SKA Project Memo

Mapping SKA ICDs to Tango Control System

2015 August 5

C.Baffa, E.Giani

Abstract

Following the SKA TM-LMC Trieste workshop decision to use Tango Control System as the standard framework for Monitoring and Control, we decided to explore some of the details of a possible implementation of SKA ICDs in terms of Tango structures.

We strongly hope this to be a first step toward a common set of Tango device driver classes shared with as much SKA software as possible. In this project memo we will speak about the TM - Element-LMC interface, yet most of this structure can be used at a lower level.

Arcetri Technical Report 3/2015

1 Introduction

This document has been planned following the agreement, at the SKA TM-LMC Trieste workshop, to use Tango Control System as the standard framework for Monitoring and Control. We decided to explore a possible implementation of SKA ICDs in terms of Tango structures.

1.1 Assumptions and Guidelines

The information used to compile this document comes from part of the SKA documentation [i] and Tango manuals[ii].
The first document collects the main guidelines and requirements for the TM-LMCs interactions. We would like to the use most of the protocol and classes described in lower level interaction as well, such as in the Element – Sub-elements and Sub-element – Components interactions.

1.2 Extension to lower level interfaces.

Most structures and ideas of the TM-Element communication can be extended to the lower level interactions. As an example, we would like to cite the Status and Mode attributes, where an inheritance matrix between state attributes and portions of SKA, reports as follows in Table 1 (adapted from a Tim#4 presentation by S.V.[iii]).

Indicator	Acc.	Element	Sub-element	Component	Sub-array Capability
Type	r/o r/w	REAL, SIMULATED, NOT-FITTED (?)	same	same	-
Control Mode	r/w	TM, LOCAL	same	same	-
Operational Mode (Admin. Assigned)	r/w	ENABLED, DISABLED, MAINTENANCE, TEST, SAFE	same	same	-
Operational State	r/o	INITIALIZING, READY, ERROR, OFF, OFF-DUTY, SHUTTING-DOWN, UNKNOWN	same	same	-
Health Status	r/o	OK, DEGRADED, FAILED	same	same	same
Usage Status	r/o	IDLE, USED	same	same	same
Redundancy	r/o	ACTIVE, STANDBY	same	same	-

Table 1: Compatibility matrix between state attributes and portions of SKA

2 Naming conventions

In a project of the size of SKA it is imperative to have clear cut naming conventions. An effort in this direction is under way[iv]. Here we will give only an example of a possible schema.

2.1 Tango naming Schema

The device is the heart of the TANGO device server model. A device is an abstract concept, it can be a piece of hardware, a collection of hardware a logical device, or a combination of all these. Each device has a unique name in the control system (and eventually one alias). Within Tango, a four field name space has been adopted consisting of:

[host:port/]domain/family/member

In this hierarchical notation, the member specifies which element within a family. The family specifies which kind of equipment within a domain. The domain groups devices related to which part of the full system they belongs to. The host corresponds to a Tango *domain*, but can be accessed remotely, also from devices belonging to other domains.

Tango name identify a specific hardware, so it makes sense to group together similar devices belonging to different device portions, however, for management reasons, also the opposite choice can be taken. This naming flexibility should be exploited by a wise whole-SKA nomenclature schema to simplify the overall management.

To access specific propriety the syntax is:

[host:port/]domain/family/member->property

2.2 A possible SKA nomenclature mapping

A possible mapping to SKA nomenclature can be:

Element:port/Sub-Element/Component/Instance

In a similar way, we can refer to an attribute or a property with the notations:

Element:port/Sub-Element/Component/Instance/attribute

Element:port/Sub-Element/Component/Instance/attribute->property

This nomenclature assume to have a Tango database server for each Element. For larger elements this can be suboptimal, and a partition can be devised.

Some examples can be:

csp:8081/pss/computing_node_pipeline/node_0739

low_dishes:8085/dishes/azimuth_motor/motor_001

csp:8081/low_beam_former/fpga_board_control/board_0354_chip_3

2.3 A second possible SKA nomenclature mapping

As the simple nomenclature devised in the previous section can be sub-optimal, we can consider an alternate mapping. In this approach we can extensively apply the Tango Alias functionality.

