# SKA Observing Control View

## Introduction

This document is an attempt to propose an architecture which marries the top level observation management view of the SKA, with the hierarchical Tango control system that incorporates the LMC systems provides by the consortia. The former discusses observations in terms of Scheduling Blocks and Scans, and the latter in terms of Tango commands and attributes. This document aims to propose a simple, flexible interface based on a common state machine used by all systems. The interface is narrow, which means it should be fairly easy to understand and diagnose. The complexity is in sub-system (LMC level) configurations that are basically structured documents (JSON or XML) that are derived from the observation definitions created by observers. The intention is that TM receives these configurations from Observation Management and passes them with relatively little interpretation to the underlying sub-systems.

The document is based on JAC systems and some discussions in Pune in the TM face to face, with modifications based on meetings and discussions since then.

## Assumptions

The fundamental assumption in this document is that SKA data taking will be discontinuous. Observations will be described by scheduling blocks, which are made up of scans, during which the science data is taken. This is the design proposed by Observation Management and the Operational Concepts document and it ensures that data isn't taken when the system isn't in a well defined configuration specified by the observation management system.

However, this is clearly a design decision and the clear alternative is to take data continuously and flag data that should be dropped by a downstream system. The design doesn't preclude this, but the assumption is that the data is dropped relatively close to source.

There is also some variations in the way some key terms are described. My interpretations of these terms are below.

### Scheduling Block

This is currently defined in the SKA Glossary as: "Scheduling Blocks are the indivisible executable units of a project and contains all information necessary to execute a single observation. A scheduling block may be stopped and cancelled but not paused and resumed."

After discussion with the SKA Operations team, I believe that the last phrase “not paused and resumed” in this context is better thought of “cannot be interrupted by another scheduling block and continued at a later time”. However, it can be paused so some engineering adjustment can be made and then continued or cancelled.

### Scan

This is not defined in the SKA Glossary. However, the Operational Concepts Document effectively defines it as “an atomic unit of execution during which the system configuration is fixed”.

Again, after discussing this with the operations team, I would prefer this to say that it is “an atomic unit of execution during which data taking is normally continuous, but data taking can be briefly paused for an operational reason”.

### Sub-array

The SKA1-Mid Subarrays Resolution Team Report defines this as “a grouping which isolates a set of resources to be controlled together to achieve a common end.” I feel this is a good definition, but in the related notes they say that “Generally, receptors in the same subarray will have the same (intended) pointing and delay centers”. They then go on and describe sub-arrays as not being the schedulable entities, but instead defining “resource pools” as being those entities. I feel that this changes the whole definition of sub-arrays and introduces a level of complexity that is not needed. The complexity of sub-arrays is not controlling the system to do a diverse set of operations, it is in the complexity of breaking the system into pieces and scheduling the pieces together. Hence, in this report sub-arrays revert to the earlier definition in which they are a scheduling concept that can only be changed at scheduling block boundaries. The system flexibility required by the sub-array report can, I think, be handled by the configuration concept, where individual parts of the sub-array are configured independently.

Both SKA telescopes will be controlled as a number of independent sub-arrays, and since they are independent, it is almost meaningless to discuss the state of the telescope as a whole - the telescope will be a union of the sub-arrays. However, it is likely that the telescope may frequently be configured so that the vast majority of dishes are in a single sub-array, with the remaining few being in a small engineering array, so it is understandable that the two concepts (Telescope state vs sub-array state) may get confused. However, the rest of this document describes a system for controlling a single sub-array - there is assumed to be a parallel system for each sub-array.

# Scope

The scope of the control system described in this document is only the top level components that participate in the observation down to, and including the top level control tasks in the LMCs. “Participate in the observation” means that they handling the real-time dataflow and are expected to be operate together to within a relatively small synchronisation. This excludes any components that just are monitoring systems (e.g. weather) or process the data after significant delays (e.g. the SDP image pipeline).

I see the control of an SKA MID sub-array during operations in terms of the following diagram.

