DSH.LMC-TM Interface Design

S.Riggi - DSH.LMC, INAF OACT

SKA LMC Harmonization Meeting
11-13 Apr 2014, Madrid

Outline

- **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

ITM Overview - Status

- **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

ITM Overview - Common Commands (I)

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

ITM Overview - Common Commands (II)

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	-	-

ITM Overview - Summary Moni Points

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)

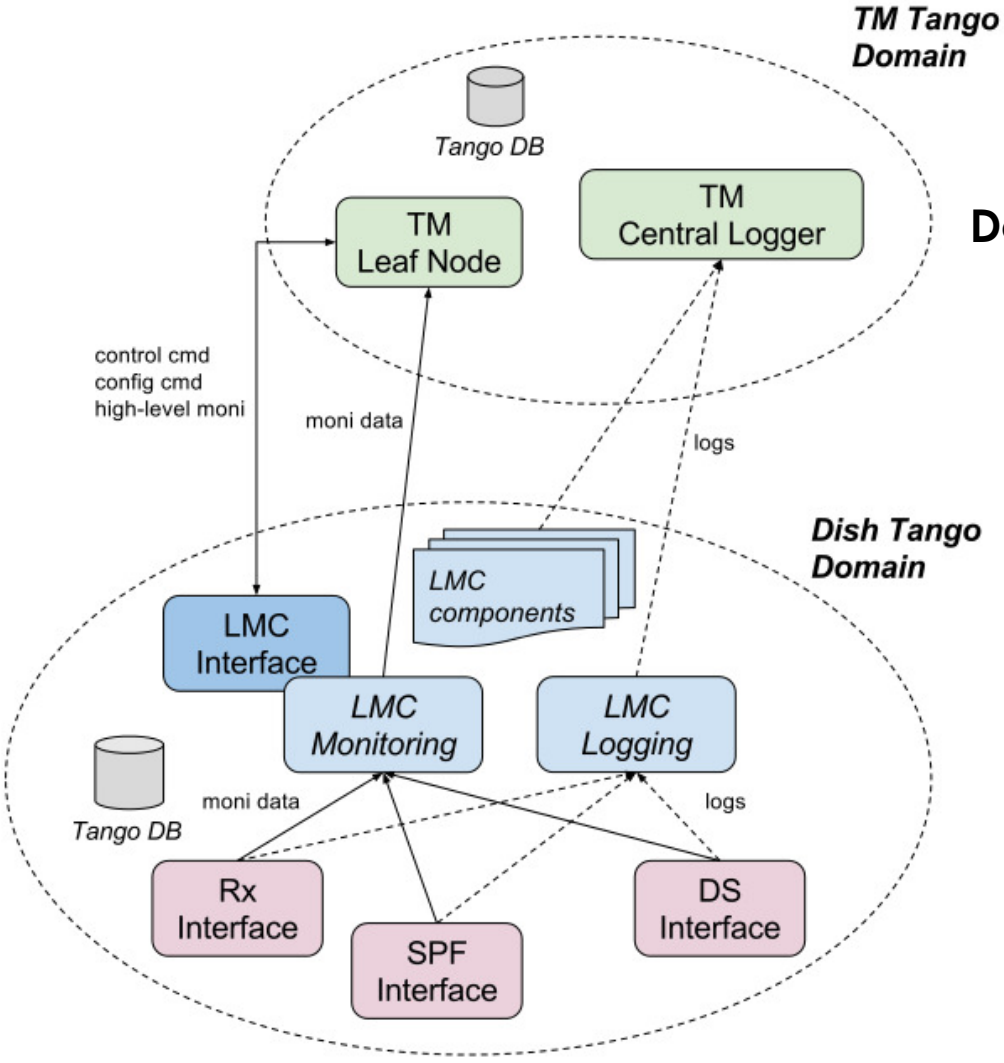
ITM Overview - Dish Moni Points

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

ITM Design - Architectural View



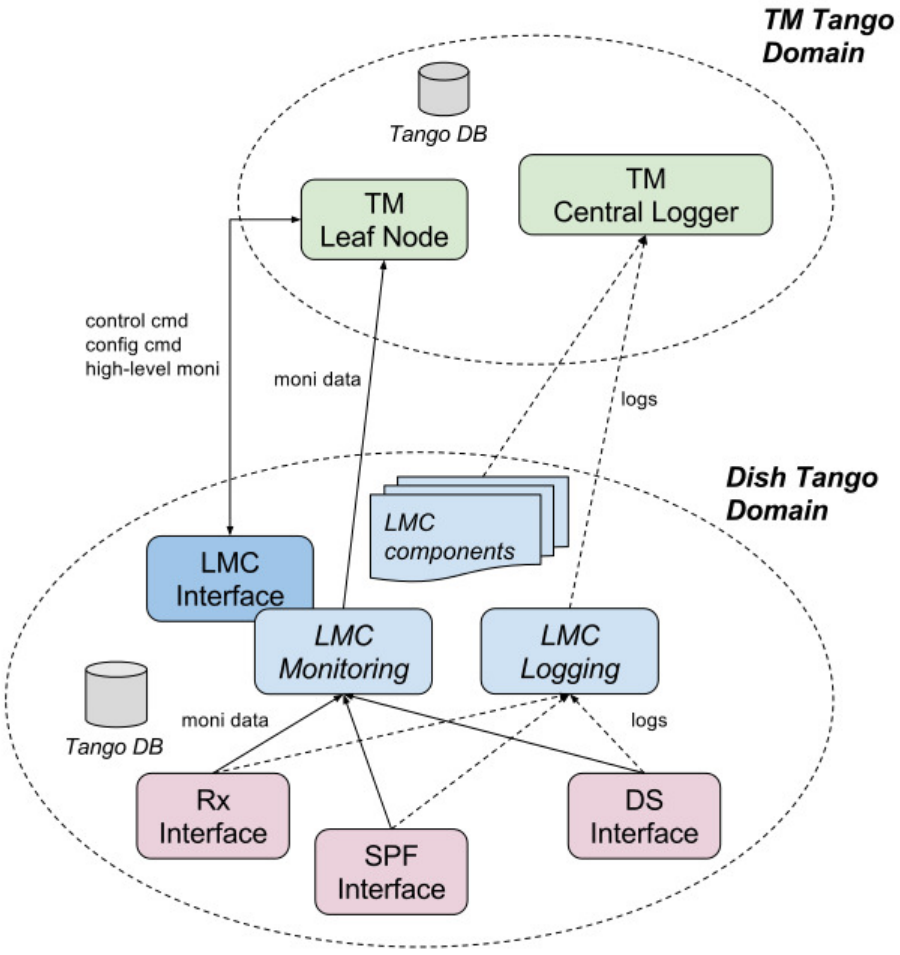
Decisions made

- 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 EngInt mode)