The basic idea is to use the Tango names in a very hardware-related way.

1. Domain is mapped to the kind of physical device: blade_server, beam_forming_FPGA, main_motors, pumps, etc.
2. Family is mapped to to the specific component: pss_node, k206_temperature_sensor, t800_power_switch, mid_filter_board, etc.
3. Instance is mapped to the specific item: rev_A_board_25, pt100_sensor_1023, fan_23.

To these physical hardware description we can associate, at least for larger objects a human comprehensible alias: capetown_dish_34, room_34_temperature_sensor, mid_channelizer_board_81, PST_engine, etc.

2.4 SKA Components and Capabilities

As SKA is a very large and complex instrument, it has been divided into components and capabilities as an aid to comprehension. the structure has been described in many global documents, whereas here will only note the SKA peculiar use of the term Capability.

In the SKA environment, 'capability' refers to any ‘logical grouping’ (functionality/product) provided by an Element to sub-divide or configure the output of that Element. Each of these logical groupings are referred to as a “capability” for which the LMC of each Element will select and allocate resources to form the capability from its available components and report health-status on that capability to the TM. TM will coordinate the allocation of Capabilities for an Experiment and Element LMC will report Capabilities to the TM

2.5 Aliases

Within Tango, each device or attribute can have an alias name. An alias is simply an open string stored in the database and it can be used every time a device or an attribute name is requested by the API’s. The rationale of the alias is to give device or attribute name a name more natural from the physicist point of view.

In a previous section (Error: Reference source not found) we have described a possible use of this capability to implement a SKA specific, human-readable, nomenclature schema.

3 Status Variables and properties

In SKA environment there are some variables which define the global status of each Element. The main internal variable is the Operating State which has a direct mapping to the Tango State attribute.

The full list of SKA status variable is:

- Control Mode;
- Operating Mode;
- Operating State;
- Health Status;
- Usage Status;
- Capability Health Status ;
- Sub-array State (Applicable for sub-arrays only).

In this chapter we will propose a schema for the implementation of SKA status as Tango attributes.

3.1 Operating State (SKA) versus State (Tango) attributes.

In Tango, there are two variables which refer to the logical state of the device (State) and a string description of the current state value (Status). In SKA the corresponding value has the name of Operating State.

SKA Operating State has seven possible values, while Tango state has 14 possible values (see Table 2). We have mapped the SKA values to the nearest Tango ones. *As Tango Status variable is a free-form string, it will always contain the SKA State name*.

SKA		Tango
OFF	This is a Powered off state.	**OFF**
READY:	This suggests that the Element is ready to operate	**ON**
SHUTTING-DOWN GOING-OFF-DUTY:	This is a transient state in which the Element is shutting down or going Off-Duty.	**MOVING**
ERROR:	An Element reports an 'Error' state when it detects a problem that affects its ability to accept certain commands or execute certain processes/operations.	**FAULT**
OFF-DUTY:	Special non-operational state in which Entity has been placed to reduce power consumption.	**STANDBY**
UNKNOWN:	TM is not aware of actual state of Element.	**UNKNOWN**
INITIALIZING:	This is a transient state in which the Element exists when it is starting up its processes.	**INIT**

Table 2: Proposed correspondence between SKA Operating State and Tango State.

The ambiguity in the Moving status can be resolved with an auxiliary status variable, *Moving-Status*.

3.2 Status and Mode variables

In the SKA framework, beside the Operating State, there are 5 others main variables which describe the basic properties of an Element (and, lower level entities). We propose to implement them as Tango attributes of *enum* type. Table 3 shows the SKA-Tango mapping.