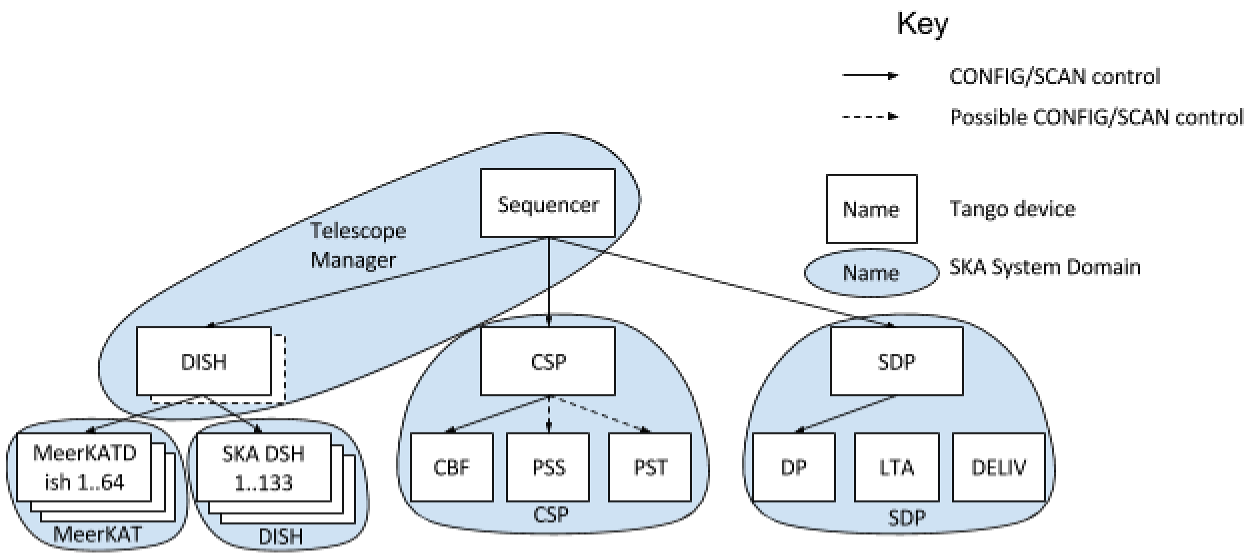
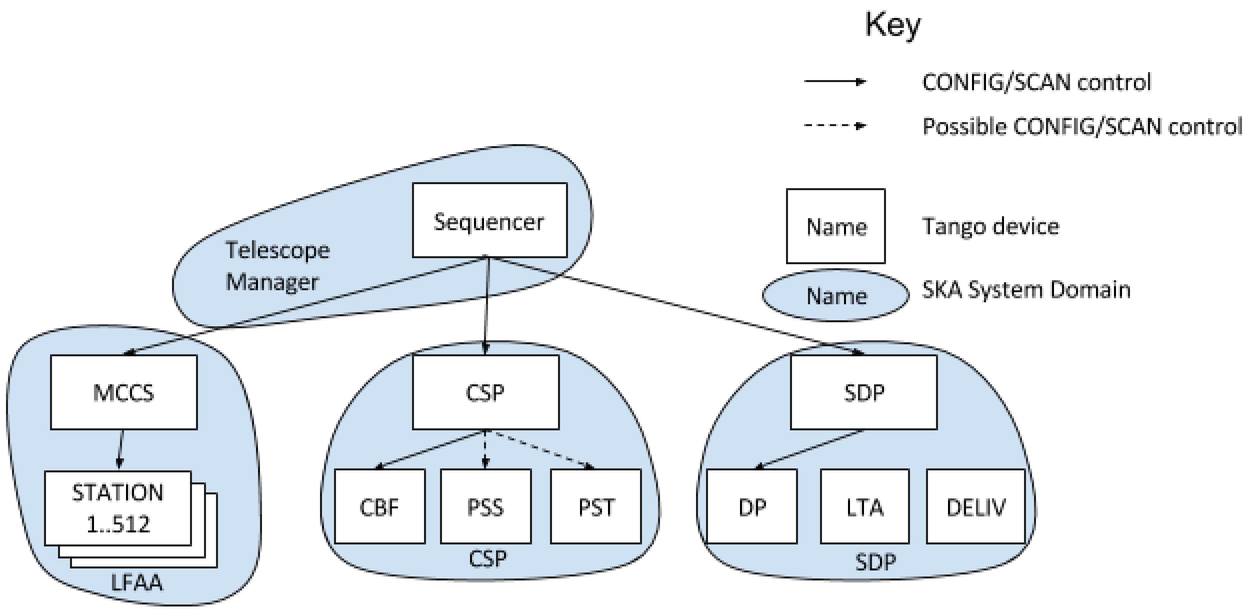


