



SKA1 CONTROL SYSTEM GUIDELINES (CS_GUIDELINES VOLUME #0)

Document number	000-000000-010
Context	PPP-PPP-PPP-TTT
Revision	01
Author	L. van den Heever
Date	2017-04-10
Document Classification	UNRESTRICTED
Status	Draft

Name	Designation	Affiliation	Signature	
Authored by:				
Lize van den Heever	Technical Lead: MeerKAT Control-and-Monitoring	SKA South Africa		
			Date:	
Co-authors:				
Andrea De Marco	Lecturer	Institute of Space Sciences and Astronomy, University of Malta		
Lorenzo Pivetta	Head, Control Systems	Elettra - Sincrotrone Trieste SCpA		
Simone Riggi	Post-doc	INAF, Osservatorio Astrofisico di Catania		
Sonja Vrcic	CSP LMC Lead	NRC Herzberg Astronomy and Astrophysics, Canada		
Owned by:				
Lorenzo Pivetta	Head, Control Systems	Elettra - Sincrotrone Trieste SCpA		
			Date:	
Approved by:				
Nick Rees	Head of Computing and Software	Square Kilometre Array Organisation		
			Date:	
Released by:				
Alistair McPherson	Deputy Director General & Head of Project	Square Kilometre Array Organisation		
			Date:	

DOCUMENT HISTORY

Revision	Date Of Issue	Engineering Change Number	Comments
A	Oct 2016	-	Draft release for 2016 Engineering Meeting
01	2017-04-10	-	First release for review bu LMC community

DOCUMENT SOFTWARE

	Package	Version	Filename
Google docs	Google Drive	2017	The source of this document is in: www.tinyurl.com/SkaLmcShared / folder LMC Harmonisation/CS_Guidelines Rev01 Google doc named "SKA1 Control System Guidelines - Rev 01"
Word processor	MS Word	Word 2007	Exported from Google docs "000-000000-010 SKA1 Control System Guidelines Rev01.docx"
Block diagrams			The source of the diagrams is in: www.tinyurl.com/SkaLmcShared / folder LMC Harmonisation/CS_Guidelines Rev01 Google slide named "SKA1 Control System Guidelines - Rev 01 - figures"
Other			

ORGANISATION DETAILS

Name	SKA Organisation
Registered Address	Jodrell Bank Observatory Lower Withington Macclesfield Cheshire SK11 9DL United Kingdom Registered in England & Wales Company Number: 07881918
Fax.	+44 (0)161 306 9600
Website	www.skatelescope.org

© Copyright 2016 SKA Organisation.



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

TABLE OF CONTENTS

1. Introduction	5
1.1 Purpose of the document	5
1.2 Scope of the document	6
1.3 Document structure	6
1.4 Relation to LSR, LIG & TIG	6
2. References	7
2.1 Applicable documents	7
2.2 Reference documents	7
3. Element Scope and Responsibilities	8
3.1 Element M&C scope and responsibilities	8
3.2 TM - Element ICDs	9
4. Terminology	10
4.1 Clarifications re monitoring, archiving and logging New	10
4.2 Clarifications re failures, alarms and health state New	10
4.3 Clarifications re SKA states/modes and TANGO states New	12
5. SKA Design Patterns	13
5.1 SKA TANGO Developers Guideline	13
5.2 Device Naming Convention for SKA	13
5.3 Integrating Distributed TANGO Facilities	16
5.4 ElementMaster - Gateway for operational monitoring and control	18
5.5 Element Telescope Model	20
5.6 Local Element Archiving vs Central Archiving	22
5.7 SKA logging	24
5.8 SKA Control Model	26
5.8.1 SCM summary	26
5.8.2 SCM interactions	27
5.8.3 SKA-MID SCM Example	29
SKA-MID discussions:	29
5.8.4 SKA-LOW SCM Example	32
SKA-LOW discussions:	32
5.8.5 SCM attributes	33
5.9 SKA Configuration and Control	37
5.10 Common Alarm handling	41
Appendix A: SKA Control Hierarchies	47
Appendix B: Element LMC Scope and Responsibilities	48
General role	48
Setup and Configuration	48
Control	49
Monitoring	49
Alarms	50
Remote Diagnostic Support	50
Equipment safety	50
Lifecycle management	51

LIST OF ABBREVIATIONS

API	Application Program Interface	PST	PulSar Timing engine
DB	Database	SADT	Synchronisation and Data Transport
DS	Device Server	SAT	Synchronisation and Timing
CBF	Correlator and BeamFormer	SCM	SKA Control Model
CS	Control System	SDD	Self Description Data
CSP	Central Signal Processor	SDP	Science Data Processor
DISH	SKA1-Mid Dishes	SW	Software
DSL	Domain Specific Language	TACO	Telescope and Accelerator Controlled with Objects
EDA	Engineering Data Archive	TANGO	TAcO Next Generation Object system
ELK	Elasticsearch, Logstash, and Kibana	TBC	To Be Confirmed
FQDN	Fully-Qualified Device Name	TBD	To Be Defined
ICD	Interface Control Document	TBW	To Be Written
IEC	International Electrotechnical Commission	TelMod	Telescope Model
IICD	Internal Interface Control Document	TIG	TANGO Interface Guidelines
LFAA	SKA1-Low Frequency Aperture Array	TLS	TANGO Logging Service
LIG	LMC Interface Guidelines	TM	Telescope Manager
LMC	Local Monitoring and Control		
LSR	LMC Scope and Responsibilities		
POGO	Plain Object Generator for tangO		
PSS	PulSar Search engine		

Terminology

Element Facility	The TANGO facility defining all the devices for an Element or each instance of an Element (in case of DSH)
Telescope Facility	SKA-MID or SKA-LOW TANGO facility that integrates all the Element Facilities into a single working instrument. TM is part of the central Telescope Facility and does not have a separate MID-TM or LOW-TM facility.
ElementMaster	The Element Master TANGO device server representing the Element
ElementAlarmHandler	The Element Alarm Handler TANGO device server that implements the Element Alarms as attributes
ElementLogger	The Element Logger TANGO device server that implements the online TANGO logging services as can be viewed by LogViewer
ElementTelModel	The Element Telescope Model TANGO device server that implements the telescope model attributes in the producer Element.

1. Introduction

1.1 Purpose of the document

This "SKA Control System Guidelines" document [**CS_Guidelines** for short] summarises generic design patterns and common approaches for LMC Harmonisation across SKA Elements to maximise the benefit of the TANGO control system framework, as well as what is available from the wider TANGO community, within the SKA project. The goal of the CS_Guidelines is to become the definitive guideline for Element LMCs but unfortunately this document could not be completed before Element LMC design work started and thus will mature with and benefit from the Element LMC work that is happening in parallel. The **CS_Guidelines** is a main parent document for a set of working documents for the areas that the ANT team focused on.

An SKA LMC workgroup (being called the ANT team) has been convened with responsibility to draft the initial version of the CS_Guidelines and drive subsequent versions to mature the SKA design pattern and concepts as required. Refer to the document distributed by Andrea Cremonini after the LMC Harmonisation workshop in Trieste in February 2016 [RD3] (available through eB).

The ANT team will work closely with and under the guidance of the SKA Head of Computing and Software (Nick Rees, as of 1 Mar 2016) in delivering what is required for LMC Harmonisation for the SKA project, focusing on TANGO and monitoring and control design patterns. The responsibility for the CS_Guidelines will be taken over by Lorenzo Pivetta when he joins the SKA project on 18 April 2017.

Although it can be beneficial to standardise on various other areas across Elements of the SKA (like virtualization, computing infrastructure, cluster monitoring, cloud infrastructures, etc) these areas are not within the scope of the ANT team and this document for now. Nick Rees has started a Software Engineering Harmonisation effort to address these aspects, with the issue of the Fundamental SKA Software and Hardware Description Language Standards [AD1].

The ANT team are:

- Andrea de Marco (LFAA) - andrea.demarco@um.edu.mt
- Lorenzo Pivetta (TANGO Consultant/SKA) - lorenzo.pivetta@elettra.eu
- Simone Riggi (DISH) - simone.riggi@gmail.com
- Lize van den Heever (TM) - lvdheever@ska.ac.za
- Sonja Vrcic (CSP) - Sonja.Vrcic@nrc-cnrc.gc.ca

Oversight for the SKAO will be through:

- Andrea Cremonini (SKA System Engineer: DISH) - a.cremonini@skatelescope.org
- Nick Rees (SKA Head of Computing and Software) - n.rees@skatelescope.org
- Juande Santander-Vela (SKA System Engineer: SDP and TM) - j.santander-vela@skatelescope.org

1.2 Scope of the document

The working area for LMC Harmonisation is on google drive at:

www.tinyurl.com/SkaLmcShared in subfolder "LMC Harmonisation" at <https://drive.google.com/drive/u/0/folders/0B8fhAW5QnZQWUmRvSmlCYld4SDA>

The CS_Guidelines document set are:

CS Guidelines volumes:

0. SKA1 Control System Guidelines (main, Volume 0)

0.1 Element TANGO Interface Template (appendix, separate file)

1. SKA1 TANGO Developers Guideline (Volume 1)

2. SKA1 TANGO Naming Convention (Volume 2)

Coordinator

Lize vd Heever

Paul Swart

Lize vd Heever

Lize vd Heever

- First document **CS_Guidelines #0. SKA1 Control System Guidelines** contains a summary of each SKA design pattern. **CS_Guidelines #0.1 Element TANGO Interface Template** is a supporting document to this document but is released as a separate file under the same document number.
- **CS_Guidelines #1 SKA1 TANGO Developers Guide** is the next important document containing various general TANGO design principles for SKA software engineers.
- **CS_Guidelines #2 - SKA1 TANGO Naming convention** will be used to control the SKA TANGO device names.

Technical Notes #3 to #8 contain background information and can be read by developers that want to know more details of options that were considered or motivations for SKA design patterns.

The Technical Notes that informed the CS_Guidelines are:

Topic

Coordinator

LMC Harmonisation - TN 3. SKA Control Model

Sonja Vrcic

LMC Harmonisation - TN 4. SKA Logging

Simone Riggi

LMC Harmonisation - TN 5. SKA Configuration & Control

Sonja Vrcic

LMC Harmonisation - TN 6. Integrating Distributed TANGO Facilities

Lize vd Heever

LMC Harmonisation - TN 7. Element Archiving & Central Archiving

Lorenzo Pivetta

LMC Harmonisation - TN 8. Element & Central Alarms Handling

Andrea de Marco

This work may be followed by more topics like:

LMC HMI requirements and guidelines, and

Combining Simulated and Real devices/facilities

1.3 Document structure

The main document **CS_Guidelines #0** (this one) contains a summary of the outcomes and decisions on each of the topics. Each topic has a working document (Technical Note) with background information and initial analysis. Where there is a mismatch between the CS_Guidelines and a Technical Note, the CS_Guidelines will take precedence.

This document comprise of:

- Section 2 lists applicable and related documents
- Section 3 describes the LMC Scope and Responsibilities (incorporating relevant sections from LSR).
- Section 4 clarifies some terminology.
- Section 5 has a paragraph summarising each SKA Design Pattern. A working document with the detailed analysis for each of these topics are referenced.
- Appendices: The following information have been added to this document as background information. These appendices may be removed in future versions.
 - Appendix A. SKA-MID and SKA-LOW Control Hierarchies
 - Appendix B. Roles from Element LMC Scope and Responsibilities.

1.4 Relation to LSR, LIG & TIG

The CS_Guidelines documentation set incorporates and replaces the LSR (LMC Scope and Responsibilities) [SD1] [SD2], LIG (LMC Interface Guidelines) [SD3] and TIG (SKA TANGO Interface Guidelines) [SD4] documents.

2. References

2.1 Applicable documents

The following documents are applicable to the extent stated herein. In the event of conflict between the contents of the applicable documents and this document, **the applicable documents** shall take precedence.

- [AD1] Fundamental SKA Software and Hardware Description Language Standards, Nick Rees,
SKA-TEL-SKO-0000661, Rev 2, Date 2016-11-24

2.2 Reference documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, **this document** shall take precedence.

- [RD1] TANGO Device Servers - Design & Implementation Guidelines, Rev 6 (available at <http://sourceforge.net/projects/tango-cs/files/doc/TangoDesignGuidelines-Revision-6.pdf/download>)
[RD2] TANGO Control System Handbook v9.2 (available at http://ftp.esrf.fr/pub/cs/tango/tango_92.pdf)
[RD3] "LMC HARMONISATION THROUGH TELESCOPES", Document number 00-000000-003, Document type PLN, Revision 01, Author A.Cremonini, C.Knapic, Date 2016-02-26, (including LMC Use Case responses from TANGO Experts).
[RD4] SKA Observing Control View - produced by N. Rees
<https://docs.google.com/document/d/1g3rH6ZsbyBouHdW2WsCRaiC4lrXRnws0KFI6-Rlb6GM/edit>
[RD5] MID-TT States&Modes RT report
<https://docs.google.com/document/d/1snn9tdnlZL-EiUMDbu5sDpnm3s7mZafv3JqjnZf2iRc/edit>

This document **supersedes** the following documents:

- [SD1] LSR: SKA-TEL-TM-0000030-TM-TELMGT-GDL-Rev01_LMC_Scope_and_Responsibilities
[SD2] LSR02: [SKA-TEL-TM-0000030-TM-TELMGT-GDL-Rev02 Draft LMC Scope and Responsibilities Annotated](#)
- annotated for TANGO
[SD3] LIG: SKA-TEL-TM-0000031-TM-TELMGT-GDL-Rev 01_LMC_Interface_Guidelines
[SD4] TIG: SKA-TEL-TM-0000161-TM-TELMGT-GDL-Rev C_TANGO_Interface_Guidelines

3. Element Scope and Responsibilities

This section captures the relevant information from the original LSR document and the conceptual agreement on the role of an Element for SKA monitoring and control as reached by LmcMadridApr2016. The set of **CS_Guidelines** now supersedes the LSR [SD1&SD2], LIG [SD3] and TIG [SD4] documents.

3.1 Element M&C scope and responsibilities

These paragraphs describe the roles and responsibilities of the Element in the control system of the Telescope at a conceptual level.

In principle:

- **Standalone Element TANGO facility:** Each instance of each Element shall implement a single standalone TANGO facility of which the Element Master will be the entry point that TM, Telescope tools or other Elements will connect to to retrieve some high-level information of the Element for general operations (with exception of DISH, there is a single instance of an Element per Telescope). The rationale here is that TM and consumer Elements need not connect to the Element TANGO DB as they are not interested in the full Element Facility, but need only know where to find the ElementMaster and Element-level devices. To support this "light integration" of multiple TANGO facilities, the ElementMaster shall expose some information about the Element facility to enable general operations, e.g. by exposing the FQDN/alias of the standard Element Level device servers (i.e. Element Logger, Element AlarmHandler, Element TelModel, Element Subarrays), and in some cases other specific information required by TM for general operations (which will be identified by the ICD process, if any). However, when TM or Telescope tools need to drill-down further into the Element facility for more detailed views, fault finding, engineering displays and such, they will connect directly to the Element facility and not expect the Element Master to expose all information of the Element Facility.
- **Primary point of control/coordination for general operations:** The ElementMaster is the primary point of control and coordination of an Element for general operations from TM. For normal operational control TM will communicate with the ElementMaster. The ElementMaster is responsible to distribute and coordinate the control within the Element hierarchy. For example in case of DSH_N the TM will instruct DSH_N ElementMaster to select a specific frequency band (a.k.a a DSH capability) without TM having to control the DSH_N sub-elements (like receivers, positioners) individually, or for CSP the TM will instruct CSP ElementMaster to configure a subarray with a set of correlation beams and pulsar search beams without instructing CBF, PSS or PST sub-elements individually.
NEW However, once the subarray has been activated and configured, a TM Subarray Node may control an Element Subarray directly without going through the ElementMaster and thus "primary point of control" is not strictly true at all times.

"Primary point of control" also does not mean that there is a single point of communication or contact between TM and the Element. Especially for low-level/drill-down monitoring, archiving, logging, fault finding and low-level manual/engineering control, the TM may connect to any lower level device within the Element hierarchy directly as needed (through the standard TANGO framework mechanisms).

- **Rolled-up monitoring and reporting:** The Element Master is the primary point of access by TM/Central tools during general operations for monitoring overall Element status, queries regarding availability and status of Capabilities (e.g. how many (pulsar) SearchBeams are available), management of resources (e.g. assigning resources to sub-arrays). The ElementMaster shall intelligently interpret information from sub-elements and lower level devices for rolled-up reporting of status, health and configuration on Element level, Element Subarrays and Element Capabilities. For general operations, TM shall not be expected to apply rules or knowledge on behalf of the Element to determine state, mode and health of the Element, its Subarrays or its Capabilities (since each Element is in the best position with the best domain knowledge to do so).

For first level operational monitoring TM shall connect only to the Element level TANGO devices (ElementMaster, Element AlarmHandler, Element Logger, Element TelModel, Element Subarrays) and not to the lower level TANGO devices within the Element hierarchy. The information to be exposed on the ElementMaster should thus be everything that is needed for general operations and the ElementMaster should certainly not re-expose the FQDNs, or attributes and commands of lower-level devices "for drill-down". The ElementMaster is expected to roll-up at least overall SCM for the Element, as well as health, modes and states for Capabilities and Subarrays. To support implementation of rolled-up attributes (for high level monitoring) a common rules-based attribute aggregator will be included in a SKA base class.

Operational monitoring: TM will subscribe to the attributes on the ElementMaster for Element, Subarray and Capability information and subscribe to alarms attributes on the ElementAlarmHandler using standard TANGO mechanisms. The Element does not need to implement any additional "reporting" functionality except for exposing relevant aggregated attributes for rolled-up monitoring on the ElementMaster device server and exposing relevant

alarms attributes for Element Alarms on the ElementAlarms device server; and ensure these attributes are appropriately configured for polling and change events and appropriately updated.

Archiving: TM will subscribe, as needed, to attribute archive events on all devices within the Element facility. The Element does not need to implement any additional "reporting" functionality for central archiving, except to configure all attributes with appropriate archiving properties.

Logging: each SKA device by default will have a CentralLogger target and TM can configure the log levels of remote logging for selected devices in the Element through the ElementLogger as needed. The ElementLogger will use the standard TANGO logging commands to distribute this to the selected devices. This is to view on-line logs through the Tango Logging Services. Log storage is described later in a design pattern.

- **Drill-down monitoring:** Elements shall expose detailed information to support fault finding and diagnostics and not hide detailed information inside the Element through defining appropriate TANGO attributes on all devices of the Element hierarchy using the available TANGO mechanisms. Drill-down shall be done by TM, which, when drill-down is needed, shall access the lower level TANGO devices inside the Element hierarchy by connecting directly to the TANGO database of the Element Facility using standard TANGO mechanisms.
- **Rolled-up SKA alarms reporting:** All Elements will adhere to the common SKA design pattern for standardised alarms reporting. SKA and Element alarms are rules-based configurable conditions that can be defined over multiple attribute values and quality factors, and are separate from the "built-in" TANGO attribute alarms that raises the attribute quality factor to `ATTR_ALARM` level based on the min/max warning/alarm ranges defined on each attribute. Each SKA AlarmHandler will expose its alarms as specifically named "alarm" attributes (format and naming to be defined in the SKA design pattern for Alarm Handling). Hierarchies of AlarmHandlers may be implemented in the Element, but the top-level ElementAlarmHandler shall interpret all internal attributes and expose appropriate rolled up Element alarm attributes. TM Central AlarmHandler will in turn define its own rule-based system-wide SKA alarms to be presented to the operators for action. TM Central AlarmHandler will be responsible to subscribe to all Element alarm attributes exposed on the top-level ElementAlarmHandlers for general operations, and thus Elements do not have to push alarms to TM. When drill-down of alarms are required, TM may connect to lower level AlarmHandlers inside the Element hierarchy. Each parent AlarmHandler shall expose the address of its child AlarmHandlers as attributes to allow TM to navigate to the lower level AlarmHandlers for drill-down and fault-finding.

Since the SKA and Element Alarms are exposed as alarms attributes on the AlarmHandler device servers, it shall be archived in the Central Archive with the other monitoring data.

- **Centralised Telescope archiving vs Element archiving:** All Elements will adhere to a common SKA design pattern with regards Element archiving and Central archiving. In principle the Element does not have to implement any special archiving functionality if using one of the available TANGO archivers (like HDB++), provided that the Element's TANGO devices expose all monitoring points as TANGO attributes throughout the device hierarchy to minimize opaque entities and that each attribute are configured for archive events as appropriate.
- **Centralised Telescope logging vs Element logging:** Each Element and each SKA TANGO device down the Element hierarchies will adhere to a common SKA design pattern with regards local logging and central logging. The goal is that TM will be able to selectively configure remote central logging on any device in the Element hierarchy in a common and consistent way for centralised log viewing and archiving, while a robust forwarding channel will be used for log storage.
- **Remote updates:** **TBC** All Elements shall adhere to a TBD SKA design pattern for remote updates of firmware and software as far as possible. The actual upgrades do not have to be done via the TANGO interface, but managing the upgrades has to be supported on the ElementMaster interface. Where this is not possible, the Element ICDs will include a description of the remote upgrade process.