SKA			Tango name
Element Type	*Real*	a Real Element is connected	Element_Type
	Simulated	a simulator is connected in place of a real Element.	
	Standby	a Element/Simulator is connected and is used as a backup device for providing redundancy	
	Not-Fitted	the Element/sub-element is not fitted.	
Control Mode	*Central*	Element is under TM control.	Control_Mode
	Local	The element is under manual/local control of the LMC.	
Operating Mode (alternative to Administrative Mode)	*Enabled*	The Element is allowed to perform activities.	Operating_Mode
	Disabled	The Element is intentionally excluded from performing activities or participating in an operation	
	Maintenance	The Element is reserved for maintenance activities like diagnostics, configuration changes, commissioning	
	Test	The Element is reserved for setup / testing activities.	
	Safe	This mode imposes functional restrictions that increase resilience to equipment damage	
Health Status	*Normal*	Element is in normal working condition	Health_Status
	Degraded	Element is functioning in degraded condition when subset of its functionality is compromised or unavailable.	
	Failed	Implies when there is major failure that prevents Element to perform its function.	
	NotOperable	Element is not available for observations due to missing dependencies.	
Usage Status	*Idle*	Element is not in use but it is available for use.	Usage_Status
	Active	Element is performing in observations, but it still has operating capacity to provide for more observations.	
	Busy (?)	Element is performing in one or more observations and cannot participate in any other observations	

Table 3: Synopsis of proposed mapping of SKA Status and Mode variable to Tango Attributes.

To that status parameter a recent memo[v] adds the variables of Table 4.

To fully specify SKA status we propose to add the TANGO-only status variables defined in Table 6.

Extending the Status variable paradigm to entities smaller than Elements, we need to take into account a slight different meaning of status variable. For instance not all are meaningful (see, for instance, Table 1) or all have the same Read/Write status on attribute (see Table 5).

SKA			Tango name
Administrative Mode (alternative to operating Mode)	*Enabled*	Element can be administered	Administrative_Mode
	Disabled	(Default) Normal Operations	
	Maintenance	Element under maintenance	
	Not-Fitted	the Element/sub-element is not fitted.	
Simulated Mode	*Real*	The element is a real hardware	Simulated_Mode
	Local	The element is simulated	
Observing Mode	*Idle*	The element is idle.	Observing_Mode
	Imaging	The element is observing images	
	PSS	The element is performing PSS	
	PST	The element is performing PST	
	VLBI	Performing VLBI observations	
	Transient search	The element performs *transient search*	
Test Mode	*Normal*	Element is in normal working condition	Test_Mode
	Test	Element is under test	
Redundancy State	*Active*	Redundancy is enabled	Redundancy_State
	Standby	Element is in standby	

Table 4: Proposed status variable, added to Table 3 ones.

3.3 Programmed parameters

Each Element (and sub-elements, etc.) has its own operating set of parameters defined in the various ICD documents. Such parameters define unequivocally the working status and the operations performed by the Element. They come from TM on the basis of the observation to be performed.

The Operating parameters are defined to have many different formats. Those formats can be mapped to the corresponding parameter type provided by Tango-Control, as specified in Table 7.

Each parameter can be defined as a *Scalar*, *Vector* or 2-dimension *Array*, in Tango notation: Tango::SCALAR , Tango::SPECTRUM , Tango::IMAGE. For attribute of the Tango::IMAGE data format, all the data are also returned in a one dimension array. The first array is value[0],[0], array element X is value[0],[X-1], array element X+1 is value[1][0] and so forth.

Mode / State + values	Element	Sub-element	LRU	SW/HW components	Sub-array	Capabilities
Administrative Mode	R/W	R/W	R/W	R/W	-	-
Observing Mode	-	-	-	-	R/W	R/O
Control Mode	R/W	R/W	R/W	R/W	-	-
Simulated Mode	R/W	R/W	R/W	R/W	-	-
Test Mode	R/W	R/W	R/W	R/W	-	-
Operational State	R/O	R/O	R/O	R/O	-	-
Health State	R/O	R/O	R/O	R/O	R/O	R/O
Usage State	R/O	R/O	R/O	R/O	R/O	R/O
Redundancy State	R/O	R/O	R/O	R/O	-	-

Table 5: Read-Write mode of SKA status variables.