*Figure 1: TANGO devices for which this document is relevant. This structure is replicated each sub-array*

… and the control of an SKA LOW sub-array similarly:



INFRA, SaDT and parts of MeerKAT have been omitted from this diagram because they are not commanded by the TM sequencer[1]. These, and some SDP systems (e.g. LTA and DELIV) are passively involved in the observations, but not actively - the only effect they will have is if there is a major fault which causes the sub-array to transition to a non-operational state.

The CONFIG/SCAN control derives from the top level observational view of the system defined by OBSMGT. In implementing this, I propose it will require each TANGO device below the Sequencer device in the above diagram to implement a uniform control state machine mechanism described below. Since it is a hierarchy, intermediate systems are both a controlled sub-system for a higher level device, and a controlling system for lower level devices in the hierarchy. I will try and ensure this context is clear below.

Note that since all observation control will be done at the sub-array level, it is assumed that the hierarchy of the controlled sub-systems distinguishes between sub-array contexts. I assume the design pattern for this is to model the control of a sub-array as a TANGO class, and implement the control in different devices for each sub-array. Whether the devices for all the sub-arrays are in one TANGO device server (i.e. one process) or in multiple device servers may be implementation dependent. However, for the purposes of the exercise, the above diagram (and all the discussion below), should be assumed to be repeated once for every sub-array, and there be an implementation defined fixed number of sub-arrays (currently 16) and some (actually most) sub-arrays contain no controlled hardware.

[1] Unless we adopt some form of software defined networking (SDN) for control of SaDT components during the observation, but at the moment any form of SDN is confined to CSP or SDP.

# Description of Observing Control

## Observing Control State Diagram

Each TANGO device that abides by the CONFIG/SCAN protocol will have an enumerated attribute called OBS_STATE.

| ***Attribute Name*** | ***Attribute Type*** | ***Description*** | ***Access*** |
|---|---|---|---|
| OBS_STATE | Enum | One of Idle, Configuring, Ready, Scanning, Paused, Aborted or Fault | Read/Write |

The observing state diagram is described below. Specific transitions of OBS_STATE can be triggered either internally or externally (but not both). The states and state transitions are described in the next two sections.

