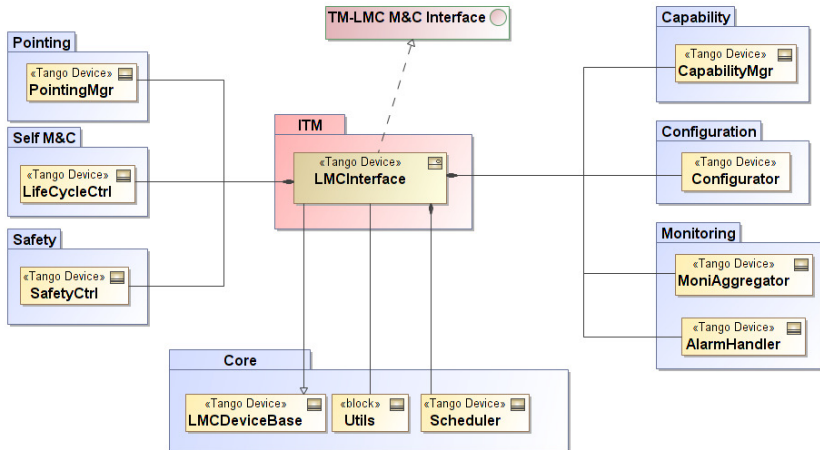
- Delegate concrete implementation of major functionalities to internal components
  - ✓ Example: Configuration (logging/device cfg), Pointing, Self Control, Life-Cycle . . .
- Avoid tons of attributes defined on the interface
- Delegate monitoring to internal devices and use attribute forwarding

## ■ Identify and re-use common functionalities across devices

- Define common low-level commands/attribute/properties in one or more LMC base devices:
  - ✓ SKA Control Model Management
  - ✓ Scheduled commands or queue management features
  - ✓ Device dynamic configuration from SDD file
  - ✓ Device alarms
  - ✓ Custom events (e.g. to GUI)
  - ✓ Standardized interface (common commands & attrs)
  - ✓ Device group features (e.g. subscribe to all points)
- Are multiple device inheritances possible in Tango?
  - ✓ Example: Partition base functionalities into distinct devices (A, B, . . .) and build a device picking only some of the base functionalities (e.g. A&C)
- Promote re-using/re-adapting of builtin Tango devices from community

# ITM Design - Functional Decomposition





# Prototype Case Study

## Scheduling

## ■ Scheduling requirements

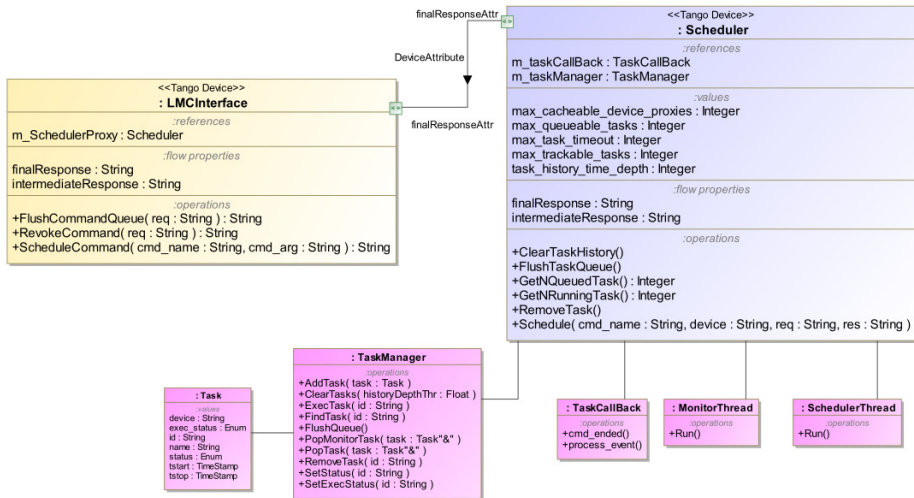
- Support these operations:
  - ✓ Execute interface commands @ future timestamp
  - ✓ Allow command queue insertion/removal/flushing
- Scheduling timing precision TBD (~ second?)
  - ✓ Pointing scheduling (@sub ms precision) to be performed by DS not by LMC
- Define use cases for scheduling (e.g. configuration, pointing, ...)