3.2 TM - Element ICDs

LMC Harmonisation - 0.1 Element TANGO Interface Template (generated by Paul Swart)

The LMC Harmonisation effort will result in standardisation that affects the TM - Element ICDs. For this a template for a TANGO based ICD has been developed.

SKA pattern for TANGO based Element ICDs:

- TM - <Element> ICD is expected to only capture the logical Element to TM interfacing for general operational control and rolled-up monitoring of the Element and its capabilities.
- The ICD will not capture the direct interaction through the TANGO framework mechanisms from TM to other devices in the element for drill-down, archiving, logging.
- A standard paragraph describing direct access through the TANGO framework for low-level control, drill-down monitoring, archiving and logging can be drafted and included in each TM - <Element> ICD if desired.
- A reference to adhere to the "SKA Control System Guidelines" [**CS_Guidelines**] should be included in each TM -

<Element> ICD.

- TM Consortium (Paul Swart) has produced a template for TM - Element ICDs (currently a google doc at [TANGO Interface Template](#)) that defines how to capture implementation details of API of TANGO Device Servers.
- The [SKA TANGO Developers Guideline](#) also includes an Appendix with a template for [Device Class Documentation](#) as generated by POGO if all fields are properly populated during device design.

4. Terminology

4.1 Clarifications re monitoring, archiving and logging ^{New}

The **CS_Guidelines** uses the following definitions:

- **monitoring** - is used in the context of a higher-level component subscribing to updates of an attribute with the intention to evaluate its value/quality factor for a specific reason,
- **archiving** - is used in the context of gathering monitoring information (attributes) from a device to save it to a monitoring archive, either the Element Archive or Central Archive, without caring about the value/quality factor of the attribute,
- **logging** - is used in the context of additional information that may be emitted by components to support fault finding and engineering activities. No information that is expected to be used for operations or expected to be monitored by another component may be only in logs: it has to be in attributes that can be monitored. Logs can also be stored and we use the term "log storage" rather than "archiving", the latter which is used to refer to the storing of monitoring information.
- Note - this is standard TANGO terminology and these definitions align with the way TANGO systems use logging, archiving and monitoring.

4.2 Clarifications re failures, alarms and health state ^{New}

Also see par [5.10 Common Alarm handling](#) for further detail.

The **CS_Guidelines** uses the following definitions:

- **failures, errors and faults** - a **failure** occurs when an item is unable to deliver the correct service and is unable to perform its required function according to its specification.. Failures should be reported to LRU level such that the part that is at fault and need to be replaced can be identified. Failures can also be software failures in which case the "part" that is the source of the problem is software component that would need a hotfix for a bug or a restart in some other cases. A **fault** is the cause of an error, or the condition that causes software to fail to perform its required function. **Error** refers to difference between Actual Output and Expected output.
- **health state** - represents the overall health of the item by continuously/regularly evaluating the condition of the item's monitoring points and reported failures and interpreting the impact of those to determine overall health. The health state is reflected as UNKNOWN, OK, DEGRADED or FAILED
- **alarms** - IEC 62682 defines an alarm as "*an audible and/or visible means of indicating to the operator an equipment malfunction, process deviation, or abnormal condition requiring a timely response.*" Note the focus here is that an alarm is always intended for the operator and requires operator intervention in some way. The intervention also has to be timely so that faults that are handled in a daily operations meeting, for example, should not generate alarms.

Definition	Description
An alarm is an audible and/or visible means of indicating	<i>There must be an indication of the alarm. An alarm limit can be configured to generate control actions or log data without it being an alarm.</i>
to the operator	<i>The indication must be targeted to the operator to be an alarm, not to provide information to an engineer, maintenance technician, or manager.</i>
an equipment malfunction, process deviation, or abnormal condition	<i>The alarm must indicate a problem, not a normal process condition. (e.g., pump stopped ,valve closed).</i>
requiring a timely response.	<i>There must be a defined operator response to correct the condition and bring the process back to a desired (safe and/or productive) state. If the operator does not need to respond promptly, then the condition should not be an alarm.</i>

TANGO attribute alarms vs Element Alarms vs SKA Alarms

For the SKA project it is unfortunate that TANGO core called its alarms on attributes "alarms" and not "error" or "fault", as SKA adopted the IEC 62682 standard where the term "alarm" includes "requiring a timely response from the operator". For this reason the SKA project uses the term "**TANGO attribute alarms**" to refer to the TANGO core alarms set on any TANGO attributes and does not have any direct relation to IEC 62682 alarms. TANGO core sets the attribute quality to `ATTR_ALARM` when the attribute value moves into its min/max warning or min/max alarm range, which in turn sets the TANGO device state to `ALARM` when any attribute is in `ATTR_ALARM`.

TANGO attribute alarms are per-attribute indications, while **Element Alarms** is a higher-level concept implemented as rule-based formulae using a combination of attribute values, attribute quality factors (that provide the TANGO attribute alarms), device states. Each **Element Alarm** is exposed as a specifically named and formatted attribute on an Element AlarmHandler device. "**Element Alarms**" follow an attribute naming convention and standardised value format, such that the SKA alarm attributes can be easily identified and grouped, and managed consistently.

The TM Central AlarmHandler is then tasked with the responsibility to implement rule-base formulae and configuration based "**SKA alarms**" which are truly aligned with the IEC 62682 standard for those Element Alarms (or combinations thereof) that need operator response. Some Element Alarms may need automated action, and not necessarily operator response. In those cases the "Element Alarm" is not declared as an "SKA alarm". The TM operator GUI is also responsible to represent the SKA Alarms and Element Alarms appropriately, even on separate displays to prevent confusion of Element Alarms with IEC 62682 alarms.

Definition	Description
TANGO attribute alarm	<i>This refers to the condition where TANGO core sets the attribute quality to <code>ATTR_ALARM</code> because the attribute value moved into its min/max warning or min/max alarm range, also resulting in the TANGO device state to be set to <code>ALARM</code>.</i>
TANGO device alarm	<i>Thes refers to the condition when a TANGO device <code>state</code> is <code>ALARM</code>. When any attribute on a device has a quality factor of <code>ATTR_ALARM</code> TANGO core sets the device state to <code>ALARM</code>.</i>
Element Alarm	<i>Element AlarmHandlers implement "Element Alarms" for all conditions that the TM needs to monitor during operations. Element Alarms will not strictly adhere to the IEC 62682 definition of "alarm" (in that it requires an action by the operator). Element Alarms may be defined even if it is only a notification to TM or the operator, or if it requires automated action by TM and not operator action. Each Element Alarm is exposed as an attribute on the Element AlarmHandler device evaluating a rule-based formula which may be a combination of attribute values, attribute quality factor (that provide the TANGO attribute alarms), device states. The Element AlarmHandlers are delegated with the responsibility to interpret the conditions within the Element and raise Element Alarms for whatever conditions the TM need to know of. The TM Central AlarmHandler is expected to only connect to the Element AlarmHandlers and not to various lower level monitoring points for general operations. The Central AlarmHandler will in turn apply rule-based formula across the Element alerts to implement Telescope level "SKA alarms".</i>
SKA Alarm	<i>A "SKA Alarm" is any Element Alarm(or combination of Element Alarms) that requires operator action (as per the IEC 62682 definition of alarm). This will be configured at the Central AlarmHandler in TM. Central AlarmHandler is responsible to implement the IEC 62682 standard in terms of which Element Alarms are deemed SKA Alarms, present the alarm display for SKA Alarms and manage the operator interaction with SKA Alarms. It will also provide additional displays for Element Alarms and notifications to operators of those, including notifications for any automated actions on Element Alarms. Where an Element Alarm does not escalate to an SKA Alarm, it may be classified as "Suppressed by Design" by TM in the Central AlarmHandler.</i>

4.3 Clarifications re SKA states/modes and TANGO states ^{New}

An attempt is made in the SKA Control Model design pattern to align all the higher-level states and modes work into a set of standard states, modes and flags. See par. [5.8 SKA Control Model](#) and the SKA Control Model working document for further detail.

The SCM attributes comprise:

state (built-in TANGO device state) -	UNKNOWN, OFF, INIT, DISABLE, STANDBY, ON, ALARM ¹ and FAULT
adminMode -	ONLINE, OFFLINE, MAINTENANCE, NOT_FITTED, RESERVE
healthState -	OK, DEGRADED, FAILED, UNKNOWN
obsState -	IDLE, CONFIGURING, READY, SCANNING, PAUSED, ABORTED or FAULT
	supported by attributes configurationDelay and configurationProgress
obsMode -	IDLE, IMG_CONTINUUM, IMG_SPECTRAL_LINE, IMG_ZOOM, PULSAR_SEARCH, TRANSIENT_SEARCH_FAST, TRANSIENT_SEARCH_SLOW, PULSAR_TIMING, VLBI

Additional attributes (if applicable):

controlMode -	REMOTE, LOCAL
simulationMode -	TRUE, FALSE
testMode -	NONE, TEST, custom values

The applicability of the SCM attributes to different levels of the control system hierarchy is further discussed in the SKA Control Model paragraph referenced above.

¹ TANGO device alarm

5. SKA Design Patterns

Now we get into the design patterns proper. This section addresses common TANGO patterns and implementation decisions to coordinate and integrate distributed stand-alone TANGO facilities and a Central TANGO facility into a single instrument, and design patterns that all Element TANGO Facilities should adhere to.

5.1 SKA TANGO Developers Guideline

[Also see **Volume 1: SKA1 TANGO Developers Guideline**]

This SKA1 TANGO Developers Guideline supersedes the TIG [SD4] and highlights important TANGO specific SKA design patterns and SKA best practices. Each SKA Developer has to adhere to the SKA1 TANGO Developers Guideline and should take ownership of it and help to improve it. It will be a continuous work in progress.

It contains design guidelines on various aspects including:

- Element and Device Modelling
- Device Class Documentation
- Tango Configuration Database
- Various Device Implementation Aspects including
 - Device Modes and States
 - Polling and Events
 - Quality Factor
 - Attributes
 - Attribute alarms
 - SKA alarms
 - Attribute archiving
 - Unsolicited information / command progress
 - Configuration and Commands
 - Single Control - Device locking
 - Commands provided by TANGO
- Naming conventions for commands, properties, attributes, enums
- List of standard commands, properties and attributes for Element Level devices and SKA devices

The most important of these are:

- Enforcing accurate configuration of Quality Factor on all attributes
- Enforcing implementation of "device_interface_changed" events for all SKA devices
- Selecting MariaDB as the TANGO configuration database

5.2 Device Naming Convention for SKA

[Also see **Volume 2: SKA1 TANGO Naming Convention**]

This topic addresses the common device naming convention across all TANGO facilities for the full SKA project.

The main SKA design patterns are as follows:

1. **Unicity of TANGO device names:** The TANGO device name triplet (/domain/family/member) should be globally unique (even within the Observatory not just within SKA_MID and SKA_LOW telescopes) without having to depend on facility host and port name.
2. **Centralised management of SKA TANGO devices:** SKA device names are managed by the SKA Device Naming Convention and all devices should adhere to the convention without exceptions.
3. **Granularity of SKA TANGO facilities (Element Facilities):** Every element (or every instance in case of DSH) will be represented by a single separate TANGO facility, we call these Element Facilities.
4. **Granularity of SKA TANGO facilities (Telescope Facilities):** Each telescope (SKA_MID and SKA_LOW) will also be represented by a central TANGO facility, we call these Central Telescope Facilities (SKA-MID and SKA-LOW).
5. **SKA Facility Naming:** SKA Facilities will be referred to with UPPERCASE with **dash "-"** as a separator

e.g. SKA-MID, MID-CSP, MID-SDP, MID-DSH-nnnn, MID-SADT, MID-SAT, SKA-LOW, LOW-CSP, LOW-SDP, LOW-LFAA, LOW-SADT, LOW-SAT

The use of "-" dash in the facility names is acceptable as "_" underscore is deprecated in hostnames.

The TANGO facility may appear in the FQDN, as an example, we can use hostname `ska-mid-tango` for the TANGO host for SKA-MID facility and the default port 10000. Thus the TANGO facility will be identified by the environment variable `TANGO_HOST=ska-mid-tango:10000` (short form without the domain name). Any FQDN belonging to this facility will then read as `tango://ska-mid-tango:10000/<domain>/<family>/<member>`

6. **SKA device naming:** SKA TANGO device name triplets (`/domain/family/member`) will be denoted in lowercase with **underscore "_"** as a separator.

Avoid the "." dot in the name, it appears ugly in the URL of the Tango REST API, and it is easy to overlook.

Avoid "-" dash in the device name (`/domain/family/member`) as it often requires escaping in code.

7. **Controlled number of SKA domains:** There will be a limited set of **SKA domains** allowed per Element; e.g. `low_csp`, `low_csp_cbf`, `low_csp_pss`, `low_csp_pst`

8. The **SKA-MID Facilities** are:

- SKA-MID: Represents the controllable devices composing the whole SKA1-MID
(Note - no separate MID-TM facility)
- MID-CSP: Represents the controllable devices composing the CSP for the SKA1-MID
- MID-SDP: Represents the controllable devices composing the SDP for the SKA1-MID
- MID-SAT: Represents the controllable devices composing the SAT subsystem for the SKA1-MID
- MID-SADT: Represents the controllable devices composing the SADT networks for the SKA1-MID
- MID-INFRA-SA: Represents the controllable devices composing the INFRA for the SKA1-MID
- MID-D0001..Dnnnn: Represents the controllable devices composing SKA1-MID dishes 1 to nnnn
- MID-M000..063: Represents the controllable devices composing the MeerKAT dishes 0 to 63
- MID-MKAT-ANC: Represents TBD products for MeerKAT

9. The **SKA-LOW Facilities** are:

- SKA-LOW: Represents the controllable devices composing the whole SKA1-LOW
(Note - no separate LOW-TM facility)
- LOW-CSP: Represents the controllable devices composing the CSP for the SKA1-LOW
- LOW-SDP: Represents the controllable devices composing the SDP for the SKA1-LOW
- LOW-LFAA: Represents the controllable devices composing the LFAA for the SKA1-LOW
- LOW-SAT: Represents the controllable devices composing the SAT subsystem for the SKA1-LOW
- LOW-SADT: Represents the controllable devices composing the SADT networks for the SKA1-LOW
- LOW-INFRA-AUS: Represents the controllable devices composing the INFRA for the SKA1-LOW
- LOW-ASKAP-WSS: Represents the ASKAP weather station

10. The **SKA-MID Domain Names** are:

- Within CSP:
 - `mid_csp`: represents the master domain for MID-CSP
 - `mid_csp_cbf`: represents the domain of the MID-CSP devices belonging to the CSP CBF
 - `mid_csp_pss`: represents the domain of the MID-CSP devices belonging to the CSP PSS
 - `mid_csp_pst`: represents the domain of the MID-CSP devices belonging to the CSP PST
- Within SDP:
 - `mid_sdp`: represents the master domain for MID-SDP
 - `mid_sdp_lta`: represents the domain of the MID-SDP devices belonging to the SDP Long Term Preservation sub-system
 - `mid_sdp_deliv`: represents the domain of the MID-SDP devices belonging to the SDP Delivery sub-system
 - `mid_sdp_dp`: represents the domain of the MID-SDP devices belonging to the SDP Data Processor sub-system
- Within DSH:
 - `mid_d0000..dnnnn`: represents the domains for devices belonging to each of the SKA1-MID dishes from 1 to nnnn
- Within SAT:
 - `mid_sat`: represents the master domain for MID-SAT
 - `mid_sat_clocks`: represents the domain for all MID-SAT devices belonging to SAT Clocks sub-system
 - `mid_sat_utc`: represents the domain for all MID-SAT devices belonging to SAT UTC scale sub-system
 - `mid_sat_frq`: represents the domain for all MID-SAT devices belonging to SAT Frequency Reference sub-system
- Within SADT:

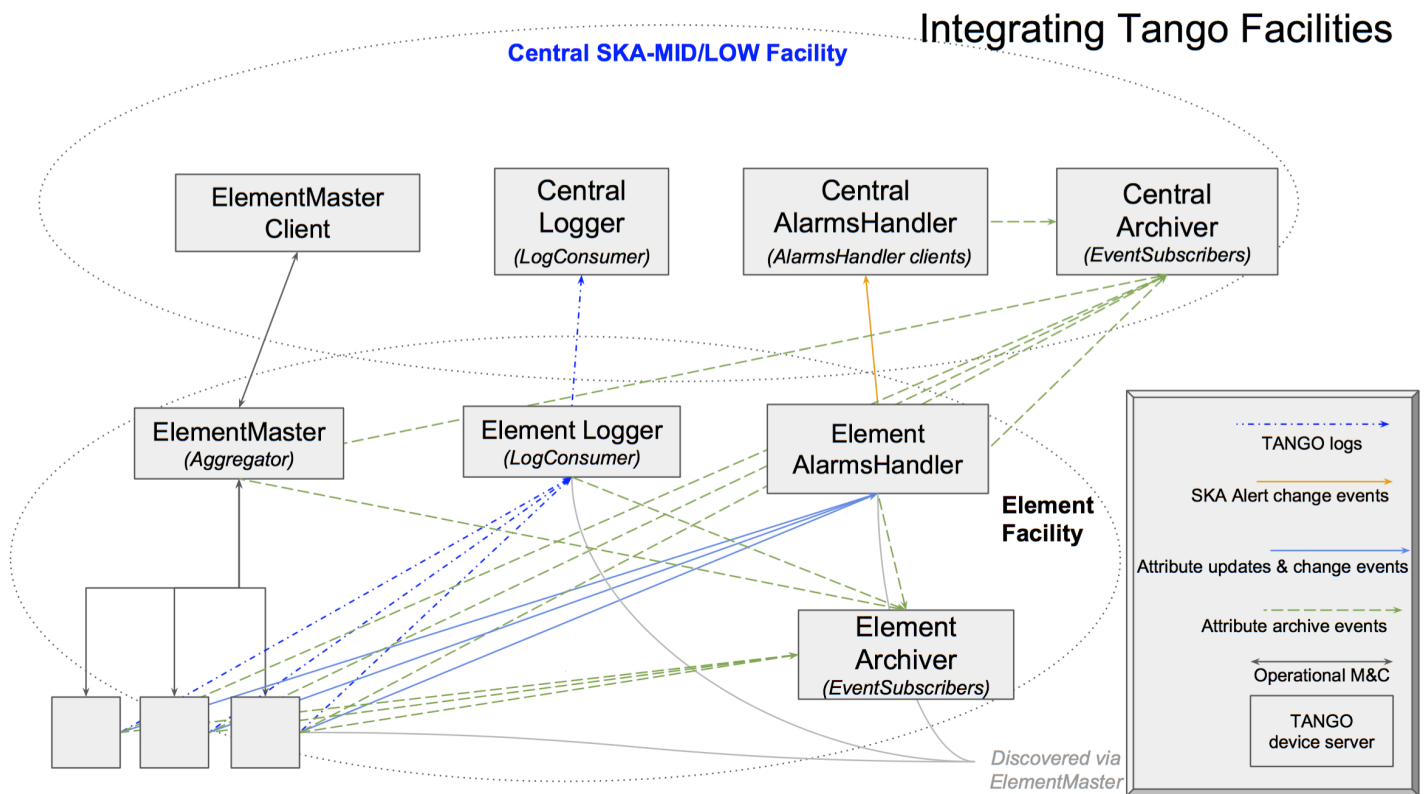
- `mid_sadt`: represents the domain for all MID-SADT controllable devices
- `mid_sadt_nmngnr`: represents the domain for all MID-SADT devices belonging to the SADT Network Manager
- `mid_sadt_ddbh`: represents the domain for all MID-SADT devices belonging to the SADT Digital Data Back-Haul
- `mid_sadt_data` (or `mid_sadt_csp_sdp`) (TBC): represents the domain for all MID-SADT devices belonging to TBCSp/SDP data network
- `mid_sadt_nsdn`: represents the domain for all MID-SADT devices belonging to the SADT NSDN
- Within `INFRA_SA`:
 - `mid_infra_sa` (TBC)
- Within AIV or (`INFRA_SA?`): TBC
 - `mid_m000..m063`: represents the domains for devices belonging to each of the MeerKAT dishes from 0 to 63.
 - `mid_mkat_anc`: represents the domain for devices belonging to MeerKAT ancillaries like weather and wind stations

11. The **SKA-LOW Domain Names** are:

- Within CSP:
 - `low_csp`: represents the master domain for LOW-CSP
 - `low_csp_cbf`: represents the domain of the LOW-CSP devices belonging to the CSP CBF
 - `low_csp_pss`: represents the domain of the LOW-CSP devices belonging to the CSP PSS
 - `low_csp_pst`: represents the domain of the LOW-CSP devices belonging to the CSP PST
- Within SDP:
 - `low_sdp`: represents the domain for all LOW-SDP controllable devices
 - `low_sdp_lta`: represents the domain of the LOW-SDP devices belonging to the SDP Long Term Preservation sub-system
 - `low_sdp_deliv`: represents the domain of the LOW-SDP devices belonging to the SDP Delivery sub-system
 - `low_sdp_dp`: represents the domain of the LOW-SDP devices belonging to the SDP Data Processor sub-system
- Within LFAA:
 - `low_lfaa`: represents the domain for all LOW-LFAA controllable devices
 - `low_lfaa_tbd`: are there a need for further domains within LFAA?
- Within SAT:
 - `low_sat`: represents the master domain for LOW-SAT
 - `low_sat_clocks`: represents the domain for all LOW-SAT devices belonging to SAT Clocks sub-system
 - `low_sat_utc`: represents the domain for all LOW-SAT devices belonging to SAT UTC scale sub-system
 - `low_sat_frq`: represents the domain for all LOW-SAT devices belonging to SAT Frequency Reference sub-system
- Within SADT:
 - `low_sadt`: represents the domain for all LOW-SADT controllable devices
 - `low_sadt_nmngnr`: represents the domain for all LOW-SADT devices belonging to the SADT Network Manager
 - `low_sadt_ddbh`: represents the domain for all LOW-SADT devices belonging to the SADT Digital Data Back-Haul
 - `low_sadt_data` (or `low_sadt_csp_data`) (TBC): represents the domain for all LOW-SADT devices belonging to the CSP/SDP data network
 - `low_sadt_nsdn` (TBC): represents the domain for all LOW-SADT devices belonging to the SADT NSDN
- Within `INFRA_AUS`:
 - `low_infra_aus` (TBC): represents the domain for all LOW-INFRA controllable devices
- Within AIV or (`INFRA_AUS?`): TBC
 - `low_askap_wss`: represents the domain for all LOW-AIV controllable devices

5.3 Integrating Distributed TANGO Facilities

[Also see **Technical Note 3. Integrating Distributed TANGO Facilities**]



This topic addresses design patterns for integrating independent and distributed TANGO facilities of the Elements with a central Telescope facility into a working unit as an instrument. It may touch on the role of the Element Master, single point of control and rolled-up reporting, and other Element standardisation aspects, Local Element logging vs Centralised logging, Local Element archiving vs Centralised archiving, Local Element alarms vs Centralised alarms as it is related, but it will not attempt to define detailed SKA design patterns for any of these aspects.