ITM Design - Architectural View

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

ITM Design - Guiding principles

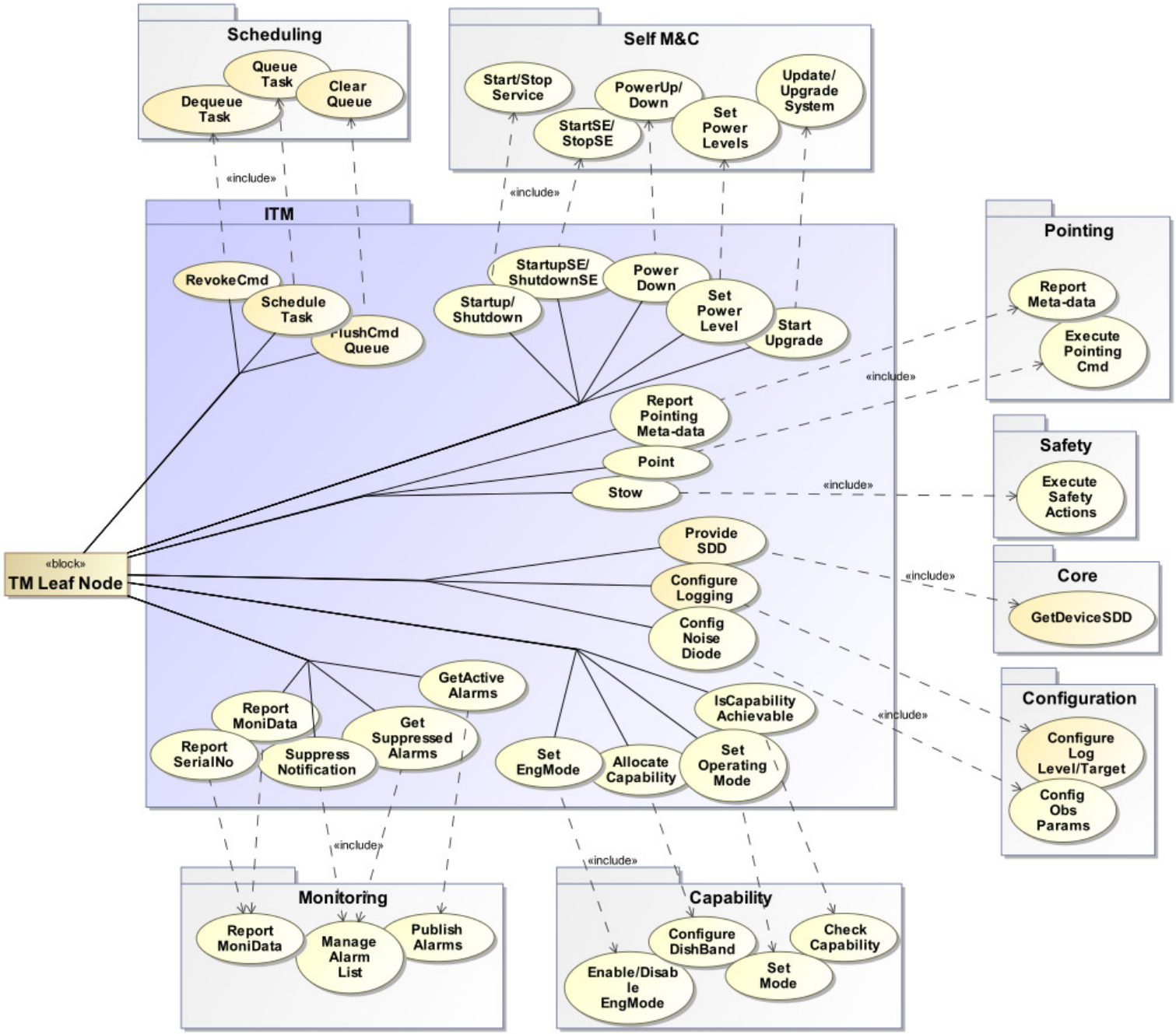
Minimize interface device complexity

- 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

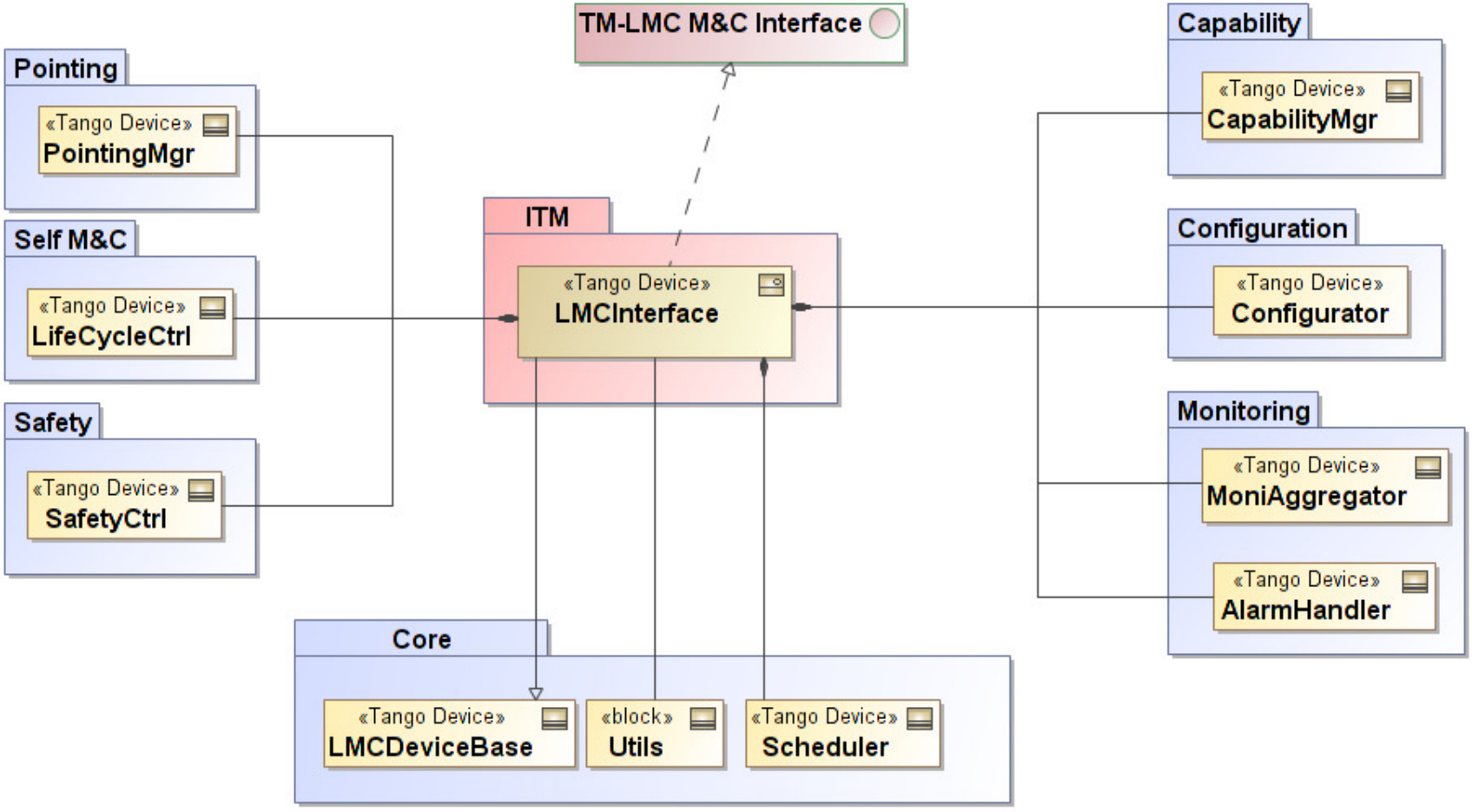
Identify and re-use common functionalities across devices

- 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



ITM Design - High-Level Architecture



Prototype Case Study
Scheduling

Scheduling Design

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, . . .)