## ■ Scheduling in TANGO

- TANGO does not support timestamped commands
- Existing community components (e.g. SARDANA MacroServer) not fitting reqs?
- Ad hoc implementation considered

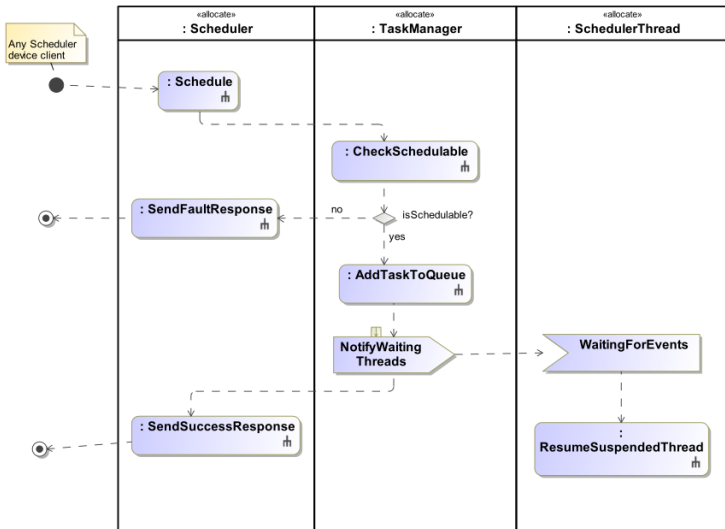
## ■ Implementation Design

- Employ a concurrent thread-safe queue pattern (recurrent, e.g. alarm system)
- *Option A*: Provide scheduling features to LMCDeviceBase
  - 😊 Devices can inherit scheduling capabilities
  - 😞 Scheduled tasks executed within the same device (handle co-located calls)
- *Option B*: Scheduler is a standalone device server
  - 😊 Simpler design, use Tango async API for command execution
- Option B followed: C++ implementation in progress (only json string cmds, task history to be done...)

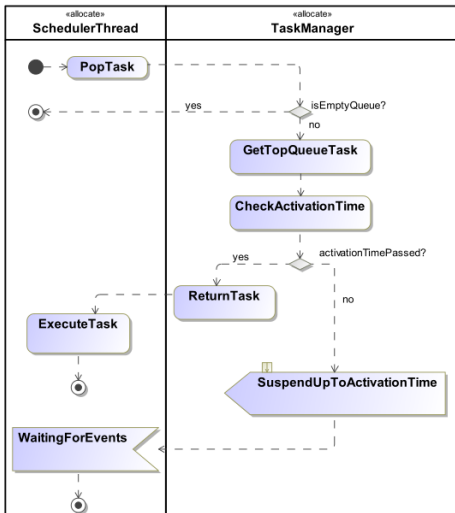




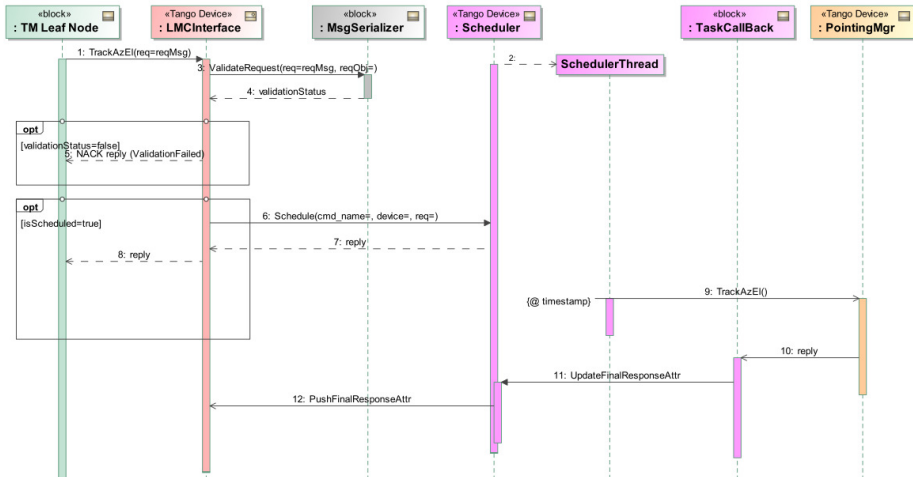
## Activity: Scheduling a task



## Activity: Executing a task



Example: Consider a scheduled track command invoked by TM



## LMC Interface device: *ScheduleTrackAzEl* command

```
Tango::DevString LMCInterface::schedule_track_az_el(Tango::DevString argin)
{
    //...
    //== PARSE REQUEST ==
    Request req;
    std::string lmc_id= this->get_name();
    if(req.Parse(argin)<0){//Invalid request
        // Log & Make fault response
        return argout;
    }

    //== CHECK SCHEDULE OPTIONS ==
    if(!req.IsScheduled() || !req.IsSchedulable()){//Missing/invalid time args
        // Log & Make fault response ...
        return argout;
    }

    //== SCHEDULE TASK ==
    std::string sched_reply= ScheduleCommand("TrackAzEl",argin);
    argout= CORBA::string_dup(sched_reply.c_str());
    return argout;
}

} //close ScheduleTrackAzEl()
```