The main SKA design patterns are as follows:

1. **Integration of facilities through discovery and introspection:** To minimize configuration errors and synchronisation errors, the integration of the different facilities for general operation will be based on discovery and inspection as far as possible, with as little knowledge as possible of the Element facilities to be registered/configured in the central Telescope Facility and consumer Element Facilities. For general operations the TM, Telescope Facility and Central Tools should also not have to connect to and interrogate the Element TANGO DatabaseDS except for drill down. Even the TANGO host:port of the producer Element Facility for the consumer Element can be brokered through TM.
To support this approach the ElementMaster device exposes the Element Level device FQDNs for operational monitoring & control of the Element. This keeps the interface simple and coherent, minimises the dependency of the consumer Element on FQDN naming convention and hides changes in the producer Element. Moreover, pointing to a real vs simulated producer facility in the Central Telescope and consumer Element Facilities is simplified as there is only one FQDN (that of the ElementMaster) to update. It also allows Elements to move Element Level devices or use simulated instead of real devices without affecting the consumer facility and thereby protects the consumer Element from any changes implemented in the producer Element. The ElementMaster provides the shortcuts by populating this information from the Element's TANGO database to guarantee consistency. This design pattern will be implemented in an SKA ElementMaster base class to be available to all Elements.
2. **Drill-down:** For deeper analysis of an Element Facility and drill-down the TM or Central Tools will connect directly to the Element Facility and use the standard TANGO mechanisms to discover and navigate the details of the

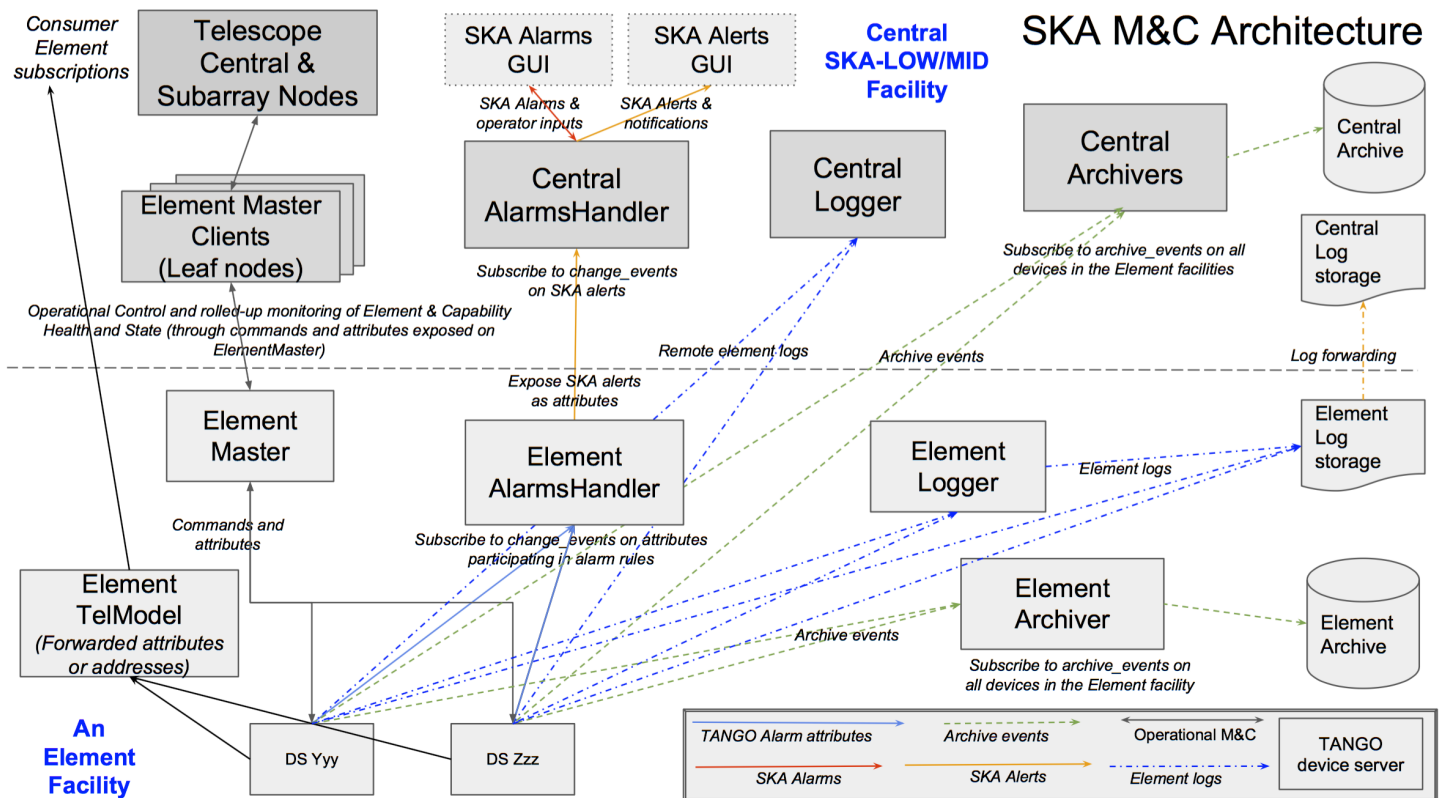
Facility's device hierarchy.

3. **Centralised vs Distributed Logging:** The Central Log Repository stores logs from all devices with the severity level as configured (default INFO and higher). Viewing of online logs is facilitated through the Tango Logging Service. The Central LogViewer provides access to view remote logs of WARNING level and higher from all Elements by connecting to the CentralLogger. For online access to logs with the lower severity levels, the LogViewer tool connects to the relevant ElementLogger. Central Logviewer will normally show the CentralLogger logs (i.e. logs with the severity level of WARNING and up from all Elements) and for drill-down of logging it will connect to the relevant ElementLogger (i.e. logs of INFO level and higher from all devices in a selected Element) as needed.
Rationale: Although some may protest that "INFO level and higher" is quite a high level of logging, it is very useful during integration, commissioning and early stages of operation. As the telescope grows in size we may change this to "WARNING and higher" if necessary.
4. **Centralised vs Distributed Alarm Handling:** CentralAlarms derives the system level SKA alarms using its own central rules based on SKA Alarms defined by the ElementAlarms. CentralAlarms Viewer subscribes to SKA alarms of the CentralAlarms device and for drill-down connects to ElementAlarms for detail.
5. **Centralised vs Distributed TelModel attributes:** ElementTelModel exposes all Telescope Model entities, either as attributes or forwarded attributes. This allows any Element that may need to subscribe to Telescope Model information to identify the appropriate Telescope Model entities. It is not envisaged that TM will provide a single Telescope Model device to expose the complete Telescope Model entities across the whole system, as any consumer Element subscribe directly, peer-to-peer, to the Telescope Model entities of any other producer Element.
6. **Centralised vs Distributed Attribute Archival:** Element Archive is a local archive to store monitoring points for a period of time and is implemented by the Element itself. Central Archive is a full archive of the monitoring points of all the Elements and is implemented by Telescope Manager.

For more detail consult the working document.

5.4 ElementMaster - for operational monitoring and control

[Also see **Technical Note #6. Integrating Distributed TANGO Facilities**]



The ElementMaster represents the Element facility for the central Telescope facility or a consumer Element facility in two contexts:

- 1) The main role of ElementMaster is:
 - a) to be the central control node for the Element and
 - b) to provide rolled-up reporting on the Element, its Subarrays and Capabilities (intelligently rolls-up status reported by all sub-elements and all equipment).

The ElementMaster represents the Element as a unit for monitoring and control during general operations and observations.

- 2) As a secondary role the ElementMaster provides:
 - a) identification of Element Level devices to central Telescope Facility and consumer Element Facilities for general operations
 - b) provides FQDN for subordinate TANGO Devices within the same facility, as needed.

Element level Subarray (apply to LFAA, CSP and SDP):

- is the control point for the Element level Subarray (observing mode configuration and control)
- maintains and reports overall status of the Subarray at the Element Level (intelligently rolls-up status reported by sub-elements, capabilities, equipment used by the subarray).
- may provide FQDN for subordinate TANGO Devices within the same facility, as needed (capabilities assigned and/or used by the subarray, if those are implemented as TANGO Devices, hardware and software components used by the subarray) mostly to support drill-down or detailed interrogation. For example CSP Subarray could provide the list of LRUs used by the subarray.

Translating this into a TANGO-based control system means that:

- Every Element will represent itself to the Telescope Manager as a single unit in an ElementMaster TANGO device. For all intents and purposes, during general operations and observations, monitoring and control will be through the top-level ElementMaster. It is expected that most operations and operational control (once the Element is properly debugged, stabilized and commissioned) will always be done via the ElementMaster, which abstracts all the operations and high-level monitoring for the Element, its Subarrays and its Capabilities. Whilst this may suggest a black-box approach for the Telescope Manager to control every Element, TANGO provides access to every TANGO device in the system. This means that it is entirely possible, for archiving, maintenance, drill-down and auditing purposes, to access every TANGO device within the Element facility.
- For general operations, TM and Central Tools use the information exposed on the ElementMaster device to find Element level devices. ElementMaster exposes attributes providing FQDNs for Element Level devices: ElementLogger, ElementAlarmHandler and ElementTelModel, and Element Subarrays, populated from the Element facility database.

For example, CentralAlarms will find the ElementAlarms device to register for all Element Alarms; CentralLogger will find the ElementLogger device to configure remote logging of devices within the Element facility, any Element may find the ElementTelModel to discover the telescope model entities and subscribe to those it is interested in.

However, CentralArchiver will connect to the Element DatabaseDS directly to discover all the devices in the Element facility to subscribe to their attribute archive events for the central archiving.

Rationale:

- There are various mechanics that can work (either finding the Element Level devices from the ElementMaster or using the producer Element TANGO DatabaseDS). Although introspection is the function of the Tango database server the rationale here is to prevent a facility from "swapping" between its own facility and the producer facility databases continuously as the producer facility database is not "available" at the consumer facility and the consumer facility will in most cases (e.g. in the case of TM) already have a client to the producer ElementMaster. It is not a "natural" TANGO way for a consumer Element to open a TANGO client connection to a remote facility's DatabaseDS but one would not want to change the TANGO host database in the consumer Element for general operations to find the producer Element and therefore the options are a client to the producer ElementMaster or the producer DatabaseDS.

Using the ElementMaster for general operations one would define a property with the FQDN of the producer ElementMaster in the consumer Element TANGO database, and from there read attributes on the producer ElementMaster to find Element Level devices. Otherwise one could define a property with the producer Element DatabaseDS (elt_host:elt_port) instead and then work with a client to the producer Element DatabaseDS exploiting the naming conventions to connect to e.g. `tango://sdp_host:sdp_port/ElementMaster` or `tango://csp_host:csp_port/TelModel`. The use cases we want to support are:

- * to ensure we can easily swap between a real or simulated producer Element facility or real/simulated Element

level device within that facility.

* to allow internal device simulators in the producer Element facility instead of the real device. In the case of introspection through the producer Element DatabaseDS it will mean the producer Element would have to ensure that the simulated device uses the real device name, while in the case where the producer ElementMaster is the gateway it means that the producer ElementMaster would change the value of the FQDN attribute to point to the simulator instead of the real device.

* to ensure that changes in the producer Element does not affect access from the consumer Element. Provided the previous bullet point is implemented, this will be similar in both cases.

Central Telescope Facility will thus have a narrow configuration of each Element Facility in that it will only register the FQDN of each ElementMaster device and open a proxy client to the ElementMaster. Reading attributes exposed on the the ElementMaster identifies the rest of the Element level devices.

- The intention is to have a very narrow configuration of the Element Facilities in the central Telescope Facility to minimize configuration misalignment and time coupling of configuration and in-the-field updates between Elements. It also minimizes the knowledge that Central Telescope Facility and other consumer Element facilities need to have of the Element's implementation and hierarchy of devices as they can then typically instantiate only the appropriate client(s) to selected Element Level device servers.
- For general operations, most TM/central tools will be interested in and have a client to the ElementMaster. Having an ElementMaster expose its Element level devices also allows SKA flexibility if further facility wide standardisation is required bringing additional or different Element level devices. In future it may prove to be useful as the ElementMaster FQDN can point to a simulated Element/device instead of the real Element/device to support ITF and lab integration.

Central point of control and high-level monitoring

The ElementMaster abstracts the operations and high-level monitoring for the Element, its Subarrays and Capabilities. To this effect the ElementMaster will be the primary control node of the Element and the ElementMaster will implement rolled-up reporting for the Element, per Capability, per Subarray, per LRU by intelligently interpreting the state and health of the Element's devices and expose appropriately named aggregated attributes on the ElementMaster device. Some of these could be forwarded attributes from lower level devices, but mostly these will be implemented through rules-based aggregates (e.g. implemented with FormulaConf from ElementMaster base class).

However, as noted before, once a Subarray has been enabled the Subarray control and monitoring can be directly through the Element Subarray device and does not have to flow through the Element Master all the time.

For more detail refer to section [2. Element Scope and Responsibilities](#) earlier in this document defining Element Scope and Responsibilities.

5.5 Element Telescope Model

[Also see Technical Note: **LMC Harmonisation - 6. Integrating Distributed TANGO Facilities**]

TBD by SKAO SW Architects

This paragraph describes one option for a consumer Element to discover and exchange items/parameters that may be for inputs/outputs for the Telescope Model. However, the Telescope Model architecture is still TBD on SKA level and thus this whole section will be reworked once more clarity has been reached.

SKA Design patterns:

1. In this design pattern "*Telescope Model*" refers to those entities (a.k.a attributes on different devices in the Element) that another consumer Element may be interested in for the Telescope Model or to do calculations for the Telescope Model. It does not include the actual algorithms and calculations required for the Telescope Model.
2. Each Element implements an Element TelModel device to enable discovery of telescope model items. The ElementMaster exposes the address of the Element TelModel device as an attribute.
3. The FQDN of the Element TelModel device of the producer Element will be registered in the consumer Element as a free property.

To support combining real and simulated facilities, the TM can broker the producer Element TelModel address. The

consumer Element will then update the property and refresh any connections as appropriate. For example, if CSP needs to consume items from SDP TelModel, then CSP can read the SDP Element TelModel FQDN from TM which can then be the FQDN of either a real or simulated SDP Facility or real or simulated SDP device. The CSP subscribes to the SDP using the information it retrieved, which eliminates the need for the consumer Element to be reconfigured (i.e. updating properties with SDP Facility or device FQDNs in CSP Facility) when the SDP changes or is simulated. In this case only the **TM** host:port / FQDN needs to be registered in each Element Facility. The rationale is that a consumer always requests the information of the producer and subscribes accordingly, instead of "knowing" details of producer Facilities or FQDNs e.g. through properties in its own Facility database or its own devices.

4. The Element TelModel device will expose all telescope model items (either as attributes on the Element TelModel device or as forwarded attributes from lower level devices. In this way, another Element or TM that may be interested in telescope model items can find the items without prior knowledge of the details on where an Element implemented the telescope model in its hierarchy. Naming conventions may be used to group Telescope Model items.
5. Identification of Telescope Model items generated by the producer Element and needed by the consumer Element will be done through the ICD process. Telescope Model items or groups of items have to be identified in the TM - Element ICD. E.g. all attributes named "BeanCount*" will contain the bean counts that was calculated and may be of interest to another Element.
6. The consumer Element will find the ElementTelModel device from the FQDN/alias exposed on the producer ElementMaster, and subscribe to the attributes of interest on the producer ElementTelModel.
7. NOTE: forwarded attributes registration, interrogation, read/write gets "forwarded" directly to the device owning the attribute from the "parent" and should not introduce any bottleneck.
An alternative to exposing each telescope model item on the ElementTelModel device itself would be to expose the FQDN/alias + attribute names of each telescope model items so that a consumer can find it on a different device where it was implemented, but this second options will not be implemented unless we find a problem with performance of forwarded attributes.
8. There may be a case where it is better to deliver some Telescope Model items through an alternative channel and not through pub/sub on TANGO attributes (like TANGO pipes or an alternative protocol like SPEAD). This pattern excludes those cases and only covers TANGO attributes forming part of the Telescope Model.

Rationale:

The Telescope Model device and the use of forwarded attributes allow for data hiding and encapsulation. The consumer does not need to be aware of internal implementation of the Element (and changes of the Element implementation). It prevents the consumer Element directly subscribing to attributes on different devices inside the producer Element.

The process will be that the consumer indicates needs in the ICD; supplier provides attribute name/type/description and other information that should be provided off-line (in order to design consumer accordingly). In some cases consumer may 'discover' attributes - for example if standard attribute names are defined.

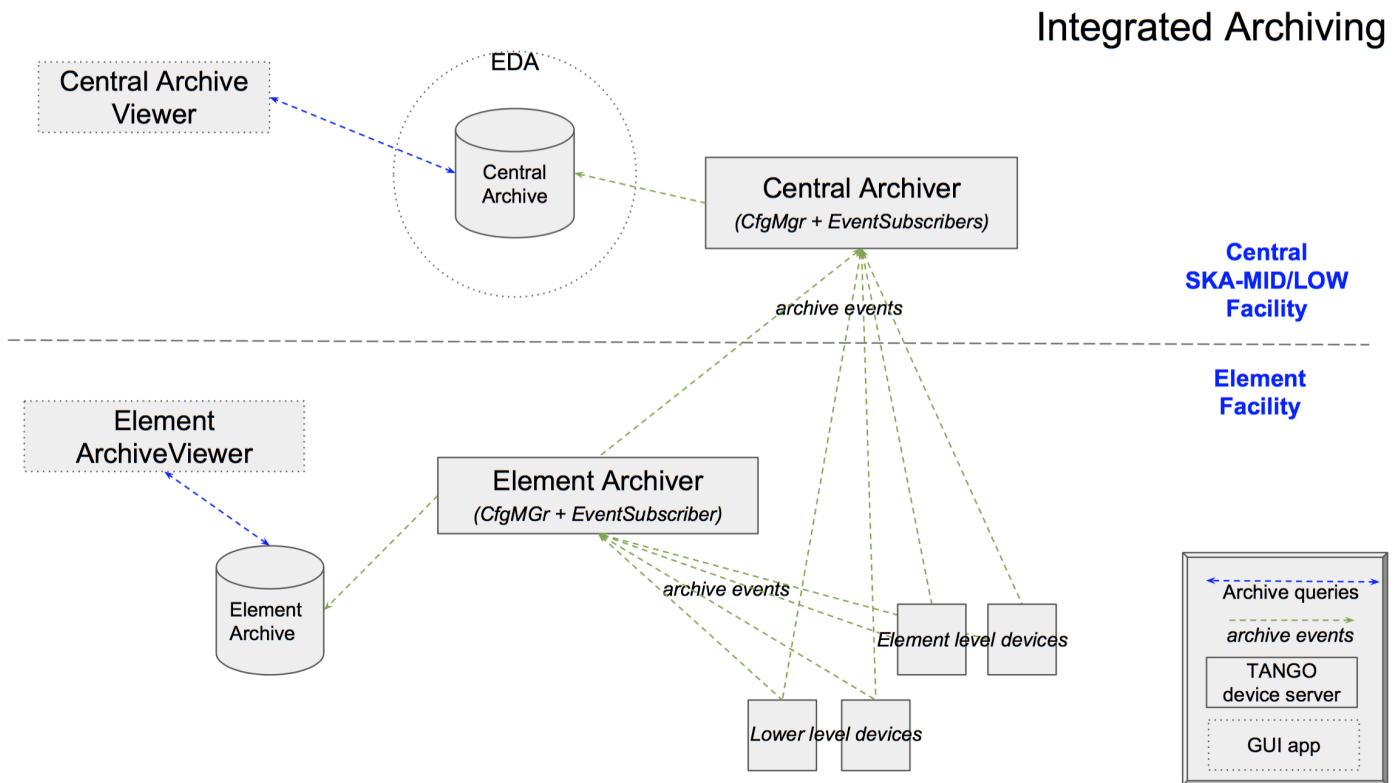
To be resolved: does TelModel include delay tracking, weights for beam forming, Jones corrections used in beam-forming (and other parameters that require periodic updates during observations). If that is the case, TM would be producer/provider and CSP consumer. It is not intended that TM forwards attributes implemented by one Element to a consumer Element, but that the consumer will subscribe directly to the producer Element Telescope Model.