SKA Variable to be fully specified	Tango Auxiliary Variables		
STATE	MOVING-STATUS	*Going-Off-Duty*	Element is going Off-Duty
		Shutting-Down	Element is shutting down
		Getting-Ready	Element is going Ready
		Not-Applicable	(Default) Normal Operations
LOGGING-LEVEL	VERBOSITY	*Normal*	Normal logging level
		Debug	Debug verbosity level
		Trace	Trace all verbosity level
		All	All information logged

Table 6: Tango auxiliary variables to fully specify SKA variables

Tango available types	
Tango::DevBoolean	Boolean
Tango::DevShort	Mapped to short signed integer
Tango::DevLong	Mapped to long signed integer (4 bytes)
Tango::DevLong64	Mapped to int64 signed integer
Tango::DevFloat	Mapped to single float
Tango::DevDouble	Mapped to double precision float
Tango::DevUChar	Mapped to unsigned char
Tango::DevUShort	Mapped to short unsigned integer
Tango::DevULong	Mapped to long signed integer (4 bytes)
Tango::DevULong64	Mapped to int64 signed integer
Tango::DevString	Mapped to character string
Tango::DevState	Tango specific structure
Tango::DevEncoded	Structure: format plus character string

Table 7: Tango available parameter types

3.4 Monitoring points

Monitor points are parameters (attributes) that aremonitored periodically .
Each SKA element and sub-element autonomously monitors and reports on the status of its Monitor Points; i.e. each sub-element monitors the parameters identified as Monitored Points, then it generates Reports according to pre-configured parameters (frequency and/or thresholds) and finally it sends the Monitor Point Reports to the pre-configured destination address. Each component has its own unique list of Monitor Points, defined in the various ICDs.

In the Tango Framework there are *local* quantities (Attributes) whose values can be read by the upper control level, (in Tango nomenclature 'client' of the device server). A periodic reading time can be specified for these quantities, as well as a condition on the value which will rise an *alarm condition* (see chapter 5).

As the Tango Attributes can have all the value types of parameters (scalar and arrays of types in Table 7), it is highly unlikely that a monitor point value will not fit in one of those types.

4 Commands

In the default setting, the SKA ICDs describe mainly changes of modes, states and parameters/attributes interrogations. Other forms of direct command aren't mandatory, while each implementation has a degree of freedom to implement what is necessary.

Command name	Input data type	Output data type
State	void	Tango::DevState
Status	void	Tango::DevString
Init	void	void
DevRestart	Tango::DevString	void
RestartServer	void	void
QueryClass	void	Tango::DevVarStringArray
QueryDevice	void	Tango::DevVarStringArray
Kill	void	void
QueryWizardClassProperty	Tango::DevString	Tango::DevVarStringArray
QueryWizardDevProperty	Tango::DevString	Tango::DevVarStringArray
QuerySubDevice	void	Tango::DevVarStringArray
StartPolling	void	void
StopPolling	void	void
AddObjPolling	Tango::DevVarLongStringArray	void
RemObjPolling	Tango::DevVarStringArray	void
UpdObjPollingPeriod	Tango::DevVarLongStringArray	void
PolledDevice	void	Tango::DevVarStringArray
DevPollStatus	Tango::DevString	Tango::DevVarStringArray
LockDevice	Tango::DevVarLongStringArray	void
UnLockDevice	Tango::DevVarLongStringArray	Tango::DevLong
ReLockDevices	Tango::DevVarStringArray	void
DevLockStatus	Tango::DevString	Tango::DevVarLongStringArray
EventSubscribeChange	Tango::DevVarStringArray	Tango::DevLong
ZmqEventSubscriptionChange	Tango::DevVarStringArray	Tango::DevVarLongStringArray
AddLoggingTarget	Tango::DevVarStringArray	void
RemoveLoggingTarget	Tango::DevVarStringArray	void
GetLoggingTarget	Tango::DevString	Tango::DevVarStringArray
GetLoggingLevel	Tango::DevVarStringArray	Tango::DevVarLongStringArray
SetLoggingLevel	Tango::DevVarLongStringArray	void
StopLogging	void	void
StartLogging	void	void

Table 8: Automatically provided Tango commands.

Tango automatically provides a basic set of commands (see Table 8). These are the implementation of a set/query of parameters/attributes and basic device controls, as start/stop polling quantities, subscribe/unsubscribe change notification and logging management.