Request parsing & validation

Helper method to invoke the Scheduler command:  
**ScheduleTask**([req, device,cmd\_name])

**TrackAzEl** is the non-scheduled command version

## Scheduler device: *ScheduleTask* command

```
Tango::DevVarLongStringArray*
Scheduler::schedule_task(
    const Tango::DevVarStringArray *argin)
{
    //== INIT & VALIDATE CMD ARGS ==
    //..

    //== CHECK SCHEDULE CONDITIONS ==
    //Check command existence, act/exp
time...
    //Check number of queued/history tasks
    //Check number of cached devices
    //..

    //== ADD TASK TO QUEUE ==
    Task task(cmd_id,
               cmd_name,
               device_name,
               T_start_local,t_end_local,
               din);
    m_TaskManager->AddTask(task);
    //..
}
```

```
void SchedulerThread::Run()
{
    Task task;

    while(!m_stopThread){
        //Pop task
        if ( (dev->m_TaskManager)->PopTask(task)
        <0 )
            continue;

        //Execute task
        if((dev->m_TaskManager)->ExecTask(task)
        <0 )
            continue;
        }//end infinite loop
    }//close Run()
}
```

Task is executed asynchronously using Tango client async API

## TaskCallBack

```
void TaskCallBack::cmd_ended(Tango::CmdDoneEvent* event)
{
    //Get event err status & msgs
    int status= eSUCCESS;
    if(event->err) {
        status= eFAILED;
        //...
    }
    //Get access to event info data & parse to Response
    //...

    //Update cmd status/response in queue & history
    //...

    // Update the FinalResponse attr & push event
    (dev->m_mutex)->lock();
    *(dev->attr_finalResponse_read)= CORBA::string_dup(event_data.
c_str());
    dev->push_change_event ("FinalResponse", dev-
>attr_finalResponse_read);
    (dev->m_mutex)->unlock();
}
```

This attr is forwarded on the LMC interface device.

DEMO

# DEMO: Device startup

The screenshot displays a Linux desktop environment with a terminal window open. The terminal shows the execution of TANGO Manager (version 6.6.6) and the startup of the TANGO Control System. A control dialog box titled "riggi-vpcs3j1e Control" is overlaid on the terminal, showing the status of 4 controlled servers. The dialog includes buttons for "Start New", "Start All", "Stop All", and "Display All". The server status is as follows:

- Level 1: SelfMonitoring/1 (Not Controlled)
- Level 2: PointingMgr/1 (Controlled)
- Level 3: Scheduler/1 (Controlled)
- Level 4: LMCInterface/1 (Controlled)

The terminal output shows the following commands and their results:

```
[riggi@riggi-VPC ~]$ TANGO Manager - 6.6.6 - Thu Sep 10 00:00:00 CEST 2009
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
DSH_LMC/
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
MySQL Tango Database
localhost:10000
Miscellaneous
riggi-vpcs3j1e
CMakeLists.txt
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
CMakeLists.txt
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
bash: cd: I: No such file or directory
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
dev->command_line tango/admin/riggi-vpcs3j1e
dev->command_line tango/admin/riggi-vpcs3j1e
dev->command_line tango/admin/riggi-vpcs3j1e
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
[riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$ [riggi@riggi-VPC ~]$
CMakeLists.txt
[riggi@riggi-VPCSA3J1E ITM]$ [riggi@riggi-VPCSA3J1E ITM]$
[riggi@riggi-VPCSA3J1E ITM]$ cd LMCInterface/
[riggi@riggi-VPCSA3J1E LMCInterface]$ ls
ClassFactory.cpp LMCInterfaceClass.h LMCInterface.h LMCInterface.xmi Makefile-
CMakeLists.txt LMCInterface.cpp LMCInterface.h~ main.cpp pntArg.dat
LMCInterfaceClass.cpp LMCInterface.cpp~ LMCInterfaceStateMachine.cpp Makefile pntArg.dat~
[riggi@riggi-VPCSA3J1E LMCInterface]$ gedit pntArg.dat &
[2] 7549
[riggi@riggi-VPCSA3J1E LMCInterface]$
```



# DEMO: Schedule a task

The screenshot displays a desktop environment with a terminal window and a configuration panel. The terminal window, titled "Terminal" and "Jive 6.7 [localhost:10000]", shows the command path: `Server:/Scheduler/L/Scheduler/DSHLMC/Scheduler/id1`. The configuration panel, titled "Device Panel [DSHLMC/LMCInterface/id1]", has tabs for "Commands", "Attributes", "Pipe", and "Admin". The "Commands" tab is active, showing an "Argin value" field with a complex string: `'(\Id':'cmd_no1','source_id':'TELMGT/LeafNode/id1','retry_number':3,'type':'?', 'activation_time':'2016-04-06T11:36:32+02:00','expiration_time':'2016-0`. Below this, there are fields for "Argin Type" and "Argout Type", both set to "DevString". A list on the left includes "ScheduleTrackAzEI" as the selected command. At the bottom of the panel are buttons for "Show description", "Execute", and "Plot". The terminal window also shows some output, including "RcsId" defined but not used.