Future aspects that need attention are:

1. Clarifying the role/need for such a TelModel device as described above.
2. Naming convention for Telescope Model items and groups and how the consumer Element will know what to look for on the producer Element. It may be that a consumer Element mostly subscribes to all the items on the producer Element Telescope Model device, but the consumer Element client decides what to subscribe to. We can also define a device attribute naming policy to support selection groups to subscribe to. These aspects will have to be managed through the ICDs (what is produced, what is consumed, how is it named/grouped).
3. Perhaps TM can even provide FQDN of the producer as a part of scan configuration - in some cases producer may be a TM TANGO Device (delay tracking) in others it may be an SDP TANGO Device (updates for Jones Matrices). Attribute names may be agreed in advance in the ICDs. Or TM and SDP may implement 'producers' per subarray, in which case CSP subarray N (where N is in the range [1..16]) would always subscribe for updates on all attributes on the same TM and/or SDP device.

5.6 Local Element Archiving vs Central Archiving

[Also see [Technical Note #7. Element Archiving & Central Archiving](#)]



TM has requirements to archive everything centrally in a Central Archive and queries will mostly be against this Central Archive. The Element Archive is a "backup" as per the element requirements (e.g. DSH that has to store the data for a day or so) and for Element standalone operations. The Element archives can store data for as long or short as they want or as per the Element requirements, can be decimated and rolled-over as decided by the Element with no regards for the Central Archive.

The Element archives are separate from the Central Archive.

Central SKA-MID/LOW facility is in charge of archiving SKA-MID/LOW devices as well as all devices in Element facilities in the SKA-MID/SKA-LOW. Uniformity and operational reasons suggest to maintain a local archive also for the SKA-MID/LOW facility itself, allowing to store any data which may be required for local operation but not advisable to be stored in the central archive. It should be noted that, with the possible exception of MID-DSH and SDP facilities, all element archives may share a common infrastructure in the CPF, so the local archives may not be physically separate. **TBD by SKAO SW Architects**

For local Element archiving the architecture comprises:

- the Element Archiving setup with a MariaDB backend,
- an HDB++ ConfigurationManager device server and
- a number of HDB++ EventSubscriber device servers.

For Central Archiving a Cassandra backend is foreseen with at least:

- one dedicated HDB++ ConfigurationManager device server. More than one ConfigurationManager device server can be deployed whenever complexity, architectural or practical reasons require it.
- at least one **dedicated** EventSubscriber device server is foreseen for each Element for central archiving. Additional EventSubscribers may be deployed for central archiving of each Element whenever performance, architectural or operational reasons require it. It will also be possible to launch an optional Subarray EventSubscriber when required, but currently there is no real need for this use case identified yet.

Central Archiving will be implemented to automatically find all devices in the Element facility and subscribe to archive

events on each of the devices in the Element facility. This eliminates the need to register each and every attribute on each device in the central archivers. This can be implemented by runtime commands on EventSubscribers from some external device server, or as a new kind of "per-device" EventSubscriber. It may even be extended to a "per-facility" EventSubscriber in which only the Element facility to archive is registered, it then connects to the Element Facility to find all the devices, and then registers for archive events on all the attributes for all the devices. It is possible to customise the archiving by using the "**SkaLevel**" property suggested elsewhere to determine which devices to include in Central Archiving and Element Archiving. Note: archiving configuration is set by the device itself and each device determines which attributes should be archived and the archive period for each.

A couple of graphical user interfaces are available for the HDB++ archiving system. One, using the C++ extraction library, is based on Qt and exploits the MathGL framework for plotting. It is capable of multiline plots as well as surface plots. Refer to HdbExtractor++ GUI documentation and HdbViewer GUI documentation for additional details [referenced from SKA Archiving guideline]. A web-based tool named eGiga, capable of retrieving data from the MySQL based backend, is also available.

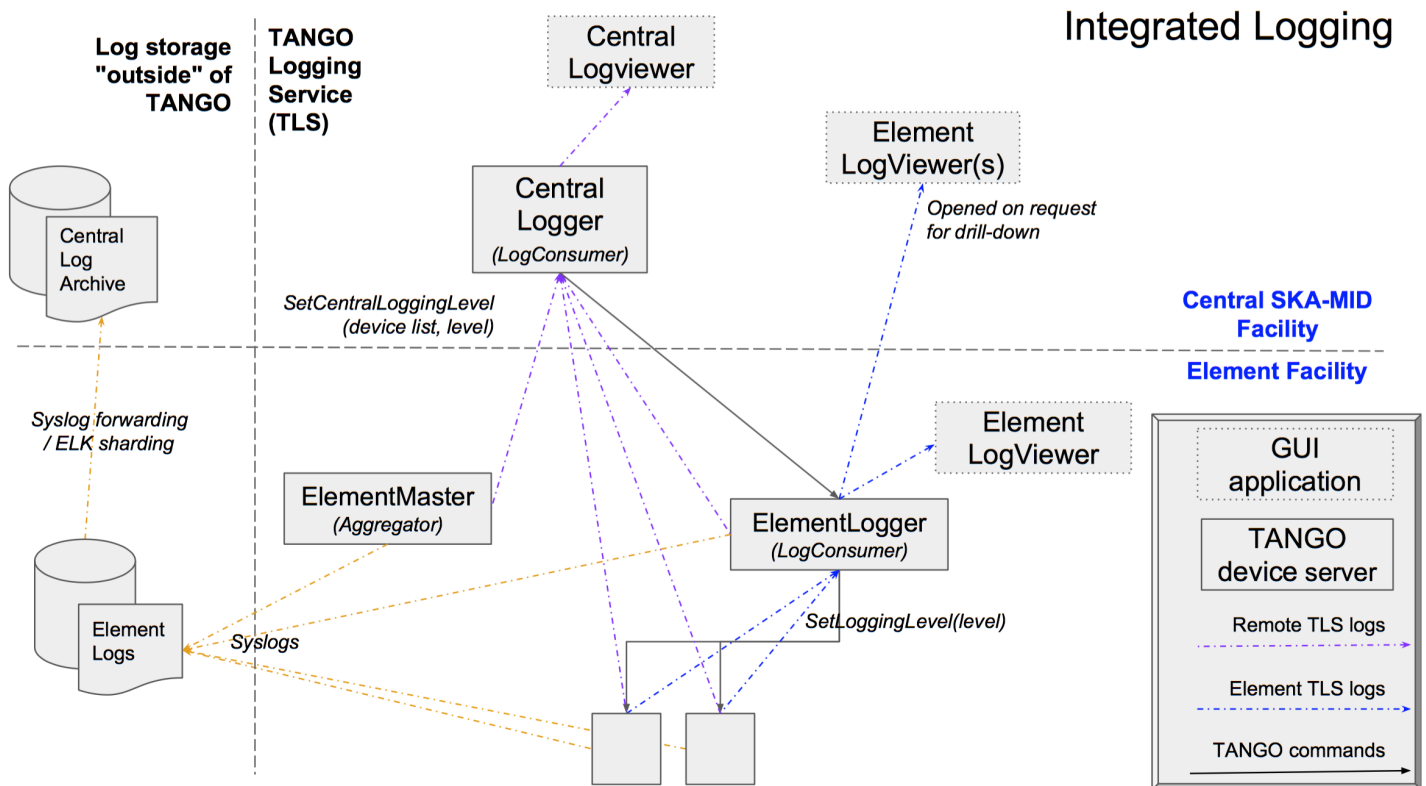
For a detailed description of the HDB++ archiving and more detail on how that will be leveraged for SKA Archiving consult the working document.

For more detail consult the working document.

NOTE: We discussed whether a mechanism is required to pull data from the Element Archives to fill gaps that may have appeared on the Central Archive due to error conditions, or such. And if this would be done via TANGO commands in some way or directly between the Element and Central Archive databases. Prefer the latter if required. This may be a future addition and is not required for initial releases.

5.7 SKA logging

[Also see **Technical Note #4. SKA Logging**]



The main SKA design patterns are as follows:

1. **Log storage vs attribute archiving** - The term "Log storage" is used to denote long persistence of log information in the SKA. It should not be confused with the term "archiving" which is used to refer to storing of time-series monitoring data (e.g. attribute values). Archiving of monitoring data and storage of logs will follow different patterns and employ different technologies.
2. **Logging supporting services:** SKA Logging will be based on two services:
 - a) Tango Logging Service (TLS) to support viewing of logs in an Element and centrally for logs across Elements
 - b) rsyslog with ELK stack to store logs
3. **TLS scope:** The Tango Logging Service is mainly to support visualisation of selected logs (typically WARNING and above) across all Elements through CentralLogger and a specific Element's logs (typically INFO and above) through ElementLogger,
4. **LogConsumer interface for log ingestion in ElementLogger:** Each Element will implement an ElementLogger device implementing the LogConsumer interface to support LogViewer for Element wide logs.
5. **LogConsumer interface for log ingestion in CentralLogger:** TM will implement a CentralLogger device implementing the LogConsumer interface to support LogViewer for WARNING and higher level logs across all Elements.
6. **Log storage main technology choice:** Syslog is used to support implementing the actual log storage. Syslogs will be forwarded for Central Log Storage (in an ELK stack).
7. **Log storage technology choice for small logs:** Smaller elements, like DSH, will store logs (probably to files) with syslog which can be forwarded to the central log store (Elastic) for long term storage.
8. **Log storage technology options for large logs:** Larger elements like CSP, SDP, TM and LFAA will also use syslog but may configure log sharding and use the ELK stack to share a log storage solution.
9. **Use cases for ELK vs TLS logs:** The central ELK log store will be used for fault finding and forensics while the TLS will mainly be used for "in-time" monitoring of log messages.
10. **ElementLogger** - An Element level LogConsumer device to support viewing the Element logs and manage remote

logging (to TM) of devices in the hierarchy - as described in the SKA Logging Guidelines.

11. **CentralLogger** - A Central LogConsumer device at TM that collates logs from all Element facilities in the telescope to view selected centralised logs across Elements. Will connect to ElementLoggers to configure the remote logging of the devices in that Element. CentralLogger can set the log levels per log target for a selected device, via ElementLogger. ElementLogger will distribute the request to the selected device using standard TANGO logging commands.
12. **Logging targets for TANGO Device Servers:** Each SKA TANGO device will log to 3 pre-configured logging targets:
 - a. ElementLogger log target at default INFO level
 - b. CentralLogger log target at default WARNING level for all Elements
 - c. Storage/syslog local server at default INFO or DEBUG level
13. **Remote logging configuration:** TM CentralLogger will configure remote logging of devices in an Element via commands to the ElementLogger:
 - a. `SetCentralLoggingLevel` - typically used by TM CentralLogger to change logging level to CentralLogger on a selected set of devices within the Element
 - b. `SetElementLoggingLevel` - typically used by ElementLogger to change logging level to ElementLogger on a selected set of devices within the Element
 - c. `SetStorageLoggingLevel` - typically used by CentralLogger or ElementLogger to change logging level to syslog storage on a selected set of devices within the Element
14. **Logging level default values as properties:** Each ElementLogger will expose **properties** for the default logging level of each of the three targets:
 - a. `CentralLoggingLevelDefault`
 - b. `ElementLoggingLevelDefault`
 - c. `StorageLoggingLevelDefault`
15. **Current logging level as attributes:** Each SKA Device expose **attributes** for the current logging level of each of the three targets:
 - a. `centralLoggingLevel`
 - b. `elementLoggingLevel`
 - c. `storageLoggingLevel`
16. **SKA logging format:** SKA will use the default logging format defined in the Fundamental SKA Standards (currently RFC 5424). The full format of a syslog message sent on the wire has three discernable parts with total length 1024 bytes or less:
 - a. **PRI:** The Priority part is a 8-bit number enclosed in angle brackets <> and representing both the message Severity (i.e the log level with the first 3 least significant bits, thus up to 8 different severities) and the Facility (i.e. the source application generating the log (the remaining 5 bits). There are predefined codes for facility and severity level. The log priority level can then be computed from severity and facility in a straightforward way.
 - b. **HEADER:** The HEADER part contains two fields:
 - i. `TIMESTAMP` (date & time) at which the message was generated, generally taken from the system time
 - ii. `HOSTNAME` (or ip address) of the server generating the message
 - c. **MSG:** The MSG part has two fields:
 - i. `TAG`, i.e. the name of the application/process that generated the message with a length not exceeding 32 characters
 - ii. `CONTENT` field, i.e. the detail of the message
17. **Logging requirements for the SKA TANGO base class:** To support this design pattern, the following has to be included in TANGO controls or base classes:
 - a. Setting the log level per log target
 - b. Connecting LogViewer to a LogConsumer in a different TANGO facility (i.e. using FQDN of the LogConsumer). LogViewer will have to be enhanced to allow selecting a LogConsumer from a different facility to view logs.
 - c. ElementLogger provide commands for TM to set the log level on any of the three targets for a selection of devices or all devices in the Element hierarchy as described in Table 4.2.1.
 - d. ElementLogger device provides logging attributes `loggingLevel`, `ElementLoggingLevel`, `CentralLoggingLevel` as described in Table 4.2.1
 - e. Each SKA device provides logging attributes `storageLoggingLevel`, `elementLoggingLevel`, `centralLoggingLevel` as described in Table 4.2.2
 - f. Top-level ElementMaster device exposes ElementLogger fully qualified device name as an attribute

For more detail and motivations consult the working document. Where the working document differs from this summary, this document takes precedence.

5.8 SKA Control Model

[Also see **Technical Note #3. SKA Control Model**]

NEW Various forums have done work on SKA modes and states on different levels and each with a slightly different focus. These include:

- SKA Control Model work of by ANT team #4, which were preceded by the original SCM work of the TM TelMgt team in the LIG and TIG (with lots of detail and thinking)
<https://docs.google.com/document/d/1422IecZUcjBelwITCGqc3Byop9FzD5UHD36CD-hy6qw/edit#>
- SKA Observation Control View [RD4]
<https://docs.google.com/document/d/1g3rH6ZsbyBouHdW2WsCRaiC4lrXRnws0KFI6-Rlb6GM/edit>
- MID-TT States&Modes RT report [RD5]
<https://docs.google.com/document/d/1snn9tdnlZL-EiUMDbu5sDpnm3s7mZafv3JqjnZf2iRc/edit>
- TANGO core [RD2]
http://ftp.esrf.fr/pub/cs/tango/tango_92.pdf

There is also an SKA MID TT Resolution team on Subarrays that may affect the SCM.

An attempt is made here in the summary of the SKA Control Model design pattern below to align all the work on SKA states and modes into a set of standard TANGO attributes and commands. These will have to be reworked if other forums update their SKA state and mode thinking.

5.8.1 SCM summary

[Note that the description of the states and modes and a description of the valid values are presented in par [5.8.5 SCM attributes](#).]

SKA design patterns for states and modes:

1. The **SCM attributes** are listed below, with the superscript **r/o** indicating a read/only attribute, and **r/w** a writable attribute. An asterisk (*) indicates a memorized attribute (see Section 7.7 in R6). Bear in mind that not all of the SCM attributes are mandatory, and some may be applicable only to specific levels.
 - a. `stater/o`: (built-in TANGO state) UNKNOWN, OFF, INIT, DISABLE, STANDBY (low-power), ON, ALARM and FAULT
 - b. `obsStater/o`: IDLE, CONFIGURING, READY, SCANNING, PAUSED, ABORTED, or FAULT
 - c. `*adminModer/w`: ONLINE, OFFLINE, MAINTENANCE, NOT_FITTED, RESERVED
Factory default is MAINTENANCE
 - d. `healthStater/o`: OK, DEGRADED, FAILED, UNKNOWN
(for Element, Capabilities, Subarrays, Components, LRUs)
 - e. `obsModer/o`: IDLE, IMG_CONTINUUM, IMG_SPECTRAL_LINE, IMG_ZOOM, PULSAR_SEARCH, TRANSIENT_SEARCH_FAST, TRANSIENT_SEARCH_SLOW, PULSAR_TIMING, VLBI
 - f. Additional attributes:
 - i. `*controlModer/w`: REMOTE, LOCAL
 - ii. `*simulationModer/w`: TRUE, FALSE
 - iii. `*testModer/w`: NONE, custom values (optional)
(other values optional and custom defined, e.g. test1, test2, test-xy)
 - g. Supporting attributes: each entity that implements `obsState` shall also provide
 - i. `configurationProgressr/o` - percentage progress
 - ii. `configurationDelayExpectedr/o` - the time it will take to prepare the requested configuration, i.e. the expected time to transition from `obsState` in CONFIGURING to READY. Rationale: although this may not strictly require an attribute and could be returned as the output parameter on the configure commands, over time it might be really useful statistics to see which modes took how long to configure and how it changes with the growth of the system, etc. But that said, it will be necessary to have two attributes then: `configurationDelayExpectedr/o` and `configurationDelayMeasuredr/o`.
2. The definition of each of these modes and states and the relation between them are defined below in section [5.8.5 SCM attributes](#).
3. **R/O attributes with a related set of commands** - `state` and `obsState`:

The `SCM state` and `obsState` attributes shall be implemented as **R/O attributes** and a **related set of commands** shall be implemented to change these. (Rationale: This approach is aligned with implementation of the TANGO device state. A set of single-action commands keeps the device interface clean whenever a not trivial state machine with complex and/or time consuming transitions has to be implemented).

The **commands** are:

- a. `state` - the TANGO device state can be changed by the built-in state commands (Off, On, Disable, etc). Raise exception for states not used by the device (not every device implements all the states).
- b. `obsState` - Observing State may be commanded through a different to-be-defined mechanism e.g. as per the description in SKA Observing Control View [RD4] which has the following commands: Configure, Reconfigure, Scan, EndObservation, Pause, Continue, EndScan, Abort and Reset). Raise an exception if requested mode is not supported. Shall be implemented per Subarray and/or Capability (as applicable, **TBC**).
4. **R/W attributes** - `adminMode`, `controlMode`, `simulationMode` and `testMode`:
The SCM attributes `adminMode`, `controlMode`, `simulationMode` and `testMode` shall be implemented as **R/W attributes** of type enumeration. Mode changes are commanded using the TANGO standard command `write_attribute()`. A device that does not support specified value or does not support requested mode transition shall raise an exception (e.g. the device is a simulator and cannot be instructed to exit simulation mode). Whenever possible, devices shall provide the same (or at least similar) interface across the board, i.e. implement the full set of attributes and use standard enumerations, and raise an exception when a request to perform unsupported transition is received (TBC).
5. **R/O attributes that cannot be changed through direct commands** - `healthState` and `obsMode`:
The following SCM attributes cannot be changed directly from an external source by command or writing to an attribute:
 - `healthState` - determined by the device
 - `obsMode` - set as a part of the configuration of the subarrays and capabilities and not directly, using a separate command.

Applicability of SCM states and modes:

The states and modes will apply differently at different levels of the control hierarchy. For example:

1. The only mandatory SCM attribute for **all** devices on all levels of the control system is the built-in TANGO `state`
2. `healthState` is expected to be implemented by many devices on different levels of the control system (may even be implemented on all devices) but will at least be mandatory on Element, Sub-Element, Subarray and Capability levels; as well as on LRUs and Applications (to identify what hardware to replace or what software to restart).
3. `obsState` and `obsMode` are typically applicable to Subarray and Capability Device Servers
4. `adminMode` is typically applicable to a device that is representing a resource that can be allocated to a subarray (e.g. a Capability) or a master controller of an Element or SubElement, or even an LRU
5. `controlMode` will typically be used for DSH, but may also be applicable to other devices like CSP, or CSP sub-elements.
6. `simulationMode` and `testMode` can be applicable to any level of device that implements these functionalities

5.8.2 SCM interactions

[Note that the description of the states and modes and a description of the valid values are presented in par [5.8.5 SCM attributes](#).]