4.1 Timed set command

We consider important to add a *timed set command* to the list of default commands. As it is available a time server with a precision of at least tens of milliseconds, we propose that each group of set commands will be registered at once, but programmed only at the wall time contained in the *timed set command argument.* This command is useful mainly in the top levels of the control hierarchy, but it might come handy at deeper levels as well. In our view, this command will ease the coordination of the various portion of SKA.

5 Alarms and Logging

Alarm management is critical for the performance and maintenance of any complex system, such as the SKA components. An important portion of LMC role is the correct handling of error conditions.

5.1 Alarms in SKA

Alarm can be defined as an undesirable event - for which the urgency of the matter is such that it requires urgent intervention: either a manual (by the operator) or an automated- action to end the condition or to reduce the effect of the condition.

SKA defines two main types of alarms:

- Transient (for instance a software exception) if the system returns at once to its normal status
- Persistent (for instance an hardware failure) if an action has to be taken.

Alarms can further be categorized into five categories:

1. Communication Alarm Type - An alarm of this type is principally associated with the procedures and/or processes required to convey information from one point to another.
2. Quality of Service Alarm Type - An alarm of this type is principally associated with degradation of quality of service.
3. Processing Error Alarm Type -An Alarm of this type is principally associated with software or
4. processing fault.
5. Equipment Alarm Type - An alarm of this type is principally associated with an equipment fault.