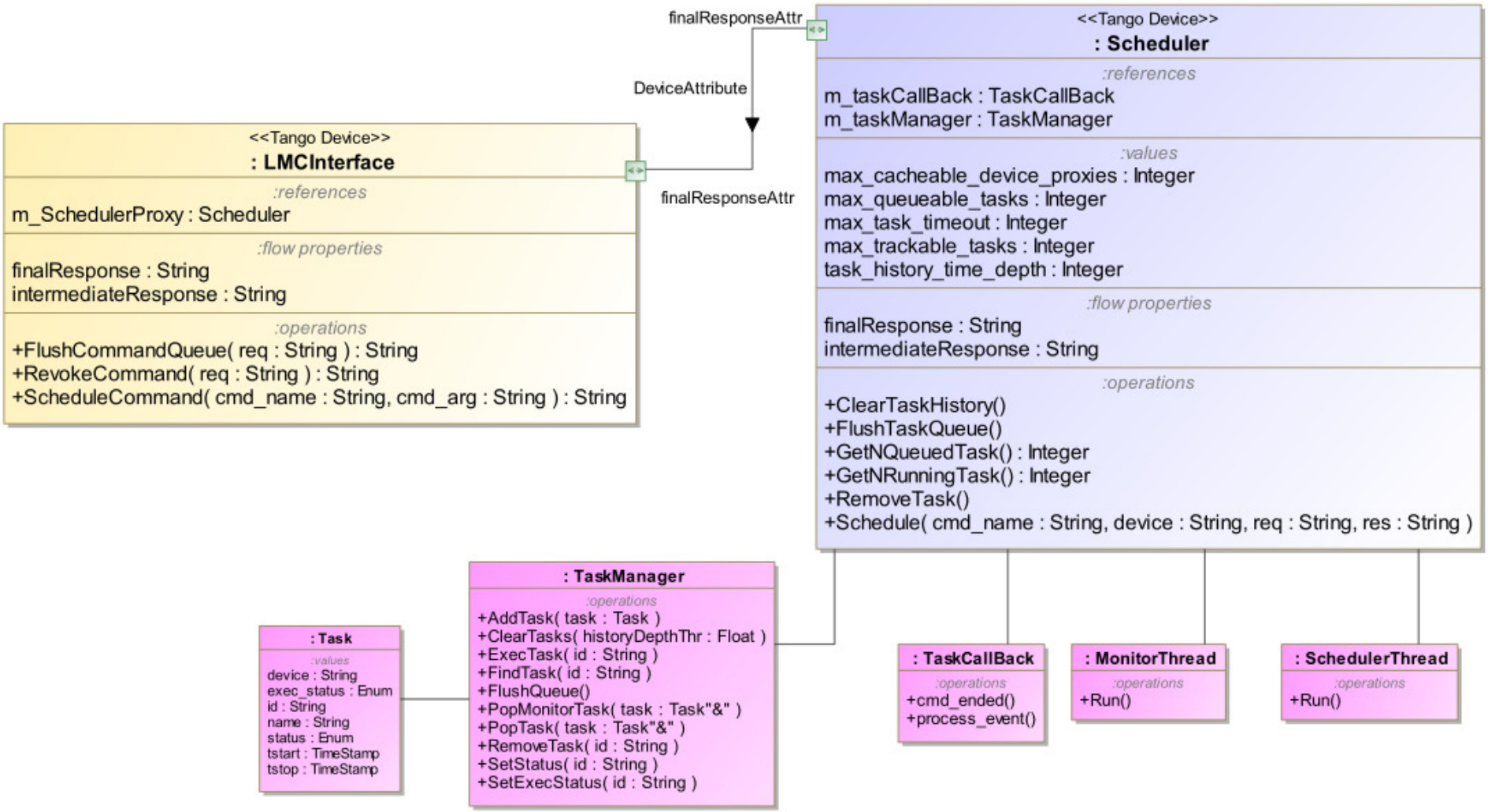
Scheduling in TANGO

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

Implementation Design

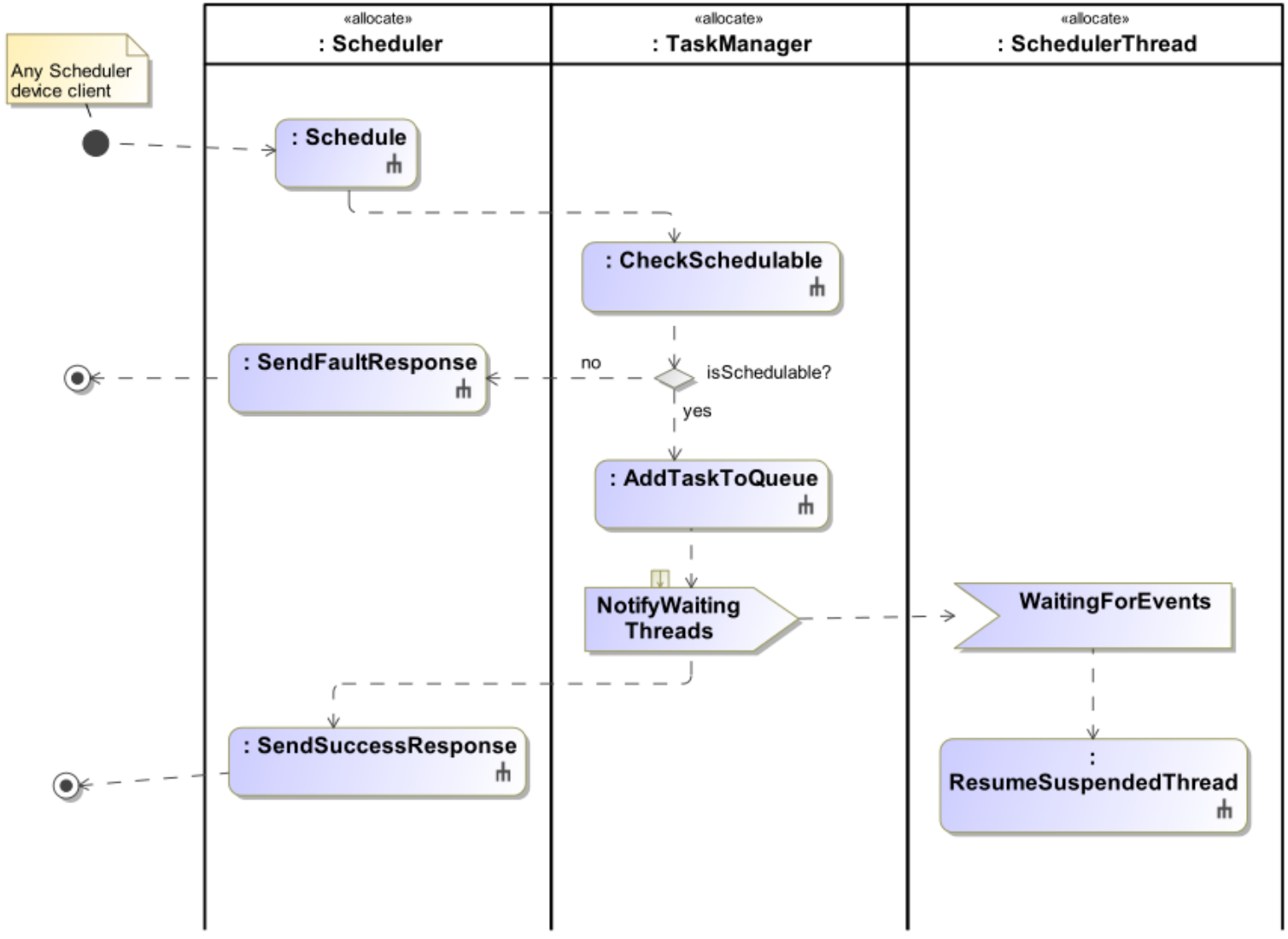
- Employ a concurrent thread-safe queue pattern (recurrent, e.g. alarm system)
- *Option A*: Provide scheduling features to LMCDeviceBase
  - ☺ Devices can inherit scheduling capabilities
  - 😐 Scheduled tasks executed within the same device (handle co-located calls)
- *Option B*: Scheduler is a standalone 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. . .)