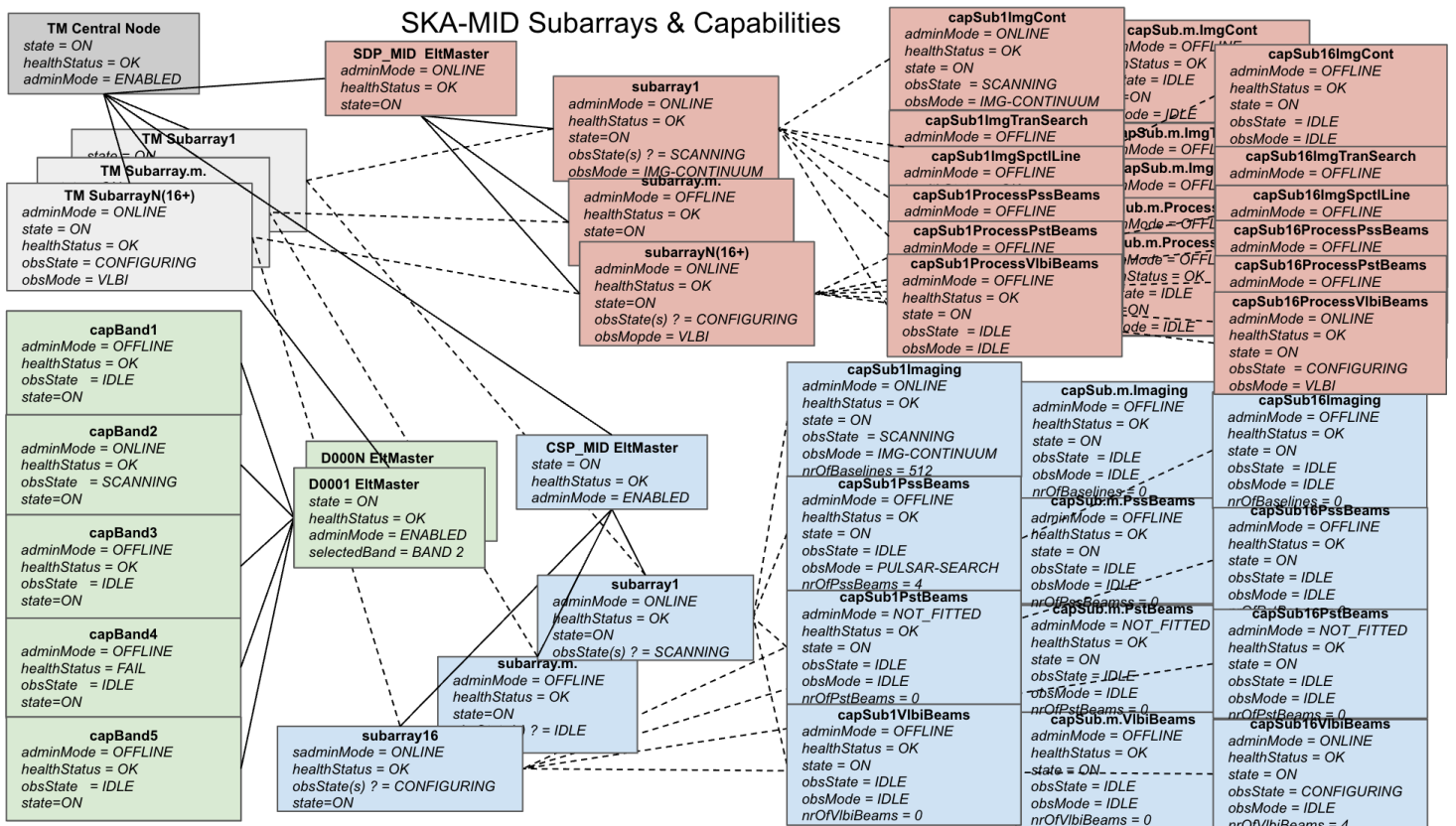
When Element `state` is marked as STANDBY/DISABLE should software sub-components (e.g. subarrays and capabilities) also reflect STANDBY/DISABLE or only entities where these are actually implemented e.g. for DISH band capabilities, or for Capabilities on a STANDBY/DISABLE Subarray.

1. `healthState`:
 - a. shall be interpreted based on children and own device state, `adminMode`, `healthState`
 - b. Element `healthState` shall take `healthState` of the subarrays and capabilities invoked on that Element into account
 - c. Subarray `healthState` shall take `healthState` of the capabilities invoked on that subarray into account
2. `adminMode`:
 - a. when `adminMode` is NOT_FITTED, TANGO state shall be DISABLE
 - b. when `adminMode` is RESERVE, TANGO state shall be DISABLE

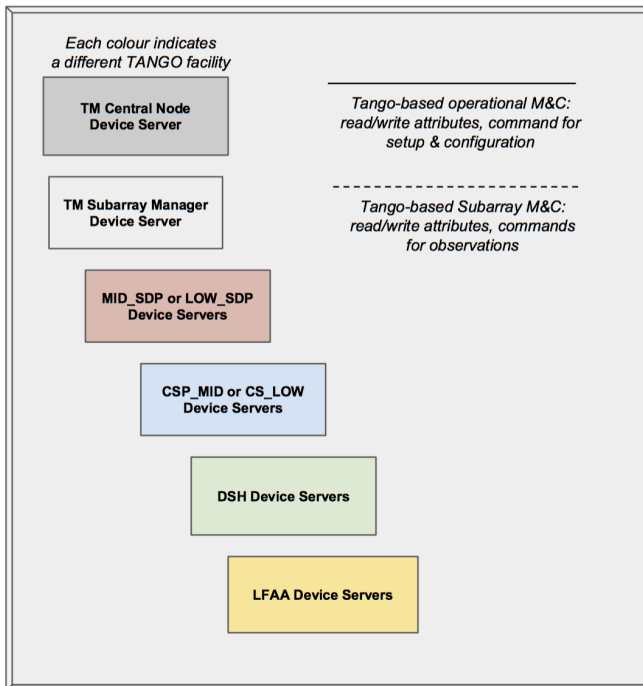
Rationale: when adminMode is NOT_FITTED, OFFLINE or RESERVE the TANGO device itself (internally) sets the TANGO State to DISABLE, which means that the attribute warn/alarm ranges are not evaluated and, as a consequence, device does not raise attribute alarms. The TM/Central tools are programmed to ignore/exclude all devices with state=DISABLE where appropriate.

3. TANGO State:
 - a. SKA has specified that equipment shall power-up in STANDBY state. The transition to STANDBY can, for example, be implemented by setting the TANGO device state to STANDBY at the end of the init_device() method by calling set_state(STANDBY) as last step before exiting init_device().
 - b. TANGO state may be DISABLE for other reasons than those listed under adminMode, for example if a Subarray has not been "created" or a Capability has not been selected/invoked.

5.8.3 SKA-MID SCM Example



Legend for SKA-Mid and SKA-LOW Subarrays & Capabilities



SKA-MID discussions:

ASSUMPTIONS:

- Each Subarray can only execute one primary SB (thus one primary obsState & obsMode)
- The subarray does not "exist" post SDP ingest and can be freed once the SDP ingest delivers the captured science data product on disk. The subarray does not have to "exist" for SDP post-processing.

GENERAL OBSERVATIONS:

1. The decision has been made that all Subarray devices as well as a device for each allowed category of Capability is instantiated for each subarray at startup (see the following 3 points for further explanation) and not dynamically.
The rationale for always instantiating 16 Subarray devices and not only instantiate a subarray devices when the subarray is activated is to make the control hierarchy less dynamic when not needed, resulting in a more consistent control hierarchy for the higher level monitoring and control with less exception processing when Subarrays and Capabilities devices do not exist.
2. This is a very simple, but deterministic approach to Subarrays and Capabilities. One could optimise and generalise it, but this simple approach is sufficient. Having a "fixed" value N (16 or more, but a configured, known number of subarray devices, not a dynamic number; and for Capabilities NxM, where M is the number of different Capabilities that can be invoked on each subarray) is not necessarily that elegant, but it does minimize edge cases that have to be tested, reduces complexities on various levels, including GUI representations, rules for inclusion/exclusion of alarm and aggregation rules, etc. The number 16 was defined by the System Engineers and deemed sufficient for SKA operations based on the operations concept of the telescope for observing, commissioning, AIV, and maintenance. 16 can easily be increased to 20 or such (except for the CSP) if more non-CSP subarrays are required. The main aim is to not have a potentially dynamic number, but a known, configured number, in the order of 16.
3. All N subarray devices are instantiated at startup in each element supporting subarrays: for example, take the TANGO Device class CspSubarray - during initialization MID-CSP instantiates 16 objects of that class. These startup as inactive (`adminMode=OFFLINE`) and are activated on command from TM as required (`adminMode=ONLINE`) and deactivated (`adminMode=OFFLINE`) on tearing down the subarray.
The rationale for using `adminMode=OFFLINE` to indicate an inactive subarray as opposed to just evaluating `obsState=IDLE`, is to distinguish between a subarray that has been "enabled" (`adminMode=ONLINE`) but is not yet observing (`obsState=IDLE`) from a subarray that is disabled (inactive/free) (`adminMode=OFFLINE`).
4. **New** TANGO Devices representing allowable Capabilities per subarray are also initiated at startup. Initially they all belong to a pool of unused resources (`adminMode=OFFLINE`). The Capability devices that will be instantiated per Subarray are:

CSP_MID Subarray -	capImaging, capPssBeams, capPstBeams, capVlbiBeams
SDP_MID Subarray -	capImaging, capImagingTranSearch, capImagingSptLine
	capProcessPssBeams, capProcessPstBeams
Dish -	capBand1, capBand2, capBand3, capBand4, capBand5
CSP_LOW Subarray -	capImaging, capPssBeams, capPstBeams
SDP_LOW Subarray -	capImaging, capImagingTranSearch, capImagingSptLine
	capProcessPssBeams, capProcessPstBeams, capProcessVlbiBeams
LFAA Subarray -	capLstnBeams

For example: the current CSP design plan is to instantiate each PSS/PST/VLBI beam as an individual TANGO Device, representing each function as an instance of a TANGO device. However, TM will only monitor the subarray1..16 and capSubNImaging, capSubNPssBeams, capSubNPstBeams, capSubNVlbiBeams devices for MID-CSP for normal operations and only drill-down to individual PSS/PST/VLBI beam devices for fault finding.
5. Subarray is enabled (activated) by setting `adminMode=ONLINE`, it can then be configured, monitored and controlled, and finally disabled (deactivated/freed) with `adminMode=OFFLINE`
6. Once the Subarray is `adminMode=ONLINE`, the Capabilities TANGO devices are invoked by setting `adminMode=ONLINE`, and can then be configured, monitored and controlled, and finally revoked with `adminMode=OFFLINE`
7. The relevant Capability devices are then invoked (`adminMode=ONLINE`) and configured with additional parameters (such as the number of PSS beams, or the number and identification of the baselines) as TM configures the capabilities on the subarray. E.g. TM will instruct CSP to activate subarrayM and assign a certain set of antennas/dishes. It will then configure subarrayM capabilities as required (number of baselines as per the antennas, number of PSS beams, number of PST beams, etc). As these configurations are done, the CPS will activate the capabilities on subarrayM by setting capSubMBaselines device `adminMode=ONLINE` or capSubMPssBeam `adminMode=ONLINE` and setting the appropriate parameters on each of the four Capability devices as appropriate.
8. The negative of using the TANGO `adminMode=OFFLINE` on subarrays and capabilities that are inactive is that TANGO only evaluates attribute alarms in TANGO `adminMode=ONLINE`.

DSH:

1. DSH Master - DSH does not know about subarrays and will only instantiated 5x Capability devices (one for each band) at startup. DSH sets selected band capability to `adminMode=ONLINE` when TM requests a frequency band and `adminMode=OFFLINE` for other band capabilities.

CSP:

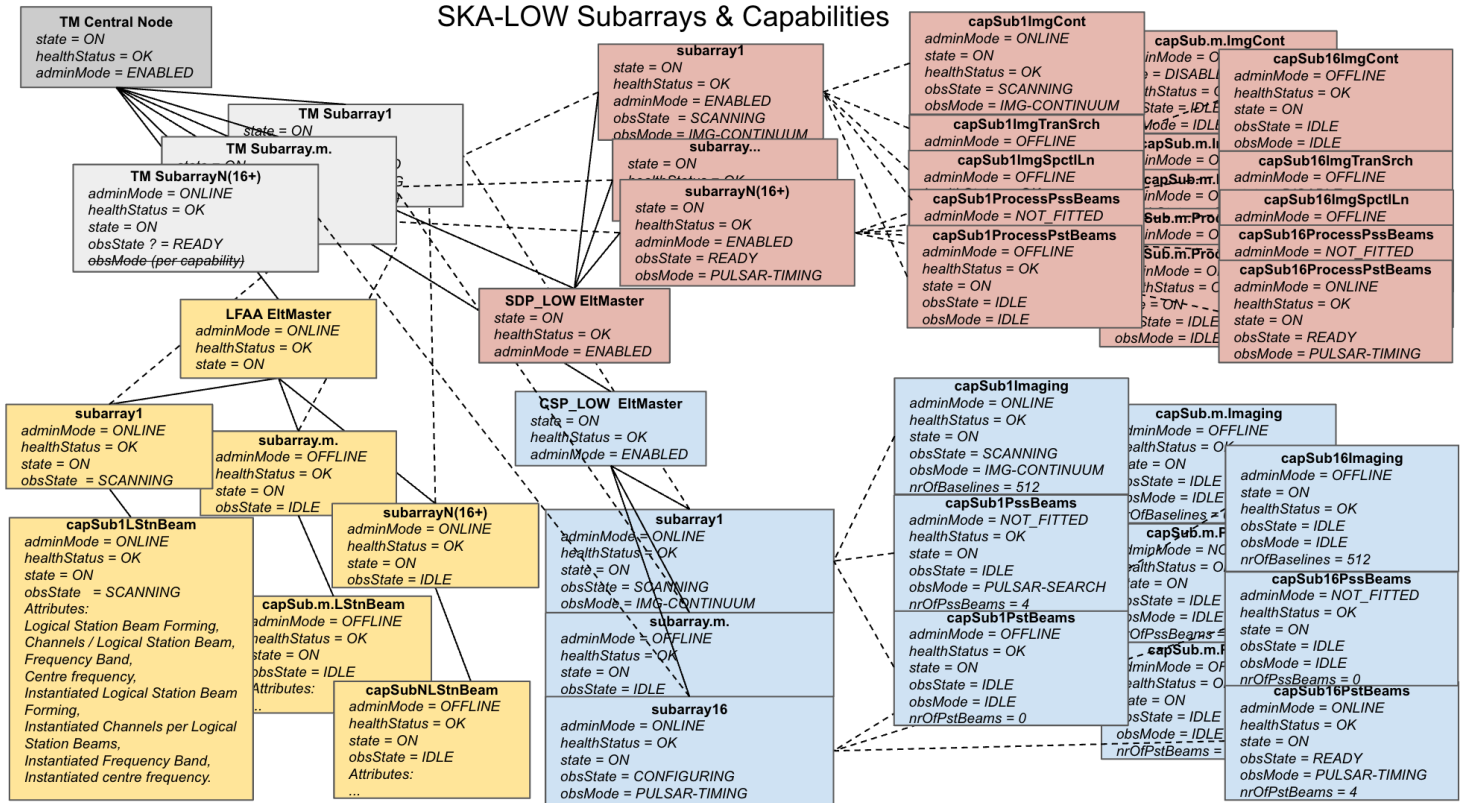
1. CSP - CBF, PSS and PST master also plays a role in the mechanics to setup and configure a subarray, but that is not included here
2. CSP subarray - `obsState` also on subarray level - reporting the overall `obsState` of the subarray. Also required per Capability as each capability will have its own configuration sequence and state machine. `obsMode` has to be on Capability level as the specific capability instructs the observing mode that is possible. Subarrays are enabled with `adminMode=ONLINE` and disabled with `adminMode=OFFLINE`
3. CSP Capability - `adminMode=OFFLINE` indicates capabilities not configured on that subarray, e.g. subarray1 in the example has only imaging and PSS beams, no PST or VLBI beams

SDP:

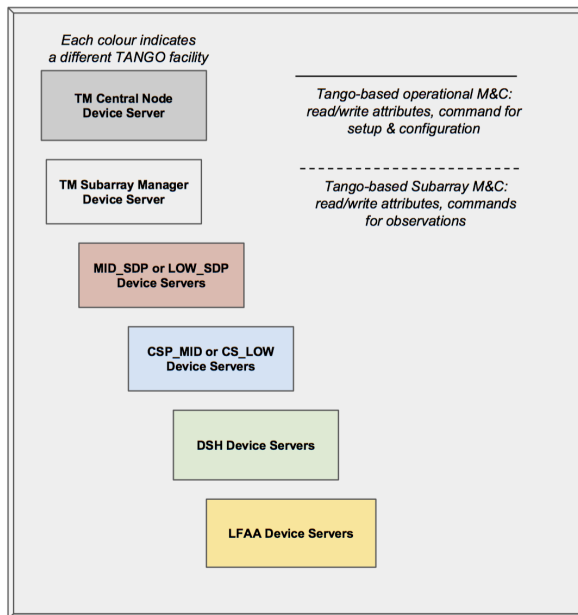
1. Also concept of "subarray" as it needs to know the "assigned resources" anyway to subscribe to the correct set of meta-data
2. And it makes SDP consistent with CSP and LFAA for control and monitoring without any cost to SDP
3. Subarrays are enabled by setting `adminMode=ONLINE`; can then be configured, monitored and controlled, and finally disabled with `adminMode=OFFLINE`
4. Capabilities are invoked by setting `adminMode=ONLINE`; can then be configured, monitored and controlled, and finally revoked with `adminMode=OFFLINE`

5.8.4 SKA-LOW SCM Example

SKA-LOW Subarrays & Capabilities



Legend for SKA-Mid and SKA-LOW Subarrays & Capabilities



SKA-LOW discussions:

See general discussion in previous paragraph in SKA-MID (but ignore the SKA-MID specific examples)

LFAA - also define subarrays (like CSP and SDP) to identify the group of resources (station, tile and antennas) in the Logical Station Beam

LFAA - assume a single Logical Station Beam per subarray

LFAA - 16 or N(16+) subarrays ?