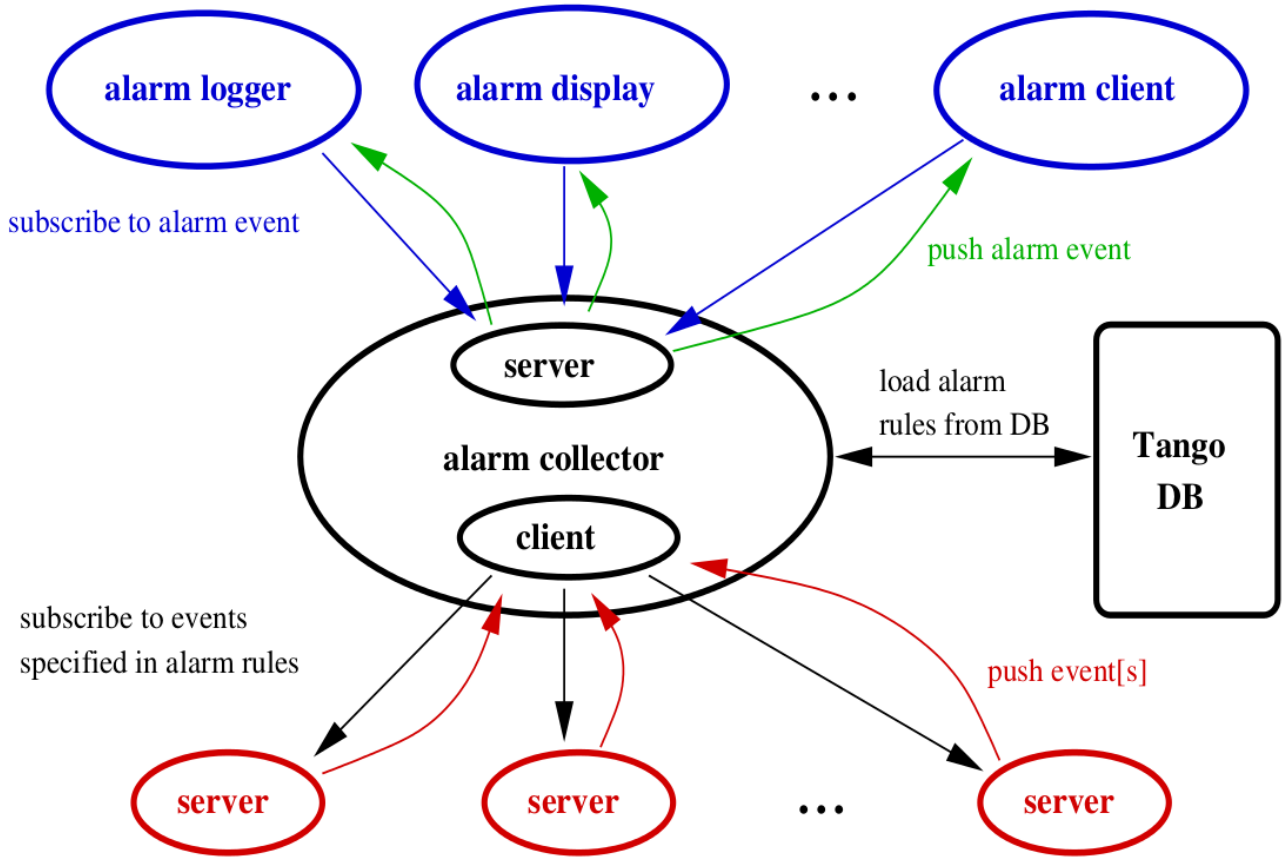

Illustration 1: Structure of Tango Alarm Collector Class.

5.2 Alarms in Tango

The Tango Control system provides a service inside the Device Server class to rise an *alarm condition* if a monitored variable is outside a programmed interval or the server code explicitly detects such a condition. Tango does not posses a general Alarm engine in its core implementation, so more complex *alarm conditions* cannot be handled in the core system.

However a fairly general Alarm handler class has been made available by one of the members of the Tango community (Elettra[vi]).

Each instance of this class receives alarm signals from up to 30 groups of device servers and, according to one or more logical expressions, programmed at run time and stored in a database, can rise one or more *alarm condition* (see Illustration 1). Henceforth we will use the term 'alarm' as a shorthand for '*alarm condition*'.

Alarm Device Server attribute

The Tango Alarm Device Server external interface implements the alarm attribute as an array of strings.

The current implementation of an alarm string is composed by some fields separated by '\t' (tab)

character, the fields are the following:

- time_stamp (seconds since 1/1/1970) of the instant when this alarm changed status
- micro seconds of the instant when this alarm changed status alarm name
- status (“NORMAL” or “ALARM”)
- acknowledge (“ACK” or “NACK”)
- count: if the status is “ALARM” the count field contains the number of subsequent events all evaluating the “ALARM” status, otherwise it is 0
- severity level (“fault”,“warning” or “log”)
- group(s) (if more than one, groups are separated by the '|' character)
- message (optional)
- new status: if it is the first time this alarm is read in the “ALARM” status, it is added the string “NEW” as the last field

Table 9 shows the required Alarm level for SKA. Up to now the Tango Alarm level is limited to three levels (LOG, WARNING, FAULT), but the extension to more levels should be not too difficult. As an alternative, we can define an attribute (Fault_Level, enum type) which can identify the fault severity.

A possible correspondence can be devised in Table 10, while the use of Fault_Level variable is detailed in the Table 11.

Severity Level	Description
Indeterminate	Alarm level used when the severity of the error condition is unknown.
Critical	This severity level indicates that a condition affecting service has occurred and immediate corrective action is required.
Major	This severity level indicates that a condition affecting service has occurred and urgent corrective action is required.
Minor	This severity level indicates that a condition has occurred that does not directly affect service, however corrective action should be taken in order to prevent a more serious fault.
Warning	Alarm level used to indicate alarm warning conditions. The warning severity level indicates the detection of a potential or impending fault that may affect service, before any significant effects have been encountered.
INFO	Information to report progress and other events of Interest.

Table 9: Required Alarm level for SKA

SKA		Tango	
Indeterminate	No information.	**NONE**	No correspondence
Critical	The device has failed. There is no workaround.	**Fault:**	An error occurred. The process signals its grade in Fault_level **Fault_level** = Critical
Major	An error has occurred. A function or operation did not complete successfully. A workaround may be possible.		**Fault_level** = Major
Minor	A condition was detected which may lead to functional degradation but the device is still fully functional.		**Fault_level** = Minor
Warning	detection of a potential or impending fault that may affect service	**WARN:**	An unexpected event occurred but it could be recovered locally
INFO	This level of logging should give information about work-flow at a coarse-grained level. Information at this level may be considered useful for tracking process flow.	**INFO:**	Provides information on actions performed