Scheduling Design

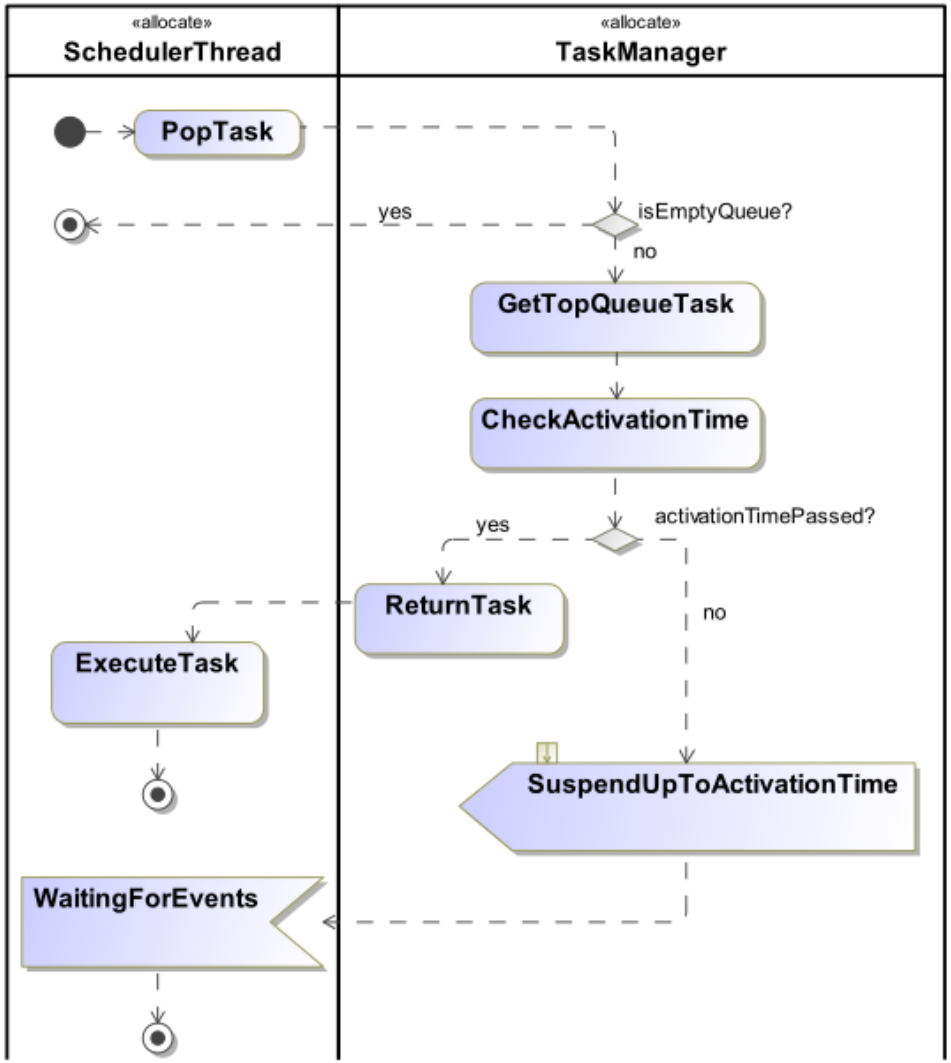


Scheduling Design - Interaction view

Activity: Scheduling a task

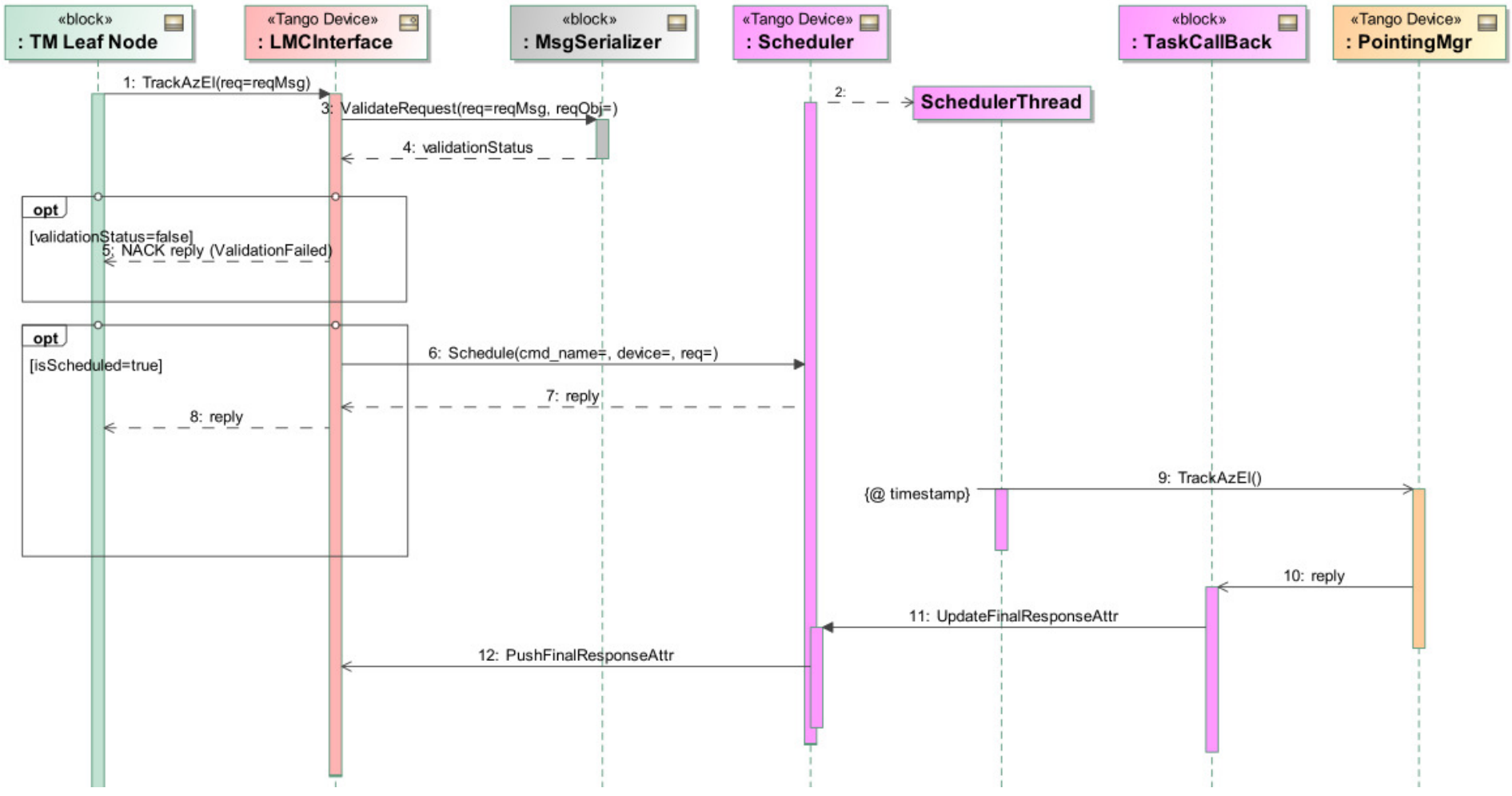


Activity: Executing a task



Scheduling Example