SDP - also concept of "subarray" as it needs to know the "group of resources", specifically to subscribe to the correct set of meta-data

5.8.5 SCM attributes

state (built in TANGO device state)	Use in SKA Telescopes
UNKNOWN - The device cannot retrieve its state. It is the case when there is a communication problem to the hardware (network cut, broken cable etc...) It could also represent an incoherent situation	Device state of an entity is reported as UNKNOWN when TM or other parts of the control system (higher in the M&C hierarchy) are not able to establish communication with the TANGO Device that reports on behalf of the entity . When not able to detect/obtain the actual state of the entity it models, due to loss of communication with equipment or software processes, TANGO Device shall report state as UNKNOWN or FAULT ; the choice will depend on the context (analysis should be performed during implementation to identify appropriate value for each case).
OFF - The device is in normal condition but is not active. E.g the power supply main circuit breaker is open; the RF transmitter has no power etc...	TANGO Device shall report (operational) state as OFF only when it can detect that the entity is indeed OFF (switched off). When the entity is unresponsive and the cause is unknown, TANGO device shall report state as UNKNOWN or FAULT (during implementation analysis should be performed to identify appropriate value for each case). Examples: • TANGO device which reports on behalf of a software process or application shall report state as OFF when the process is not active (most Operating Systems are able to report the list of active processes). • TANGO device which reports on behalf of an LRU should report state as OFF only when it is able to detect that the LRU is indeed switched off or lost power. Otherwise, state of an unresponsive entity (Element, Sub-element, LRU, component) shall be reported as UNKNOWN or FAULT (depending on the context). NOTE: TANGO Rev 9 defines two aliases for this state. SKA preferred mnemonic is OFF ; CLOSE and INSERT should be used only when required to model the equipment accurately.
INIT - This state is reserved to the starting phase of the device server. It means that the software is not fully operational and that the user must wait	TANGO Device should report (operational) state as INIT when the TANGO Device itself and/or the entity it models is performing initialization; in this state the entity is not fully operational, it is not ready to perform its core functionality. Part of the functionality may be available, e.g. ability to report state and other attributes. This is a transient state, when initialization is completed, the entity transitions in the next state as per state machine. Not all TANGO Devices implement and report this state. Once initialisation has completed the device should transition to its default operational state (which for SKA is STANDBY if low-power modes is applicable, else ON)
DISABLE (hibernate)- The device cannot be switched ON for an external reason. E.g. the power supply has it's door open, the safety conditions are not satisfactory to allow the device to operate	Not all entities implement and report this state; some SKA Elements and Sub-elements are required to implement this state. For Subarray and Capabilities SKA uses DISABLE to indicate entities that are not active; e.g. in DSH the receiver bands that are not selected is set to DISABLE , and in TM, LFAA, CSP and SDP the Subarrays not "created" and Capabilities not instantiated will be set to DISABLE . TANGO Device where <code>adminMode=NOT_FITTED, OFFLINE, or RESERVED</code> shall report (operational) state as DISABLE . In general, monitoring functionality is available and limited control functionality may be available (i.e. ability to change <code>adminMode</code> or operating mode) Consideration: When TANGO v10 is available with custom labels, we MAY define the SKA alias HIBERNATE .
STANDBY (low-power) ² - The device is not fully active but is ready to operate. This state does not exist in many devices but may be useful when the device has an intermediate state between OFF and ON . E.g the main circuit breaker is closed but there	Used to report low-power state implemented by some of the entities (Elements, Sub-elements, assemblies). Not all entities implement and report this state and, if implemented, this is the default state to land in after power-up. For each entity that implements STANDBY state, documentation shall clearly

² **NOTE:** SKA has a requirement that all equipment come up in **STANDBY** state. A device representing equipment where this state is applicable shall startup and after **INIT** (if implemented) the device will transition to **STANDBY** state. If **STANDBY** is not applicable, the device shall transition to **ON**.

is no output current. Usually STANDBY is used when it can be immediately switched ON. While OFF is used when a certain time is necessary before switching ON.	describe which functions and parts of the system are available - functionality may vary. Some entities may provide additional indicator to report the percent of nominal or maximum power being used.. Note that SKA use of STANDBY is not aligned with the original TANGO definition: "The device is not fully active but is ready to operate." (SKA devices in STANDBY state are not ready to operate). Consideration: When TANGO v10 is available with custom labels, we MAY define the SKA alias LOW-POWER
ON - This state could have been called OK or OPERATIONAL. It means that the device is in its operational state. (E.g. the power supply is giving its nominal current, the motor is ON and ready to move, the instrument is operating). This state is modified by the Attribute alarm checking of the <code>DeviceImpl:dev_state</code> method. i.e if the State is ON and one attribute has it's quality factor to ATTR_WARNING or ATTR_ALARM, then the State is modified to ALARM.	All TANGO Devices shall report the normal operational state as ON. No ALARM quality factor ATTR_WARN or ATTR_ALARM is present. This is the nominal state that all TANGO Devices shall report most of the time when nothing fails during normal operations. Suggestion: When TANGO v10 is available, define the SKA alias READY.
ALARM - The device is operating but at least one of the attributes is out of range. It can be linked to alarm conditions set by attribute properties or a specific case. (E.g. temperature alarm on a stepper motor, end switch pressed on a stepper motor, up water level in a tank, etc...). In alarm, usually the device does it's job but the operator has to perform an action to avoid a bigger problem that may switch the state to FAULT.	This state may be considered as a special case of ON. Tango Device state is set to ALARM by the TANGO core when at least one attribute has quality factor ATTR_WARN or ATTR_ALARM and TANGO Device state is ON. When the ALARM quality factor for all attributes is NORMAL, the TANGO core sets the TANGO Device state back to ON. NOTE: TANGO core generates attribute alarms <u>only when TANGO Device state is ON or ALARM</u>. TANGO core does not generate attribute alarms in other states - for alarm reporting in other states custom implementation is required (see the guidelines for error and alarm reporting).
FAULT - The device has a major failure that prevents it to work. For instance, A power supply has stopped due to over temperature A motor cannot move because it has fault conditions. Usually we cannot get out from this state without an intervention on the hardware or a reset command.	TANGO device state is set to FAULT when an unrecoverable error is detected which most probably requires human intervention to recover, rendering the device unusable. NOTE: TANGO core does not evaluate attribute qualities for a device where <code>state=FAULT</code>, which means that TANGO core does not generate alarms for device where <code>state=FAULT</code>. Rationale for TANGO implementation: a device is unusable and there is no reason to get device alarm thresholds evaluated anymore. (TANGO designers did not take into account that further risks may be detected, e.g. equipment may overheat.) Guideline for implementation: TANGO Device shall transfer to FAULT only when detected issue/error requires human intervention. Also, transition to FAULT should be avoided if it is describable/required that TANGO core keeps evaluating attributes (more precisely quality factor of the attributes). It is unlikely that software devices will implement transition to FAULT, unless a truly unrecoverable error has been detected. For example: if a Software Component loses connection to a database it should be able to recover once the connection has been re-established; when the TANGO state is set to FAULT, the attribute monitoring the connection to the database is not being evaluated and would not return to ATTR_VALID when the connection is re-established. When the connection with the database is lost, set <code>healthState=FAILED</code>, do not enter FAULT state.
RUNNING, MOVING	NOT USED BY SKA Rationale: in SKA there are few moving parts, and those that are moving (like dish motor controllers) are expected to do so and it is their normal operational state. When <code>state=RUNNING/MOVING</code> TANGO core does not evaluate and raise attribute alarms (only does it in the ON and ALARM states); in SKA we choose to rather keep the TANGO <code>state=ON</code> to ensure the attribute alarms are continuously evaluated.
OFF aliases: CLOSE, INSERT ON aliases: OPEN, EXTRACT	MOSTLY NOT USED BY SKA NOTE: TANGO defines two aliases for ON, namely OPEN and EXTRACT, and for OFF, namely CLOSE and INSERT. The SKA preferred mnemonics are ON and OFF; OPEN/EXTRACT or CLOSE/INSERT should only be used when it models the equipment accurately and the use of ON or OFF would be confusing.

adminMode (memorized)	Description:
ONLINE	SKA operations declared that the entity can be used for observing (or other function it implements). During normal operations Elements and subarrays (and all other entities) shall be in this mode. TANGO Devices that implement adminMode as read-only attribute shall always report adminMode=ONLINE. adminMode=ONLINE is also used to indicate active Subarrays and invoked Capabilities.
MAINTENANCE (factory default)	SKA operations declared that the entity is reserved for maintenance and cannot be part of scientific observations, but can be used for observing in a 'Maintenance Subarray'. MAINTENANCE mode has different meaning for different entities, depending on the context and functionality. Some entities may implement different behaviour when in MAINTENANCE mode. For each TANGO Device the differences in behaviour and functionality in MAINTENANCE mode SHALL be documented. MAINTENANCE is the factory default for the Administrative Mode. Transition out of NOT_FITTED is always via MAINTENANCE (factory default adminMode); an engineer/operator has to verify that the entity is operational as expected before it is set to ONLINE (or OFFLINE).
OFFLINE	SKA operations declared that the entity cannot be used for observing or other function it provides. A subset of the monitor and control functionality may be supported in this mode. adminMode=OFFLINE is also used to indicate unused Subarrays and unused Capabilities.
NOT_FITTED	SKA operations declared the entity as NOT_FITTED (and therefore cannot be used for observing or other function it provides). TM shall not send commands or queries to the Element (entity) while in this mode. Higher level entities (Element, Sub-element, component, Subarray and/or Capability) which 'use' NOT_FITTED equipment shall report operational state as DISABLE. If only a subset of higher-level functionality is affected, overall state of the higher level entity that uses NOT_FITTED equipment may be reported as ON but with healthState=DEGRADED. Additional queries may be necessary to identify which functionality and capabilities are available. Higher-level entities shall intelligently exclude NOT_FITTED items from Health Status and SKA Alarms; e.g. if a receiver band in DSH is NOT_FITTED and there is no communication to that receiver band, then DSH shall not raise SKA alarms for that entity and it should not report healthState=FAILED because of an entity that is NOT_FITTED.
RESERVED	This mode is used to identify additional equipment that is in STANDBY to take over when the operational equipment fails. This equipment does not take part in the operations at this point in time.
healthState (report only)	Description:
UNKNOWN	Initial state when health state of entity could not yet be determined.
OK	TANGO Device reports this state when ready for use. Or when entity is NOT_FITTED or RESERVE. The rationale for reporting health as OK when an entity is NOT_FITTED or RESERVE is to ensure that it does not pop-up unnecessarily on drill-down fault displays as UNKNOWN, DEGRADED or FAILED while it is expected to not be available.
DEGRADED	TANGO Device reports this state when only part of functionality is available. This value is optional and shall be implemented only where it is useful. For example, a subarray may report Health Status as DEGRADED if one of the dishes that belongs to a subarray is unresponsive or reports Health as FAILED. Difference between DEGRADED and FAILED health shall be clearly identified (quantified) and documented. For example, the difference between DEGRADED and FAILED subarray can be defined as the number or percent of the dishes available, the number or percent of the baselines available, sensitivity, or some other criterion. More than one criteria may be defined for a TANGO Device.
FAILED	TANGO Device reports this state when unable to perform core functionality and produce valid output.
obsState	Description: Observing State of subarray/capability (as per SKA Observing Control View and SKA MID-TT State and Mode RT report)
IDLE	Sub-array, resource, Capability is not used for observing, it does not produce output products. The exact implementation is TBD for each Element/Sub-element that implements sub-arrays and

	Capabilities (is there a standard setup for an <code>IDLE</code> sub-array/Capability, or does subarray/Capability when transferring to <code>IDLE</code> state keep the latest setup and only stops transmission of the output products.)
CONFIGURING	Sub-array is being prepared for a specific scan. On entry to the state no assumptions can be made about the previous conditions. This is a transient state. sub-array/Capability automatically transitions to <code>obsState=READY</code> when configuration is successfully completed. If an error is encountered, TANGO Device may a) report failure and revert to the previous configuration, b) proceed with reconfiguration, transition to <code>obsState=READY</code> and set <code>healthState=DEGRADED</code> (if possible notify the originator of the request that configuration is not 100% successful) or c) if serious failure is encountered, transition to <code>obsState=FAULT</code> , <code>healthState=FAILED</code> .
READY	Sub-array is fully prepared for the next scan, but not actually taking data or moving in the observed coordinate system (i.e. it may be tracking, but not moving relative to the coordinate system). Elements other than TM may not be aware of this state, i.e. CSP may report subarray state as <code>SCANNING</code> .
SCANNING	Sub-array is taking data and, if needed, all components are synchronously moving in the observed coordinate system. Any changes to the sub-systems are happening automatically (e.g. DISHes are receiving pointing updates, CSP is receiving updates for delay tracking).
PAUSED	TBC by SKAO SW Architects Sub-array is fully prepared for the next observation, but not actually taking data or moving in the observed system. Similar to <code>READY</code> state. If required, then functionality required by DISHes, LFAA, CSP is TBD (do they keep signal processing and stop transmitting output data? What happens to observations that are time/position sensitive and cannot resume after a pause?)
ABORTED	Subarray has had previous state interrupted by controller and is in an undefined state. Note: DISH, LFAA and CSP probably do not need to implement this state.
FAULT	Sub-array has detected an internal error making it impossible to remain in the previous state. <i>[This shall trigger a healthState update of the subarray/capability]</i>
obsMode (report only)	Description: Each TANGO Device that implements obsState shall report the current Observing Mode. (as per SKA Observing Control View and SKA MID-TT State and Mode RT report)
IDLE, IMG_CONTINUUM, IMG_SPECTRAL_LINE, IMG_ZOOM, PULSAR_SEARCH, TRANSIENT_SEARCH_FAST, TRANSIENT_SEARCH_SLOW, PULSAR_TIMING, VLBI	The <code>obsMode</code> shall be reported as <code>IDLE</code> when <code>obsState</code> is <code>IDLE</code> else will correctly report the appropriate value. More than one observing mode can be active in the same subarray at the same time.
controlMode (memorized)	Description:
REMOTE	TANGO Device accepts commands from all clients.
LOCAL	TANGO Device accepts only from a 'local' client and ignores commands and queries received from TM or any other 'remote' clients. The Local clients has to release <code>LOCAL</code> control before <code>REMOTE</code> clients can take control again. [NOTE: <code>LOCAL controlMode</code> is not a safety feature but rather a usability feature. Safety has to be implemented separately to the control paths.]
simulationMode (memorized)	Description:
FALSE	A Real Element (entity) is connected to TM.
TRUE	A Simulator is connected to TM. (Or the real entity acts as a simulator.)
testMode (memorized)	Description:

NONE	Normal mode of operation. No test mode active.
TEST or Custom values (examples: Test , Test1 , Test2 , Test3)	Element (entity) behaviour and/or set of commands differ for the normal operating mode. To be implemented only by devices that implemented one or more test modes. The Element documentation shall provide detailed description.

5.9 SKA Configuration and Control

[Also see **Technical Note #5. SKA Configuration & Control**]

A set of Use Cases has been drafted in the Technical Note. Element Consortia need to consider whether these adequately capture their Element's use cases and give inputs on missing use cases or updates. As it stands the design patterns are still Work-In-Progress and inputs are welcome.

This topic will include design patterns for:

1. **Element configuration/sequencing:** Setting many parameters on multiple devices and the role of Element in configuration, complex configuration of multiple sub-elements concurrently.
2. **Structured command parameters & command validation:** TANGO experts and guidelines recommend to use TANGO predefined types for defining command arguments whenever possible. Employing a different data serialization schema (e.g. json, xml or others) is considered a TANGO anti-pattern and may also affect performances.

On the other hand, TANGO does not allow to pass more than one parameter as command argument, although the current proposal is to add pipe support for commands as well. See

http://www.tango-controls.org/media/filer_public/16/35/16358d7b-4cb6-4c23-a2e4-226ae2264ea3/tango_kernel_meeting_sep2016.pdf

The provided structured types are limited to TANGO::DevVarLongStringArray and TANGO::DevVarDoubleStringArray, which are not sufficient to satisfy SKA Element needs, particularly in the case of commands defined at the higher levels in the control system hierarchy (e.g. related to observation scheduling and configuration, etc). A standardized message schema will be therefore designed for SKA to handle rich arguments in commands defined on each LMC TANGO device. It is desirable to adopt the same schema also in other areas of standardization, for example when defining a suitable format for SKA device alarms/events.

This pattern will therefore address the following issues:

- Optimal serialization strategy (e.g. message format, suitable libraries in all employed languages, ...) to be adopted for structured argument passing;
 - Level of applicability of the serialization schema, e.g. if it is used only in those cases where TANGO native schema is insufficient or if it will sit on top of TANGO schema, thus overriding it for all commands;
 - Design patterns for consistent command validation across Elements.
3. **Using commands vs attributes:** According to TANGO guidelines, defining and using commands to set/get attribute values in place of the TANGO builtin read_XXX/write_XXX attribute methods is considered an anti-pattern. This SKA pattern will explore possible exceptions to this rule. One possibility could be using writable attributes and write_XXX methods for physical parameters (e.g. sensor values, motor positions etc) while read-only attributes and setter commands for modes, capabilities, etc. This pattern should address:
 - when to employ read-only attributes vs read-write attributes
 - how to handle setting of many attributes at once, e.g. when to employ write_attributes vs alternative approaches
 - is there a use case for writable "requested" attributes vs read-only "actual" attributes (e.g. for requested vs actual antenna pointing positions). Note that TANGO RW attributes have both, the requested value (setpoint) and the reading.
 4. **Long running/future dated commands:** As specified in the TANGO guidelines, long running commands (e.g. taking longer times than the predefined TANGO timeout) can be invoked by clients using the provided TANGO asynchronous API. TANGO offers also a valid tool for performing ordered sequences of operations or macros (e.g. long configuration tasks, known safety actions) via the Sardana Macro server. TANGO does not allow to execute commands at a desired future timestamp. Commands are executed by a device server as soon as they are invoked by clients, either synchronously or asynchronously. Command-queuing features could be required by some Elements or by TM to perform controlled actions (e.g. configuration, life-cycle/maintenance operations, TBD) at known times over the entire array.

For long running commands there are basically two approaches:

- a. use an async method, if you can afford the server being busy during the execution (only the client will be freed)

- b. use a dedicated thread for the command processing, reporting the ongoing progress with a dedicated attribute.

If a client asks for something already done in case a) the async command ended notification will just come immediately, in case b) the client will receive two events on the reporting attribute (reacting to the request and, immediately after, reporting "completed").

5. This pattern may in future address:
 1. Suitable strategy to enable queued timestamped commands within Element device servers in a common way
 2. Level of applicability of scheduling features, e.g. if scheduling has to be performed only at the higher levels in the hierarchy (TM-level) or also at LMC Element level, or whether simple command-queueing of future timestamped commands at the Element level will suffice in all cases
 3. Best approach to implement controlled sequencing of commands in devices
4. **Progress feedback:** How and when to indicate progress to TM; e.g. startup, upgrades, long running commands/configurations. String attribute(s) monitored for a change event or Pipes or other.
 - Progress is reported via state attributes, e.g. obsState transition from CONFIG to READY, and via supporting attributes. For commands that involve many subordinate components (TANGO Devices, LRUs, software processes), TANGO device which coordinates execution (usually representing Element, sub-element or subarray) shall implement one or more attributes to track and report progress. Progress may be reported as the number of subordinate components that completed the task, as the percent of completeness. Additionally, a list of components where task has been completed may be provided. In the case of start-up, shut-down, upgrades and other life-cycle components all three should be implemented, so that clients may choose the level of detail to access. Thresholds may be defined so that clients receive notifications automatically.
 - In addition to monitoring progress, TANGO Devices at the higher levels of hierarchy (e.g. Element Master and subarray) should be able to provide an estimate for the duration of some specific commands. The estimates may be implemented as read-write memorised attributes, to be updated by externally. This caters for a use case where knowledge of duration will improve over time and one would be able to tweak the duration configurations to better align with reality resulting in the overall system functioning better. Optionally, devices may be able to update estimates based on the previously executed commands or implementation/empirical knowledge.

SKA design patterns for Configuration and Control:

- 1) When modelling a device for Configuration and Control always apply TANGO best practices (approach and mechanism) unless SKA Guidelines recommend different approach.
- 2) Basic TANGO types can be used in Scalar, Spectrum or Image format (basically single values, 1 or 2 dimensional arrays) for cases where multiples of similar type parameters need to be configured.
 - a) Example: calibration coefficients can be managed with a Spectrum
- 3) Adhering to the TANGO guidelines, avoid "commands" where "read-write attributes" are best suited. In the TANGO model any value that can be read/set is represented by an Attribute. Attributes can be read-only or read-write depending whether the physical value they are representing can just be read or set-and-read. Some attributes may be "memorised" making them persistent over power cycles. Some attribute examples are:
 - a) motor position (read-write)
 - b) encoder position (read-only)
 - c) power supply current and voltage (read-write/read-only, depending on the operating mode i.e. voltage or current source)
 - d) antenna azimuth and elevation (read-write)
 - e) temperature sensor (read-only)
 - f) wind sensor (read-only)
 - g) calibration coefficients (read-write)
 - h) version list (read-only)
 - i) delay centre frequency (read-only, set by command or different process)
 - j) delays send rate (read-only)
- 4) Do not timestamp any command for which it is not critical (i.e. antenna pointing to a target position is time critical, but invoking a new CSP capability is invoked (after CSP mode is configured) it can just be done on command/ASAP). In some cases, to meet real requirements like a tight time specification for a CSP mode change, it may be beneficial or required to use time-stamped commands to the CSP to deliver configuration in advance of activation time to speed up re-configuration. However, there should be only a very limited set of commands for which this is required.
- 5) Multiple Attributes belonging to the same TANGO device can be read or written in one call using the

- read_attributes and write_attributes methods. Do not implement Commands with structured parameters whenever reading or writing a number of Attributes will do the job.
- 6) TANGO groups can be used to run commands or to read/write attributes across several different devices. Groups can be composed for homogeneous devices (as is the case in most instances), or for different devices implementing the same subset of TANGO attributes/commands. This can be used effectively to distribute configuration/commands to a large group of devices.
 - 7) Long running methods may keep the TANGO device server busy. As a general rule, avoid coding methods that take longer than 500ms to execute. Consider spawning a dedicated thread in a TANGO device server to deal with longer executions. An additional dedicated Attribute can be used for ongoing progress reporting, if required. The client can subscribe to events on the progress attribute or a change in state/mode to follow progress. More detail on the design pattern for command response is in the next paragraph.
 - 8) Scan Configuration progress is reported using the attributes configurationProgress and configurationDelayExpected. configurationProgress shall be updated periodically as the configuration progresses. configurationProgress of 100% (and a transition to a obsState=READY) indicates configuration completion.
 - 9) **TBC by SKA SW Architects** The philosophy of "Telescope Model solutions" need to be defined- i.e. are we shipping data and results around, or do we apply a solutions library (like katpoint) to implement calculations locally. (Distributed vs centralized).
 - 10) TANGO Pipes can be used where applicable, i.e. when Attributes are not suitable, to move larger or heterogeneous data sets. These should be analysed and identified in the ICDs. Relying on multiple attributes for data transfer is a common model in TANGO, and one should not consider that case as a rationale for using a pipe.
 - 11) Error handling strategy: TANGO relies on exceptions to report errors. Clients should always be able to cope with exceptions and react properly. Some considerations that are TBD:
 - a) Tagging data stream as invalid vs explicit exceptions/faults/alarms, e.g. what to do if only some of the FPGAs in CSP fails to configure
 - b) Retry policy
 - c) When to continue / when to abort
 - d) Delegating reporting/decision of health down to Element or Subarray (e.g. % of dishes/LRUs in CBF/SDP/LFAA in warn/error per capability indicates DEGRADED or FAILED healthState),
 - e) Judgement to continue or not should be in TM.ObsMgt, but reporting is delegated down to Element e.g. delay weights, pointing updates, calibration results
 - 12) For "TelModel" computations may put a "update time" and "lifetime" on the results (in attributes). The attributes' built-in timestamp can be used to indicate "calculated time". The consumer may get the information by subscribing to the "update time" attribute change events and reading the results from the other attributes. The consumer raises an alarm/fault if the information is not updated sufficiently frequently. Naming conventions on attributes can be used to "group" the related "solution attributes" with their "update time" and "lifetime" and these should be identified in the ICDs.
 - 13) Subarrays should instantiate TANGO groups where applicable to provide "common control" e.g. "point all dishes" or "set target position on all stations" or "update all FPGAs".
 - 14) Definition and implementation of Capabilities [Also see section [5.8.3 SKA-MID SCM Example](#) and [5.8.4 SKA-LOW SCM Example](#).]:
 - a) A TANGO devices is implemented to represent each available CapabilityType on each subarray (representing complex functionality, almost like a virtual function). For example, the CSP MID will instantiate capSubnImaging, capSubnPssBeams, capSubnPstBeams and capSubnVlbiBeams on each subarray to represent the group of capabilities on Subarray n. For example frequencyBand is an attribute of a subarray (as in: subarray is observing in Band 3), may also be defined as an attribute at the DISH Element level (DISH is observing in Band 3), but within DISH Element LMC each frequency band will be represented by a Capability TANGO device which is used to monitor and control equipment that provides Band 3 functionality.
 - b) Recommended TANGO approach is to create TANGO devices during initialization, and use state to indicate what is available for use. SKA will apply recommended TANGO design patterns, i.e. instantiate Subarray and Capability devices during initialization and not dynamically as they are invoked/revoked, based on available equipment and configuration (e.g. capSubn<CapName>). It is up to each Element to decide if each capability (e.g. each PSS Beam in CSP) is also implemented as a TANGO device.
 - 15) Subarray control - once the Subarray is activated through the ElementMaster, then subsequent observation control of the subarray will be through the Subarray device and not through ElementMaster device.