Terminal window content:

```
Server:/Scheduler/L/Scheduler/DSHLMC/Scheduler/id1
[ 30%] Build
/home/riggi/
not used [-W
static cons
```

Device Panel [DSHLMC/LMCInterface/id1] content:

Argin value: quotes needed for string with space or special char  
'(\Id':'cmd\_no1','source\_id':'TELMGT/LeafNode/id1','retry\_number':3,'type':'?', 'activation\_time':'2016-04-06T11:36:32+02:00','expiration\_time':'2016-0

Argin Type: DevString  
Argout Type: DevString

Commands list:

- Init
- RemoveAttr
- RemoveAttr
- RestoreDevConfig
- Revoke
- ScheduleTrackAzEI**
- State
- Status

Buttons: Show description, Execute, Plot

# DEMO: Revoke/Flush tasks

The screenshot displays a Jive 6.7 terminal window with a 'Device Panel' for 'DSHLMC/LMInterface/id1'. The panel has four tabs: 'Commands', 'Attributes', 'Pipe', and 'Admin'. The 'Commands' tab is active, showing a list of commands on the left, with 'GetQueuedCommandInfo' selected. The main area shows the configuration for this command: 'Argin Type' and 'Argout Type' are both set to 'DevString'. There are three buttons: 'Show description', 'Execute', and 'Plot'. The terminal window shows a shell prompt and some output, including a message about 'RcsId' being defined but not used.

```
Server: Scheduler/1/Scheduler/DSHLMC/Scheduler/id1
Device Info
-----
Device:      dshlmc/scheduler/id1
Type_id:    IDL:Tango/Device_5:1.0
Iiop_version: 1.2
Host:       151.97.17.296 (151.97.17.296)
Alternate addr.: 10.0.0.45
Alternate addr.: 172.17.0.1

Device Panel [DSHLMC/LMInterface/id1]
Commands  Attributes  Pipe  Admin
Argin value: quotes needed for string with space or special char
'({!id!:'cmd_no1','source_id':'TELMGT/LeafNode/id1','retry_number':3,'type':'?',!activation_time!:'2016-04-06T13:48:32+02:00','expiration_time!:'2016-04-23T23:59:59+02:00'})'
GetQueuedCommandInfo Argin Type Argout Type
Init DevString DevString
RemoveAttr
RemoveAttrs
RestoreDevConfig
Revoke
ScheduleTrackAzEl
State
[
]

Show description
Execute
Plot

Clear history Dismiss

-- Up-to-date: /home/riggi/Analysis/SKAProjects/ska-dsh_lm_install/include/SDD.h
[riggi@riggi-VPCSA3J1E ska-dsh_lm_build]$
```

# DEMO: Execute task

The screenshot displays the Jive 6.7 web interface for a device configuration. The top navigation bar includes 'Applications', 'Places', 'System', and system status information: 'mer apr 6, 11:55:57' and '17°C'. The main window title is 'Jive 6.7 [localhost:10000]'. The breadcrumb path is 'Server:/LMCInterface/1/LMCInterface/DSHLMC/LMCInterface/id1'. The left sidebar shows a tree view with 'Server' selected, containing 'LMCInterface' and 'LogConsumer'. The 'Device Info' panel on the right lists the following details:

- Device: dshlmc/lmcinterface/id1
- type\_id: IDL:Tango/Device\_5:1.0
- liop\_version: 1.2
- host: 151.97.17.296 (151.97.17.296)
- alternate addr.: 10.0.0.45
- alternate addr.: 172.17.0.1
- port: 57399
- Server: LMCInterface/1
- Server PID: 20061
- Exported: true
- Last exported: 6th April 2016 at 11:44:39

The 'Device Panel [DSHLMC/LMCInterface/id1]' is active, showing the 'Commands' tab. The 'Argin value' field contains a complex JSON string: `"{"id":"cmd_no1","source_id":"TELMGT/LeafNode/id1","retry_number":3,"type":"T","activation_time":"2016-04-06T11:55:10+02:00","expiration_time":"2016-04-06T11:55:10+02:00"}`. The 'Argout Type' is set to 'DevString'. A list of commands is visible on the left, with 'ScheduleTrackAzEl' selected. The 'Execute' button is highlighted in blue. Other buttons include 'Show description', 'Plot', 'Clear history', and 'Dismiss'.