Example: Consider a scheduled track command invoked by TM



Scheduling Example - Sample code

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

Scheduling Example - Sample code

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

Scheduling Example - Sample code

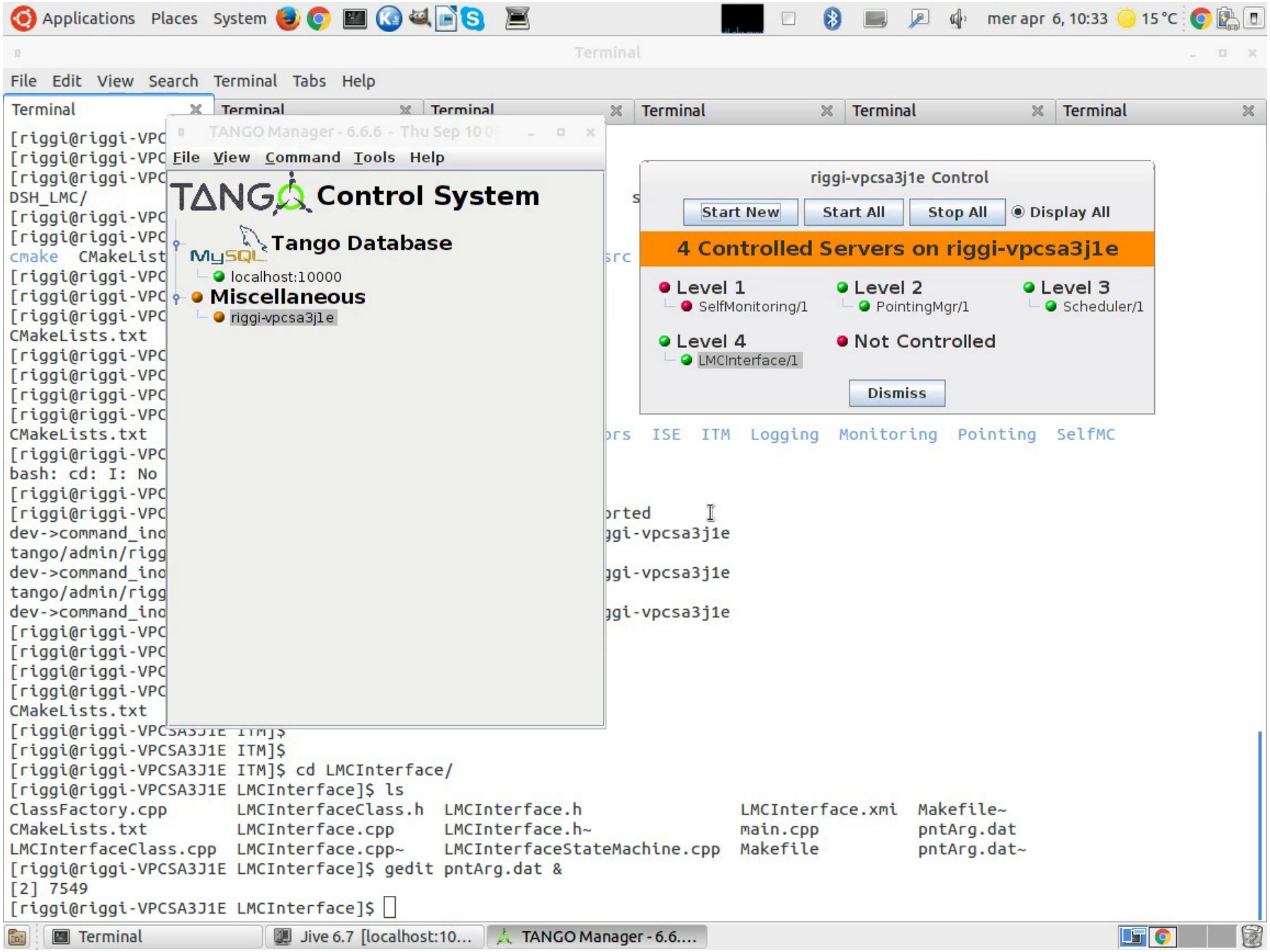
TaskCallBack

```
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();
}
```

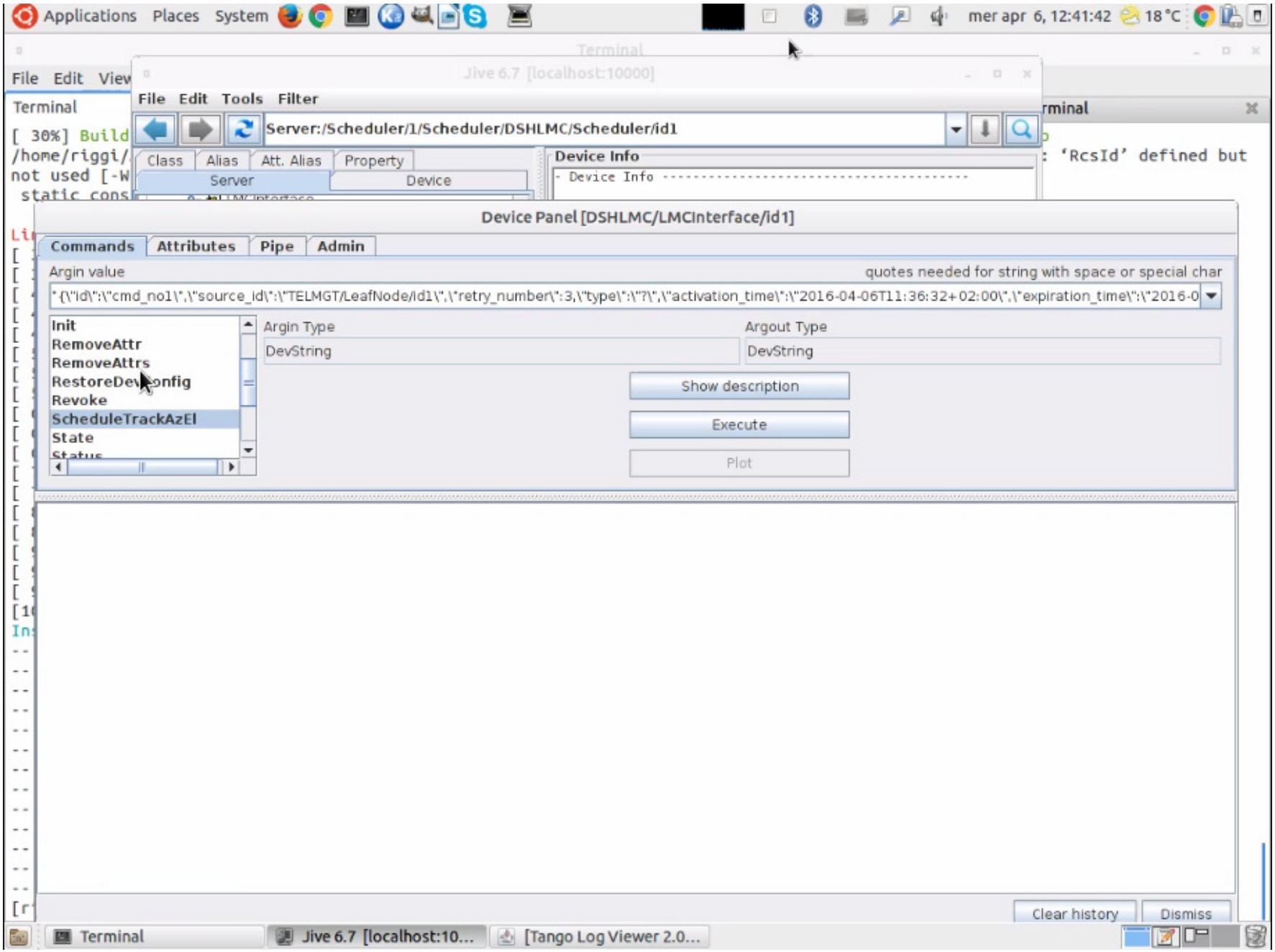
This attr is forwarded on the LMC interface device.