SKA Design pattern for command/response (long-running or timestamped commands):

- 1) For normal commands the Tango pattern is that nothing is returned (aka Tango::DevVoid) and a Tango::Exception in

case of failures.

Where applicable (for long-running or future timestamped commands) a standard SKA command response, including a result code and explanatory message, may be returned.

- 2) Where an SKA command response is returned, the following applies:
 - a) the `command_out` parameter shall return a `DevVarLongStringArray`. The long value shall carry the "resultCode" and the string will carry an optional descriptive message.
 - b) The result code shall be completed from a `DevEnum`: OK, STARTED, QUEUED, FAILED, UNKNOWN.
 - OK - command executed, may return an optional confirmation in the message
 - QUEUED - command accepted, will be executed at future timestamp, may update a progress indicator, or generate a `user_event`; may return an optional descriptive message
 - STARTED - command accepted and execution will start immediately; may update a progress indicator, or generate a `user_event`; may return an optional descriptive message
 - FAILED - command could not be executed; returning the descriptive message recommended
 - c) The message part of the response may never be parsed by the clients to make decisions on and is purely meant as additional information, e.g. for displaying results to operators/engineers/commissioners, or add to log stream of the client to capture activity.
 - d) Where monitoring of progress on a long-running or future time-stamped command is required, the device can define a specific `xxxProgress` attribute that can be updated, or a related "state" attribute to be monitored by the client for change events.
[NOTE: The attributes related to and affected by commands from TM shall be identified in the ICD.]

Some use cases for normal TANGO commands are:

- 1) Command cannot be executed, e.g. state machine deny execution, failures, failed to be scheduled, or command is not recognised etc
--> throw exception
- 2) Command was already scheduled and is being currently executed (for example dish configuration). The system is busy implementing the same command or already in the requested state
--> return nothing, accept (no-op) the command and don't throw an exception (TBC)

Some use cases for TANGO commands with a result code are (e.g. long running or time-stamped commands) are:

- 3) Command cannot be executed, e.g. requested mode/capability not installed, resources not available
--> return flag FAILED+informative message
- 4) Command already executing
--> return flag STARTED+informative message
- 5) Command will be executed later and is put in a queue
--> return flag QUEUED+informative message
- 6) Command can be executed and a worker thread is started
--> return flag STARTED+informative message
- 7) Command can be executed but the final action is already in place (no-op). For example set an operating mode that is already present (e.g. previously applied). Not worth to reapply it (unless forced?)
--> return flag OK + informative message,
like for example ("Mode already applied") and set status/progress attribute to "completed" as applicable
- 8) other cases?

Notes on control loops during scan execution:

- 1) Updates during scan execution (while `obsState=SCANNING`):
 - a) Jones Matrices:
 - LOW:
 - PST is 301989888 Bits = ~36MB.
 - PSS is 18874368 bits = ~2.25MB
 - PST - Jones matrix applied per coarse channel, station and beam.
 - PSS - Station Jones matrix applied per coarse channel (same for all beams)
 - MID - changes are imminent, will publish as soon as new estimates become available.
 - SDP to publish Jones Matrices as array attributes and TM and CSP.LOW will subscribe. May also use PIPE, or even alternative protocol like SPEAD, depending on size of data, throughput required, archiving requirements and such. To be clarified in ICD work.

- Jones Matrices may be implemented as tables (vectors or images). Time when a new set of values becomes valid (i.e. when CBF shall start using a new set of coefficients) and the lifetime shall be indicated. Ideally, a new set of coefficients, activation time and life time should be identified by naming convention to "group" the set of attributes together. CBF would subscribe to the set of attributes and read the results when on "trigger" attribute change event. The results and their related trigger attributes to be identified in the ICD work.
- b) Calibration of instrumentation delays - short SB point at calibrator, calculates instrumental delays and provide the solutions to TM (partly pointing correction and partly delay correction).
- c) CSP prefers TM to sum all the delays together into a single delay correction.
- d) As a part of a scan configuration TM provides FQDN (or other pointer) where to subscribe for pointing, delay tracking, Jones Matrices, weights used in beamforming, RFI updates, etc. CSP can subscribe directly to SDP TANGO Device, but the information where to subscribe should be received via TM.

5.10 Common Alarm handling

[Also see **Technical Note #8. Element & Central Alarms Handling**]

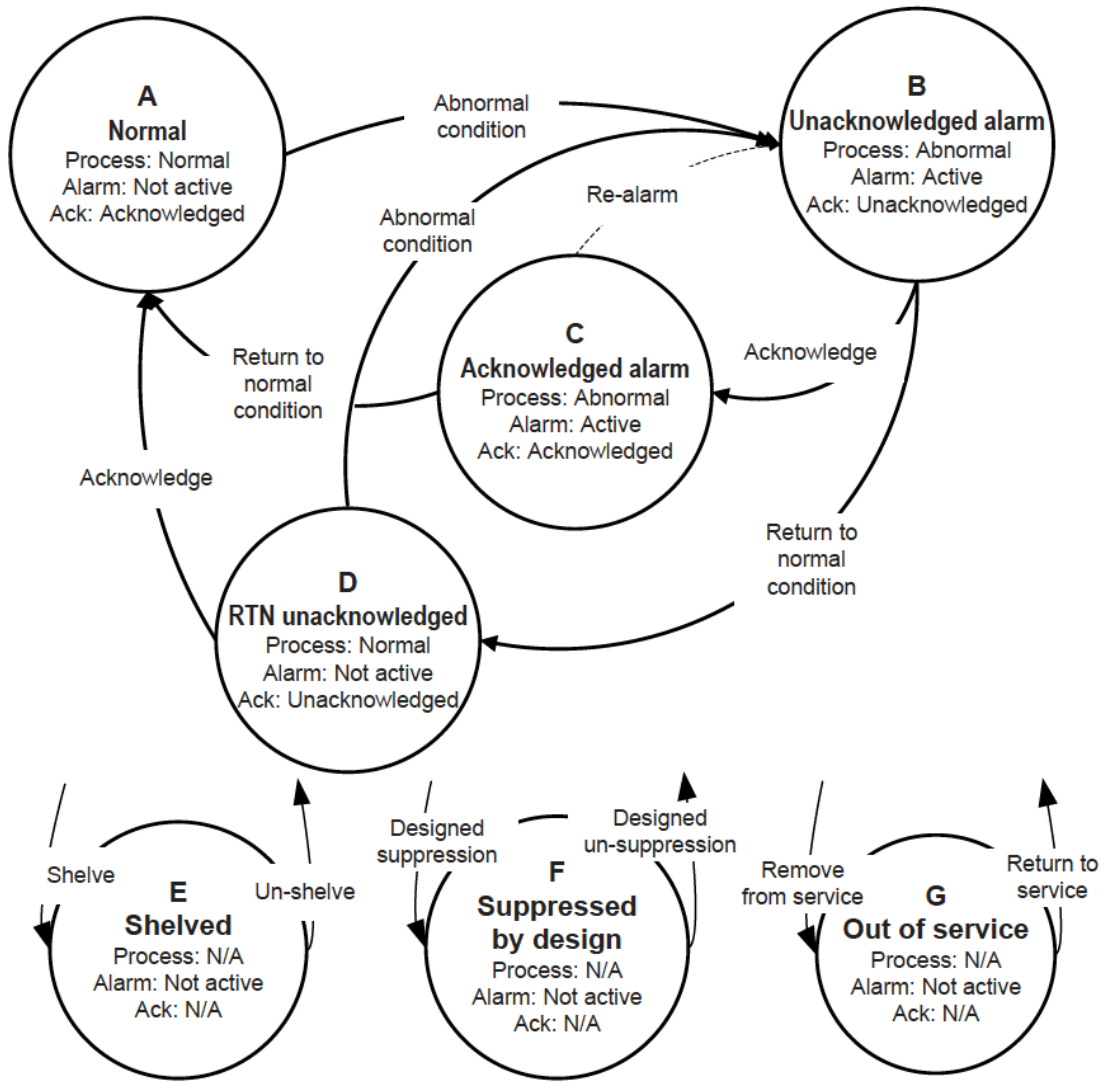
SKAO has adopted IEC 62682 Alarm Management worldwide standard for the SKA project. An infographic on the standard can be found at http://www.exida.com/2015/Alarm_Management_Rationalization_Poster_Sept2015.pdf

NOTE The SKA Alarms design pattern has been reworked to align with the adopted standard. The detail in the working document needs to be aligned with this summary, or may remain as is and be noted as background information.

IEC 62682 ALARM STATES

ID	Mnemonic	State name	Process condition	Alarm status	Annunciate status	Acknowledge status
A	NORM	Normal alarm state	Normal	Inactive	Not annunciated	Acknowledged
B	UNACK	Unacknowledged alarm state	Abnormal	Active	Annunciated	Unacknowledged
C	ACKED	Acknowledges alarm state	Abnormal	Active	Annunciated	Acknowledged
D	RTNUN	Returned to normal unacknowledged alarm state	Normal	Inactive	Annunciated	Unacknowledged
E	SHLVD	Shelved state	Normal or Abnormal	Active or Inactive	Suppressed	Not applicable
F	DSUPR	Suppressed-by-design state	Normal or Abnormal	Active or Inactive	Suppressed	Not applicable
G	OOSRV	Out-of-service alarm state	Normal or Abnormal	Active or Inactive	Suppressed	Not applicable

IEC 62682 - Alarm State Transition Diagram



IEC

NOTE 1 States E, F, and G can connect to any alarm state in the diagram.

NOTE 2 The dotted line indicates an infrequently implemented option.

Read par [3.2 Clarifications re failures, alarms and health state](#) defining Tango attribute alarms, Tango device alarms, Element Alarms and SKA Alarms. Note that the following design pattern comprise architectural design decisions for the SKA project and are not TANGO limitations.

The definitions are:

- **TANGO attribute alarm** - This refers to the condition where TANGO core sets the attribute quality to ATTR_ALARM because the attribute value moved into its min/max warning or min/max alarm range, also resulting in the TANGO device state to be set to ALARM.
- **TANGO device alarm** - Thes refers to the condition when a TANGO device state is ALARM. When any attribute on a device has a quality factor of ATTR_ALARM TANGO core sets the device state to ALARM.
- **Element Alarms** - Element AlarmHandlers implement "Element Alarms" for all conditions that the TM needs to monitor during operations. Element Alarms will not strictly adhere to the IEC 62682 definition of "alarm" (in that it requires an action by the operator). Element Alarms may be defined even if it is only a notification to TM or the operator, or if it requires automated action by TM and not operator action. Each Element Alarm is exposed as an attribute on the Element AlarmHandler device evaluating a rule-based formula which may be a combination of attribute values, attribute quality factor (i.e. TANGO attribute alarms), device states.

The Element AlarmHandlers are delegated with the responsibility to interpret the conditions within the Element and raise Element Alarms for whatever conditions the TM need to know of.

The TM Central AlarmHandler is expected to only connect to the Element AlarmHandlers and not to various lower level monitoring points for general operations. The Central AlarmHandler will in turn apply rule-based formula across the Element alerts to implement Telescope level "SKA alarms".

- **SKA alarm** - A "SKA Alarm" is any Element Alarm (or combination of Element Alarms) that requires operator action (as per the IEC 62682 definition of alarm). This will be configured at the Central AlarmHandler in TM.

Central AlarmHandler is responsible to implement the IEC 62682 standard in terms of which Element Alarms are deemed SKA Alarms, present the alarm display for SKA Alarms and manage the operator interaction with SKA alarms. Additional displays for Element Alarms will be provided with notifications to operators and information about automated actions on Element Alarms.

It is foreseen that Element Alarms will not strictly adhere to the IEC 62682 definition of "alarm" (which defines that an action by the operator is required), as it may only be a notification to TM or the operator, or it may require automated action by TM and not require action by the operator.

Where an Element Alarm does not escalate to an SKA Alarm, it may be classified as "Suppressed by Design" by TM in the Central AlarmHandler.

TANGO attribute alarms are the building blocks, together with combinations of TANGO attribute values and attribute quality factors to build **Element Alarms**, implemented by Element AlarmHandlers. TANGO attribute alarms triggers **TANGO device alarms**.

Element Alarms can be dealt with at different levels before reaching the operator, and are the building blocks for SKA Alarms.

SKA Alarms, are the ones which are defined to have a full IEC 62682 lifecycle, and presentation to and interaction by the operator. SKA Alarms can be derived either bottom-up, from the risks and reported alerts from the Elements, and top-down, from the class of condition changes that operators in similar facilities need to be aware of. SKA Alarms need to be defined at system level, and needs input from Operations, Logistics and System Engineers.

The goal of this design pattern is to design a mechanism such that there is no bottleneck for a system the size of SKA.

Therefore the Element level alarm evaluation is delegated to each Element and the instruction is that whatever conditions the TM need to know of must be interpreted within the Element and must be exposed on the Element AlarmHandler as Element Alarms (including the TM itself). Devices from within the Element should not simply send through their TANGO attribute alarms to TM and expect TM to interpret what that means on behalf of the Element; the Element is in charge to do that evaluation in the Element context and present the summarised judgment in Element Alarms.

If many devices from across Elements simply sent their attribute alarms to TM for interpretation, that means that TM can become incredibly complex having to interpret conditions on behalf of the Elements, and can potentially be out-of-date with respect to alarm design, which may create hazards in the system. The Elements are better placed to do that evaluation in the Element context. For example, in a vacuum-cooled receiver there may be many attributes going to ATTR_ALARM when the vacuum is lost, but these should be interpreted by the Element AlarmHandler to raise an Element Alarm that notifies TM of "Vacuum broken" instead of multiple separate attribute alarms.

This way the Central AlarmHandler also does not become a bottleneck as it does not have to connect to numerous attributes on different devices within Element facilities to evaluate a myriad of rules. The Central AlarmHandler is expected to only connect to the Element AlarmHandlers and not to various lower level monitoring points for general operations. In most elements (except CSP) there will probably only be one Element AlarmHandler and thus the Element AlarmHandler will not introduce any additional latency. In the case of the CSP, if there is a CBF, PSS and PST AlarmHandler we may consider to let Central AlarmHandler connect directly to those, and not just the CSP Element AlarmHandler for better latency.

TM Central AlarmHandler has the responsibility to determine which Element Alarms (or combinations thereof) is raised to SKA Alarms. This will be rules-based and configured at the TM Central AlarmHandler.

TM also has the responsibility to present the SKA Alarms and Element Alarm displays with appropriate functionality to manage operator interaction with SKA alarms, notifications to operators for Element Alarms and any automated actions that TM may implement on Element Alarms. For the TM, the TM AlarmHandler is the one that subscribes to the various monitoring points across the TM components as per the TM alarm rules, and the Central AlarmHandler should only connect to the TM AlarmHandler and not to all TM components. The SKA design pattern defines that the Element AlarmHandler has to evaluate the rules of the Element Alarms across the attributes and raise the appropriate Element Alarms.

The complexity of this approach, which we will have to work through in detail in the coming months, lies in the Element level decisions of what conditions to expose as Element Alarms - especially with regards to Element, Subarray and Capability states and healthStates, because it is assumed that the TM LeafNodes and TM Subarray Nodes will already monitor

the Element, Subarray and Capability state and healthState, and it may(?) be sufficient for alarms related to those to be raised via the TM and not duplicated on the Element AlarmHandler.

In addition, a lot of emphasize has to be placed on the Elements to model their equipment accurately in their ElementMaster, Subarray, Capability and Element AlarmHandler devices; and, underlying all this, to appropriately define the warning and alarm ranges on all attributes to ensure those can be used as building blocks for Element Alarms.

The main SKA design pattern for alarms is as follows:

1. Each Element will interpret the monitoring points, states and health of its devices and define appropriate Element Alarms to be reported to TM.
2. The Central AlarmHandler will interpret the Element Alarms and raise SKA Alarms as per IEC 62682 Alarm Management standard, whenever required. Some Element Alarms will not translate into "SKA Alarms" requiring input from the operators. TM Central AlarmHandler will be responsible for management of operator interaction with "SKA Alarms".
3. Element AlarmHandlers and Central AlarmHandler will be based on an SKA AlarmHandler base class that shall expose an alarm attribute for each Elements and SKA alarm (for which the naming convention and format of the attribute value and the behaviour of the quality factor will be defined). Each Element and SKA Alarm will be configured as a rule-based aggregation based on attribute values, quality factors, device states and modes.
4. The suggestion is that Element and SKA alarm attributes are somewhat similar to the current alarm attribute on the Elettra Alarms DS, except that there will be one attribute per each Element Alarm and each SKA alarm.
5. **New** Note that it is the responsibility of the Element to intelligently evaluate the attributes of all TANGO devices in its facility and apply rules to expose the appropriate Element Alarm Attributes for any condition that an Element identifies as being necessary to report to the TM Central AlarmHandler.
6. **New** Not all Element level alarms are required to adhere to the IEC 62682 definition (which would have implied that each Element Alarm should require operator input, which is not the case for Element Alarms).
7. **New** TM Central AlarmHandler will subscribe to all Element Alarm attributes and apply Telescope level SKA Alarm conditions to determine which conditions are IEC 62682 alarms to be presented to the operator for action.
8. **New** We therefore introduce the term "SKA alarm" and use it for Telescope wide SKA Alarms that are configured for display to and action by the operators as per IEC 62682. Some other conditions on Central AlarmHandler may be implemented to ensure a consistent evaluation and trigger mechanism is in place but may not be IEC compliant SKA Alarms. E.g. an alarm for automatic wind stow of dishes may not be deemed a "SKA Alarm" since the action will be automated within TM and the operator will not be required to respond to it. This condition does not get raised as an "SKA Alarm".
9. ElementAlarms device will expose the FQDN/alias of any child AlarmHandler devices to support TM drill-down on alarms by viewing the lower level alarm attributes exposed on the child AlarmHandler devices. Central AlarmHandler subscribe to all the Element Alarm attributes on each Element AlarmHandler.
10. **SKA Alarm attribute name:** The SKA Alarm attribute name shall have "alarm" as prefix and thus the attribute will be named "alarm<AlarmName>". E.g. alarmRfiSystemWarning, as an example.
11. **SKA Alarm attribute value:** will be coded into a DevEnum, in the ID range A-G, as per the IEC 62682 (see table **FIXME**). Foreseen DevEnum labels are NORMAL, ALARM, ACKNOWLEDGED, RETURNED_TO_NORMAL, SHELVED, SUPPRESSED_BY_DESIGN, OUT_OF_SERVICE.
12. **Additional alarm information:** The AlarmHandler shall manage additional information for each alarm and report it through standard commands or Pipe on request. **TBC**

[NOTE: these implementation details may change in future as there are currently work being done on some of the existing TANGO alarm handlers (PyAlarm and Elettra Alarms) amongst other things to align with IEC 62682, but the detail will be included in an SKA AlarmHandler base class that each Element will use and thus the focus for now can be on identifying the conditions to be reported to TM and defining the rules for those.]