*Table 10: A possible correspondence between SKA and Tango Alarm level. To fully specify Fault level we propose the use of an auxiliary variable (*Fault_Level*).*

SKA reference	Tango Auxiliary Variables for Alarm		
ALARM-LEVEL	FAULT_LEVEL	*None*	Element is in normal working condition
		Critical	Element has a critical Alarm
		Major	Element has a major Alarm
		Minor	Element has a minor Alarm

Table 11: Tango auxiliary variables to fully specify SKA alarm

Along with the alarm notification SKA requires the following informations:

- Alarm Name
- Source of Alarm (name of element LMC and sub element/component)
- Severity Level
- Time-stamp
- Cause of Alarm
- Parameters etc.

Comparing with the Tango alarm attribute, we can build the following mapping table:

SKA Fields	Tango Fields
Time-stamp	time_stamp (second + microseconds)
--	status (“NORMAL” or “ALARM”)
--	acknowledge status
--	count
Severity Level	severity level
Alarm Name	alarm name
Source of Alarm	free form message. To be carefully specified.
Cause of Alarm	
Parameters etc.	
--	new status

SKA alarm fields mapping to Tango Alarm fields

The Tango free form message should have a clear formatting to let an easy extraction of sub-fields. A possible format can be “source-of-alarm ; cause-of-alarm ; parameter0 , parameter1 , … “.

We insert explicitly the source-of-alarm field for clarity and ease of operation, but this information can be, in principle, also obtained from the alarm name.

Alarm Device Server Commands

The Tango Alarm Device Server provides the next commands:

- ack: acknowledge alarms present in the alarm table. Alarms in status NORMAL/ACK are removed from the alarm table
- configured: read an array of configured alarms. This command accepts an argument used to filter the list on the alarm name. If the argument is 0, all the configured alarms are returned. The returned string is composed by some fields separated by the '\t' character. The fields are:
    - time-stamp of the instant when this alarm was added;
    - alarm name;
    - alarm formula (evaluated to handle the alarm condition);
    - time threshold (period of time during which the formula has to evaluate true and after which the alarm changes status to ALARM);
    - severity level;
    - group(s);
    - message;
    - actions (two Tango command separated by a `;` character);

- load: load a new alarm;
- remove: remove an alarm from the configuration. This command accepts as argument the alarm name;

LMC Guidelines document states that each SKA Element shall make provision for TM to:

- suppress a specific alarm
- clear specific alarm or all alarms for a particular Element/sub-element/component
- obtain the list of active alarms (for Element/sub-element/component)

The Tango Alarm Server commands can partially fulfil these functions and in the following table we propose a possible match.

SKA Fields	Tango Fields
Suppress	Configuring the time threshold its possible to suppress an alarm for a time.
Clear	ack alarm. Can we do this on a group base?
List of active alarms	Configured

Moreover, the Alarm Device Server provides commands to configure, add and delete alarms.

5.3 Advanced alarm handling technique

Alarm shelving

Alarm shelving is a mechanism used to temporarily suppress malfunctioning alarms in a controlled manner.

The Tango Alarm System can implement such features using the time threshold of the alarm but it seems to lack:

- a specific state for this alarm condition (for example SUPPRESSED)
- the mechanism to list the suppressed alarms which can also be used, as a reminder, to report periodically to LMC/TM the list of the suppressed alarms
- auto-re-enabling (requested ?)

Nuisance alarms

Nuisance alarms are alarms that annunciate excessively, unnecessarily, or do not return to the normal state after the correct response is taken (e.g., chattering, fleeting, or stale alarms) . The cause of such alarms are often wrong setting of the limits.

The common mitigation strategy for reduction is using dead-band in conjunction with time delay. The dead-band for an attribute is defined at the level of the Device Server (not alarm system), configuring

appropriately the minimum and maximum levels for *warning* and *alarm* properties of the specific device attribute.

State-base alarms and alarm flood

Alarm floods generally arise during a change of the operating state (such as start-up, shut-down, initialization etc.).

The alarm formula associated to each configured alarm can be written to conform to the proper settings for each state.