DEMO

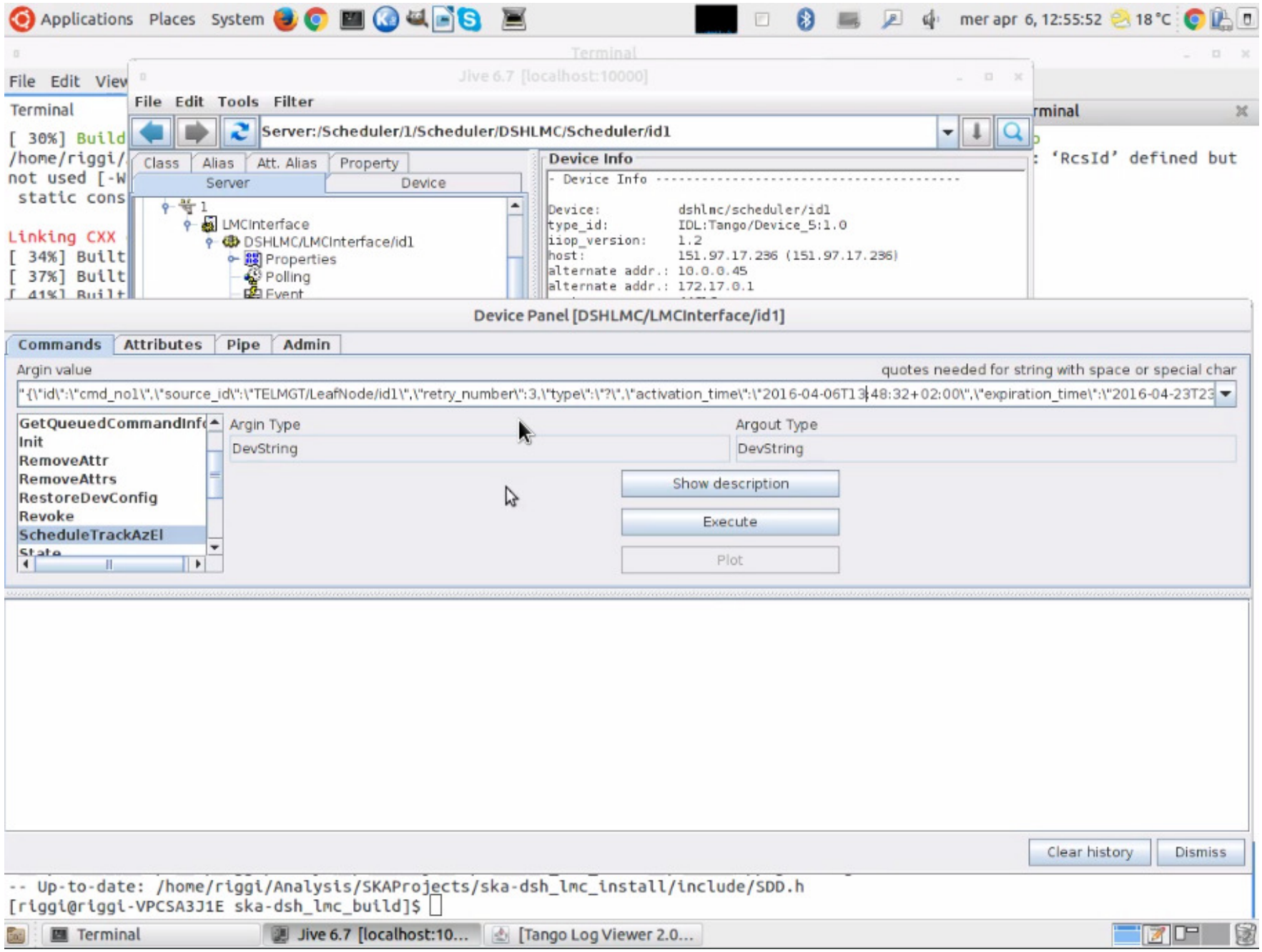
DEMO: Device startup



DEMO: Schedule a task



DEMO: Revoke/Flush tasks



DEMO: Execute task