Standard commands that may be implemented by all AlarmHandlers can be:

- GetAlarmRule - Get all configuration info of the alarm, e.g. rule, defined action, etc - if not yet available
- GetAlarmData - Get list of current value & quality factor & status of all attributes participating in the alarm rule; alarm priority, severity and group may be included
- GetAlarmAdditionalInfo - additional info as per TM requirements. This message is necessary to get additional alarm information which will be included in structured fields such that the client can extract the information.

[NOTE: can be used for additional information like source-of-alarm, cause-of-alarm, desired-response]

- SetAlarmPriority:
 - i. NEW - set by system that first detects transition from NORMAL status to ALARM
 - ii. ACK - set by operator when he/she has reacted to the new alarm
 - iii. SHELVED - set by operator to temporarily suppress alarm reporting **TBC** or just handled in the

operator GUI and not delegated to the AlarmHandlers

- iv. NONE – set by operator to clear managed alarms (only valid if alarm status is NORMAL)

Alternative: **TANGO Pipe** can be used to carry the additional alarms information. When the client needs additional alarm information it can write the alarm name into the pipe and read back all the related additional information on the Pipe.

13. **SKA alarm attribute value:** shall be of the format (e.g. json dictionary or DevStringArray) containing:

status	(NORMAL, ALARM, SUPPRESSED_BY_DESIGN, OUT_OF_SERVICE) Actual status of the Alarm. If alarm condition is present, then status will be either: ALARM (for the normal case) or SUPPRESSED_BY_DESIGN (if NOT_FITTED) or OUT_OF_SERVICE (if MAINTENANCE) TBC - should SUPPRESSED_BY_DESIGN and OUT_OF_SERVICE always be reported for those cases where it applies even when alarm condition is NOT present, or report SUPPRESSED_BY_DESIGN/OUT_OF_SERVICE only when the condition is Present i.e. in place of ALARM, else report NORMAL - Is this specified in the IEC standard?
priority	(NONE, NEW, ACK, SHELVED) - user interaction with the alarm, NONE indicates alarm is not present, NEW is set when a new alarm triggers, ACK is set by operator to indicate that action has been taken, SHELVED is a temporary suppression of the alarm by the operator
severity	(fault, warning or log) <i>[In current TANGO systems this is used for filtering based on severity, whenever a large number of alarms are triggered. In this scenario you may want to first concentrate on System Alarms, that's anything that should be addressed immediately.]</i> TBC Does SKA need this now that we've adopted IEC 62682?
group	Multiple groups are separated by the pipe ' ' character <i>[In current TANGO systems group is also used for filtering to support users (technicians or engineers to visualize only the alarms belonging to a specific subsystem, e.g anything that belongs to the RF plants, or to the vacuum subsystem). This can be addressed to some extent in the alarm name, and the source of the AlarmHandler, but may results in too long Attribute names.]</i> TBC how is this used? Does SKA need this?
timestamp	Superseded for SKA by the attribute timestamp
alarm_name	Superseded for SKA by the attribute name
acknowledge	Superseded for SKA by priority
message	For SKA will be included in GetAlarmAdditionalInfo
new_status	Superseded for SKA by priority

14. **SKA alarm attribute quality:** Use the Quality Attribute of the SKA alerts and SKA alarms attributes (ATTR_VALID and ATTR_ALARM) to indicate when the condition is present or returns to normal e.g.
- o when the alarm value has status=ALARM, the alarm attribute's quality is ATTR_ALARM
 - o else (i.e. value has status=(NORMAL|SUPPRESSED_BY_DESIGN|OUT_OF_SERVICE)), the alarm attribute's quality is ATTR_VALID
15. TM CentralAlarms/Operator GUI will manage alarm shelving/suppression as required.
Does SKA want to implement delegation of alarm Shelving to AlarmHandlers and not just in the Alarm GUIs? No - TM will manage alarm shelving/suppression except for SUPPRESSED_BY_DESIGN (not_fitted) and OUT_OF_SERVICE (maintenance)

Alarm Processing

1. When an entity is in adminMode=NOT_FITTED the alarm value always has (status, priority) = (SUPPRESSED_BY_DESIGN, NONE).
2. When an entity is in adminMode=MAINTENANCE the alarm is always (status, priority) = (OUT_OF_SERVICE, NONE).
3. An alarm that is not present has (status, priority) = (NORMAL, NONE)
4. When the condition triggers the alarm changes to (status, priority) = (ALARM, NEW)
5. When operator acknowledges the alarm the priority changes to ACK
6. When operator suppresses an alarm the priority changes to SHELVED
7. When operator un-suppresses an alarm the priority changes back to ACK if alarm is present, else NONE
8. When alarm triggers with status, priority = (ALARM, NEW) and clears without an operator having acknowledge the alarm the alarm will be status, priority = (NORMAL, NEW) which indicates to the operator that an alarm has been present in the meantime.

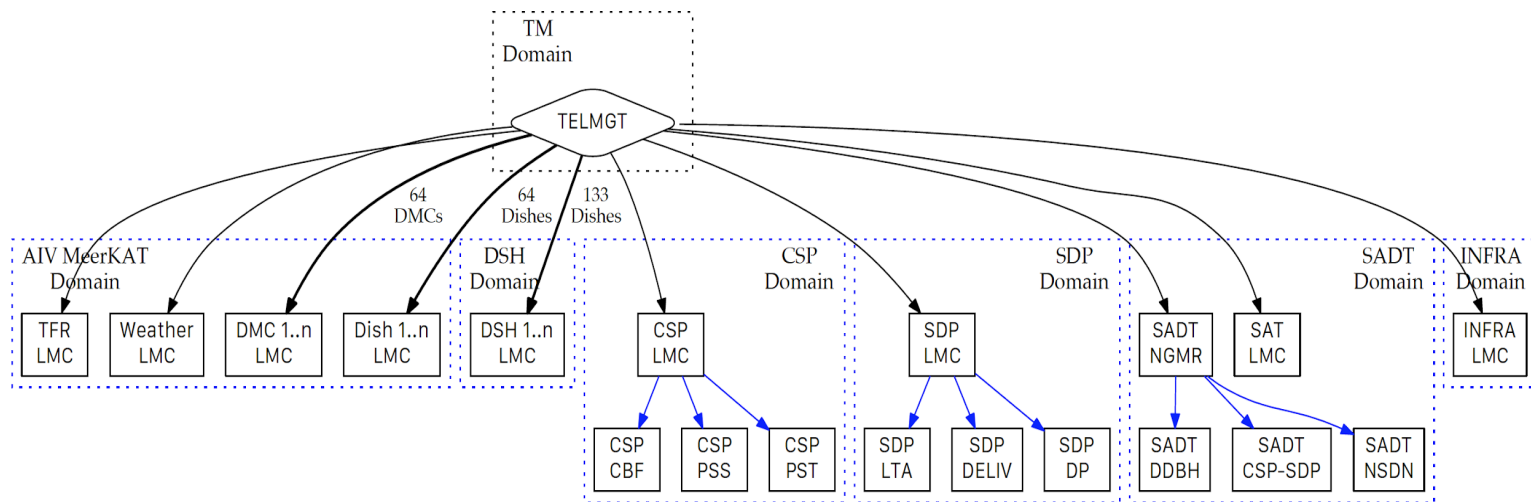
For more background consult the working document.

NOTE: The SKA project will need to develop extensions/adaptations to the Elettra Alarms, PyAlarm and Alarm viewer to support this design pattern. Currently the Elettra Alarms DS exposes a single "alarm" attribute that issues a change event when any alarm triggers. Elettra has indicated that work is planned to expose an alarm attribute for each alarm (similar to FormulaConf) and to integrate the FormulaConf rules engine into the device server. The current Elettra alarm attribute value is a string with fields separated by tab '\t' character. The fields in the current Elettra alarm attribute are:

time_stamp (seconds since UNIX epoch, 1/1/1970) of the instant when this alarm changed status);
microseconds (microseconds of the instant when this alarm changed status),
alarm_name (alarm name);
status (NORMAL or ALARM);
acknowledge (one of 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 (one of fault, warning or log);
group (multiple groups are separated by the pipe '|' character);
message (optional string);
new status (if it is the first time this alarm is read in the ALARM status, the string NEW is added as the last field).

Appendix A: SKA Control Hierarchies

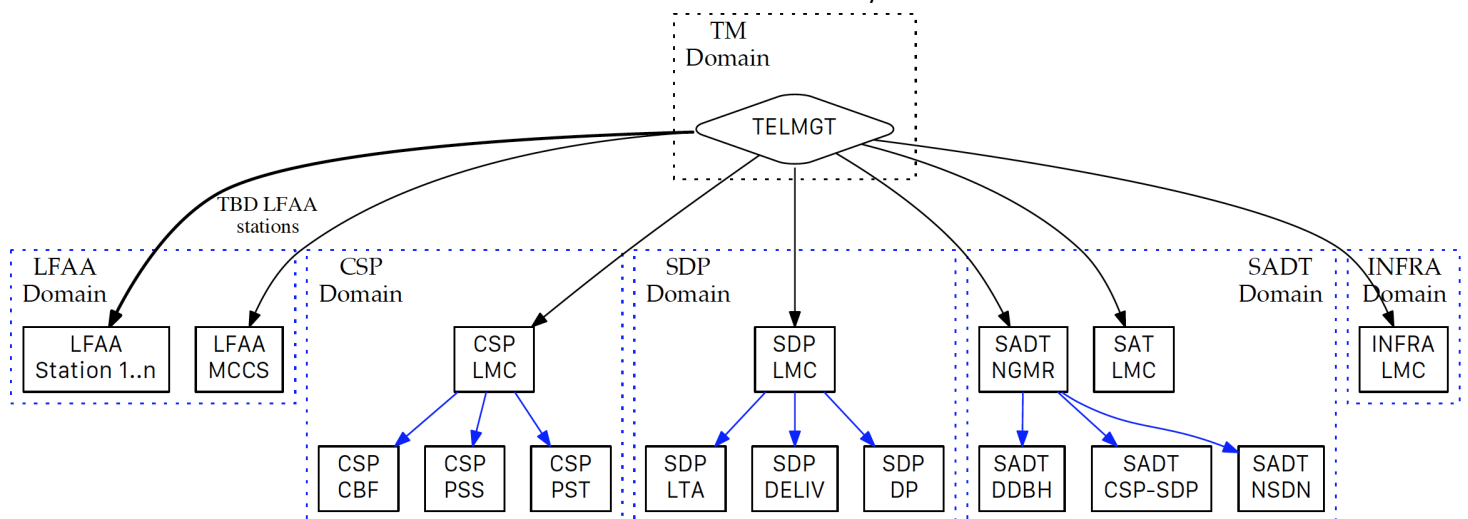
SKA-MID Control Hierarchy



TBD - get source of diagram and update.

Sub-elements to SAT LMC are SAT CLOCKS, SAT FRQ, SAT UTC

SKA-LOW Control Hierarchy



TBD - get source of diagram and update.

Sub-elements to SAT LMC are SAT CLOCKS, SAT FRQ, SAT UTC

Appendix B: Element LMC Scope and Responsibilities

The following roles for an Element LMC been taken from the original LSR document are mostly still applicable.

General role

Element role:

1. Implementation of interaction protocol between LMC's and TM that comprise of higher level message exchange formats that reside at the application layer of the OSI.
[**: this is realised by the Tango framework]
2. Setup and configuration of the Element's sub-elements/components/Capabilities on instruction from TM and implementation of interface to set and get configurable parameters of the Element/sub-elements/components/Capabilities.
[**: to be described in the TM-<Element> LMC ICD]
3. Internal control of the Element's components to achieve higher-level goals through Capabilities.
4. Definition of higher-level control model (valid command sequences, their parameters and attributes, typically in the form of a state machine) in accordance with the "LMC Interface Guidelines", and capturing the control model in the TM - <Element> LMC ICD.
[**: a standard SKA Control Model for all Element LMCs will be described in the Guidelines documents]
5. Implementation of SKA Control Model. This is the mapping of the Element's internal state to an abstract common state & mode indicators (as defined in the CS_Guidelines documents) E.g. Initializing, Ready, Faulty etc.
6. Representation of Element functionality through higher level Capabilities.
7. Monitor the Element as a system, and its hardware and software components and provide rolled-up and drill-down views of health and status of the Element hierarchy, configuration parameters, behavioral and performance parameters and event logs.
[**: Rolled-up health and status for the Element and its Capabilities have to be implemented by the Element LMC by intelligently interpreting the lower level device hierarchy of the Element. Drill-down views of health and status of the Element hierarchy is built into the Tango control system framework and no additional work is required by the Element LMC.]
8. Monitor the Element Capabilities and provide rolled-up views of health and status of the Element Capabilities.
9. Escalate alarms to TM that need non-local handling or be made visible to operators / engineers / scientists for diagnostics and/or science data interpretation. Alarms that can be dealt with automatically and locally (by the Element LMC) are the responsibility of the LMC. A subset of these local alarms may be reported to the TM, based on LMC configuration and TM's subscription for items of its interest. The LMC has to implement any immediate response and actions to handle the condition.
10. Identify and handle all local safety scenarios involving both human safety and equipment protection, especially those requiring a short-term response. The LMC itself must be able to maintain the integrity and safety of the Element. TM may provide a secondary redundant level of safety protection for preventative actions.
11. Maintain high-level connectivity (like responding to watchdog/check-alive requests and sending heartbeats) with TM and provide a stream of monitoring data and events/alarms as agreed upon in the ICD.
[**: This is provided by the Tango control system framework]
12. Provide an interrogation/discovery (self-describing) interface to retrieve the set of valid commands, parameters, responses, monitoring points, events/alarms.
[**: This is provided by the Tango control system framework]
13. Provide remote diagnostic support for the Element, including Element specific queries as defined by each Element LMC and captured in the Element LMC ICD.
[**: This is provided by the Tango control system framework]
14. Provide remote update/upgrade support for the Element.
15. Provide local override (local control mode) in order to control and diagnose the problem locally.

Setup and Configuration

Element role:

1. Implement setup, configuration and calibration of all its components based on local default configuration, or TM triggers (as defined by the LMC in the ICD).
2. Accept configuration commands from TM (as defined by the LMC in the ICD) and implement them across its internal components.
3. Respond to TM commands and configurations (accept/reject).

4. Report change of status for components and Capabilities (configurations) to TM.
5. Manage the Element state/mode and publish the aggregated state to TM in accordance with the SKA Control Model. TBC
6. Report configuration parameters, software and firmware versions and hardware serial numbers.
7. Maintain health, including detecting and isolating faulty components, manage interlocks logic and triggering component replacement as needed. TBC
8. Select components to form Capabilities that expose higher level abstractions to TM, rather than expect TM to deal with individual components within the Element. It is expected that LMC has intimate knowledge of all its internal components, while TM has only minimal knowledge related to orchestrating logic across components within an Element, higher-level event/alarm handling and overall status management.
9. Handle not-fitted and maintenance internal components intelligently and suppressing irrelevant events/alarms and data streams, excluding not-fitted components from failure detection reports.
10. Forward configuration request from TM, to all affected components (if they are able to handle future configuration changes) and in general to prepare for configuration change.

Control

Element role:

1. Implement detailed control and coordination of the individual internal components within the Element to present a high level control abstraction to TM, typically in the form of an abstract SKA Control Model state machine and commanding and configuring Capabilities. Element LMC utilizes SKA Control Model to identify the set of legal commands from TM, the (abstract) states in which they are valid, and the effect of those commands and provide the same as part of self-describing interface.
2. Define the mapping from the set of Element states to the common SKA Control Model defined by TM.
3. Provide an interrogation self-discovery interface to retrieve the list of legal commands, with parameters, description and return status of each command.
[**: Interrogation of the commands, parameters and attributes of any Tango device is provided as a standard mechanism in the Tango control system framework.]
4. Manage dependent and sequenced behaviour among its own components (For example, startup a component only when cryo has reached the required temperature).

Monitoring

LMC Role:

1. Monitor health status, performance, behavioral parameters and configuration of all sub-elements, assemblies, components and LRUs. Implement preliminary processing at the Element level to diagnose faults and failures and take corrective action. Escalate events and fault situations to TM and operators as needed to facilitate the right longer-term responses. Provide advisory inputs to TM on higher level detection and response actions, including recommended operator actions.
2. Interpret the monitoring points on the Element and the current conditions. It reports rolled-up Element state as per sub-elements/assemblies/component states (down to LRU level) and Element Capabilities to TM. Element LMC advise TM on what are the options to deal with particular situations and the criteria on which decisions should be made among the options. Advisory inputs will be provided during design phase.
3. Provide rolled-up and drill-down views of the health status and performance parameters, to enable hierarchical reporting and diagnostics.
4. Report on the availability and rolled-up health of Capabilities, to enable observation scheduling.
5. Provide sufficient monitoring to update that portion of the Telescope data that falls within the Element responsibility to ensure the Telescope data is synchronized across the whole system at all times.
6. Element LMC should intelligently exclude assemblies/components which are not-fitted or maintenance, to prevent false failure reports and spurious data as appropriate.
7. Maintain logs of relevant events/alarm and failures. Include them in the monitoring data streams as appropriate, to be included in the engineering logs for diagnostics and science data interpretation purposes.
8. Implement fault detection, report failures, report failure prediction and usage monitoring points (provide early warning for impending failures and to schedule preventative maintenance) as derived during an FMECA process of the Element.
9. Allow configuration of the monitoring and reporting functionality (frequency of monitoring, what is to be monitored and reported, reporting mechanism).

[**: The Tango control system framework provides standard mechanisms for configuring polling and archive events.]

10. Provide an interrogation of monitoring points, alarms and events on the LMC interface (self-describing interface).

[**: The Tango control system framework provides standard mechanisms for interrogation of Tango devices.]

Alarms

Element Role:

1. Detect and report alarms, including situations that may affect safety of people or other entities in the environment, or result in damage of equipment. Provide timely response actions to deal with safety-critical situations and alarms.
2. Advise TM on the levels/criticality/severity of alarms and desired response actions, and provide interrogation interfaces. Provide input on alarm correlation as part of self-describing interface or during ICD definition, to help TM prevent alarm floods.
3. Provide configuration of event/alarm related behaviour.
[**: The Tango community provides AlarmHandlers that have a standard mechanism to configure alarms, which are rules based on any attributes.]
4. Provide mechanism to suppress specific alarms.
[**: Tango community AlarmHandlers uses a time period to suppress alarms. TBJ if additional mechanism to suppress alarms via a command from TM to Element LMC is required.]
5. Provide mechanism to allow definition of additional alarms over the LMC interface (TBJ).
[**: The Tango community provides AlarmHandlers that are rules based on any attributes. TBJ if TM need to be able to define additional alarms. TM could have its own AlarmHandler with specific rules to provide additional alarms when required without involving the Element LMCs.]

Remote Diagnostic Support

Element Role:

1. Identify faulty LRUs and maintenance required on components. Take faulty components out of service. Provide notifications to TM of faulty components and maintenance requirements.
[**: Tango attributes for health status should be used to realise this]
2. Implement monitoring points based on the FMECA analysis as appropriate to detect the cause of faulty behavior

[**: Tango attributes should be used to realise this]
3. Provide facilities to support FMECA activities of TM, operators and engineers, including drill-down mechanisms, conformance to a defined consistent reporting structure to enable standardized drill-down and diagnostics capabilities.
[**: Tango framework provides a standard mechanism for drill-down and diagnostics]
4. Provide remote diagnostic interface to engineers that they can directly interact with the Element or component, including remote logging through the Element LMC interface (the level of which should be configurable).
[**: Tango framework provides a standard mechanism for drill-down and diagnostics by connecting to the Element Tango Facility directly]
5. It is highly desirable to implement local logging to support fault finding during integrating, commissioning and troubleshooting. Local log files should be remotely downloadable. Local logging, where available, should use rotating files (or cycle logs in a single file or just continuing to log forever and so on).
6. Provide support for remote logging, including configuration of remote logging (destination, logging format, logging levels and so on).

Equipment safety

Element Role:

1. Trigger shutdown or other self-protection actions (For example, stowing antennas), when threats are detected.
2. Reporting safety conditions to TM as alarms.
3. Implement safe shutdown procedures when commanded to do so by TM or operators.

Lifecycle management

Element Role:

1. Support status management for upgrade of software/hardware.
2. Support remote upgrade of software and firmware (at least for DSH LMC and LFAA LMC) and support hardware upgrades.
3. Support remote restart of software/firmware components.
4. Support remote power down of the Element and individual components.
5. Support remote power on, power down, restart of the Element if required.
6. Report LRU serial numbers, as well as software, firmware and hardware versions.