5.4 Alarm reduction

Since Tango alarm system can be nested, we propose to introduce at Element/Sub-Element level a layer of such Device Server, in order to implement a censoring/prioritization policies. The exact rules, stored in a database, can be managed remotely, for instance by TM, as needed, also during the operation phase.

5.5 Alarm logging

The presence of a free format alarm description field in Tango alarm system give the possibility to include in the Alarm log a registration of all SKA required informations (see Table 12).

The Tango Alarm system can be easily extended to include a local static log of events/log in a round robin database, such as the RRDTOOLS[vii]. We strongly suggest the collection of such statistical information as a strategy to improve the system reliability also after the instrument initial deployment.

5.6 Logging level

At LMC level there should be the possibility to assign a logging level to log message to filter the verbosity of the logs. SKA defines 8 logging levels while Tango only 6. We propose to add a Parameter (Verbosity) which can differentiate between the last three levels (Debug, Trace and All). The proposed structure can be seen in Table 12.

SKA		Tango	
OFF	No information. Devices should never log messages on OFF logging level.	**OFF:**	Nothing is logged
FATAL	The device has failed. There is no workaround. Recovery is not possible	**FATAL:**	A fatal error occurred. The process is about to abort
ERROR	An error has occurred. A function or operation did not complete successfully. A workaround may be possible. The device can continue, potentially with degraded functionality	**ERROR:**	An (unrecoverable) error occurred but the process is still alive
WARN	A condition was detected which may lead to functional degradation but the device is still fully functional. Logging information at this level should not directly impact the performance of the device	**WARN:**	An error occurred but could be recovered locally
INFO	This level of logging should give information about workflow at a coarse-grained level. Information at this level may be considered useful for tracking process flow.	**INFO:**	Provides information on important actions performed
DEBUG	Verbose output used for detailed analysis and debugging of a device. Logging information at this level may impact performance of the device.	**DEBUG:**	Generates detailed information describing the internal behaviour of a device **Verbosity** = Debug
TRACE	Extremely verbose output for detailed analysis and debugging of a device. This level of logging should show function call stacks and provide a high level of debug		**Verbosity** = Trace
ALL	is the lowest possible logging level and is intended to turn on all logging		**Verbosity** = All

Table 12: Mapping of SKA logging level with Tango logging level and Verbosity variable.

6 Access protection

Tango Control provides a controlled access system (ACL Module). It is a simple controlled access system which does not provide encrypted communication or sophisticated authentication. It simply defines which user (based on computer logging authentication) is allowed to run which command (or write attribute) on which device and from which host. The information used to configure this controlled access feature are stored in the Tango database and it can be accessed by a specific Tango device server.

In our opinion this simple protection schema is sufficient to limit the access to specific areas of SKA to specific operators. Such authorizations can be managed remotely, for instance by TM, as needed. To access remote installation specifically crafted VPN can be devised, maintaining the same protection level of local devices.

7 Conclusions

In this project memo we have outlined a possible implementation of most of SKA requirements in Tango-Control.

We see a lot of potential in having a common set of Tango device driver classes shared with the SKA software, for this having positive implication at TM – Element – LMC level as well as lower level. This is why we would like to start new discussion which aims to the development of a set of uniform tools and approaches in all SKA environment.

Updates on this topic will be carefully noted in future versions of this memo.

Table of Contents

i "*LMC Interface Guidelines Document*", S.R. Chaudhuri et all, Revision R, 2015-03-17, SKA document **SKA-TEL-TM-0000031**
ii "*The TANGO Control System Manual*", The TANGO Team, Revision 8.1, June 27, 2013,
iii "*SKA CSP Local Monitor and Control*" S. Vrcic, **SKA CSP TIM#4**, Cape Town, SA, 15 April, 2015
iv "Proposed SKA Global Nomenclature", S. Vrcic, 2015, in preparation.
v "*SKA CSP Modes and States"*, S. Vrcic, SKA CSP Memo 0015, 2015-05-12
vi "*Development of the TANGO Alarm System*", L.Pivetta, ICALEPCS 2005
vii "*The RRD Tools*" http://oss.oetiker.ch/rrdtool/