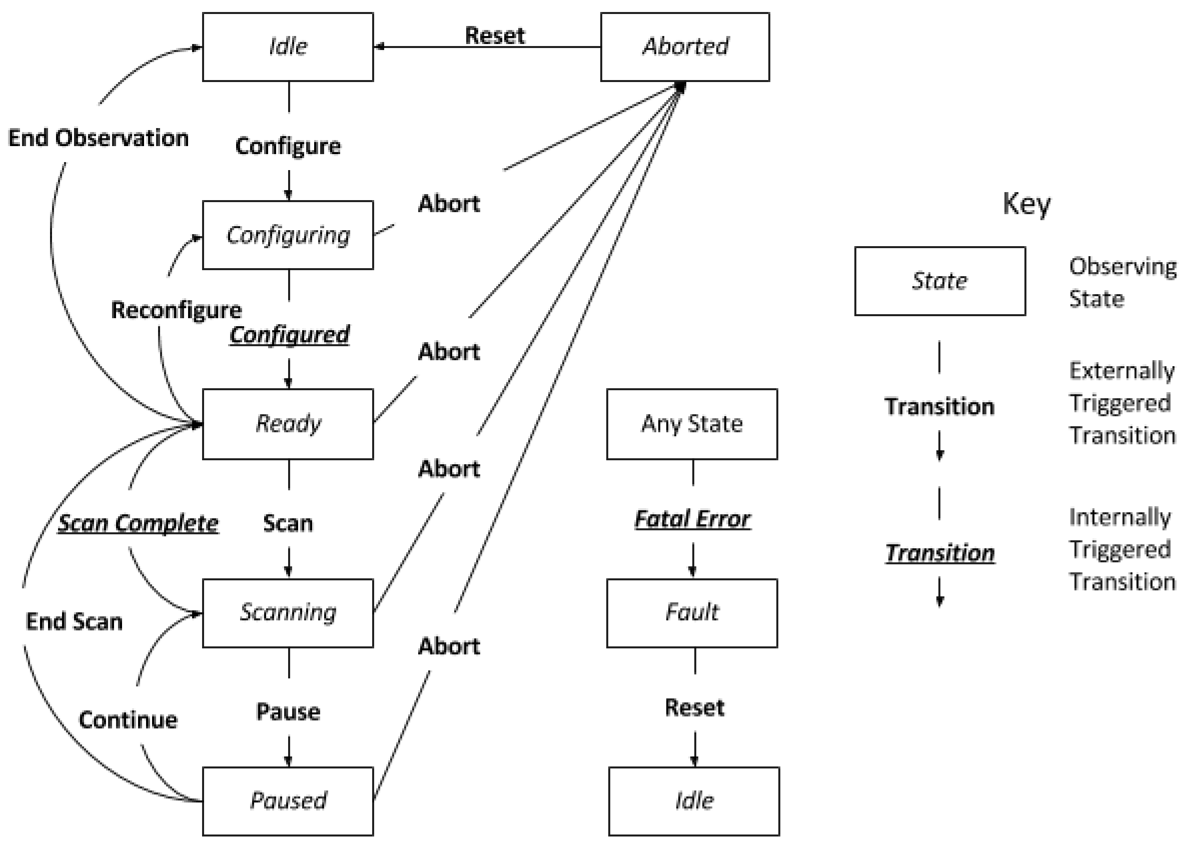


*Figure 2: Observing Control state transition diagram*

## Description of Observing Control States

| ***State*** | ***Description*** |
|---|---|
| *Idle* | Sub-array is ready to observe, but is in an undefined configuration. |
| *Configuring* | System is being prepared for a specific scan. On entry to the state no assumptions can be made about the previous conditions. It is a transient state and will automatically transition to *Ready* when it completes normally. |
| *Ready* | System 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). |
| *Scanning* | System is taking data and, if needed, and all components are synchronously moving in the observed coordinate system. Any changes to the sub-systems are happening automatically |
| *Paused* | System is fully prepared for the next observation, but not actually taking data or |

| | |
|---|---|
| | moving in the observed system. Similar to *Ready* state. |
| *Aborted* | System has had previous state interrupted by controller and is in an undefined state. |
| *Fault* | System has detected an internal error making it is impossible to remain in the previous state. |

## Description of Observing Control State Transitions

With the exception of “Reconfigure”, all externally triggered state transitions are generated by a TANGO command of the same name. The design intention is that the commands will be broadcast using the Tango “Group” mechanism, so all external transitions will be triggered across the system nearly simultaneously.

All commands will return a success or failure (which will typically be signified by throwing an exception in the client). If a sub-system command returns a failure then the controlling system will sends an “Abort” command to any controlled sub-system before itself returning a failure. (Limited retry’s are low-level responsibilities).

| ***Transition*** | ***Description*** |
|---|---|
| **Configure** | There are two possibilities, both of which should be handled:<br>1. Command has a single integer parameter, which is a Scheduling Block Identifier (SBI).<br>2. Command has a single string parameter, which is a JSON string containing the configuration information. One of the items in the JSON string will be the SBI. The JSON structure will be organised hierarchically, and the system should ignore any elements that are not relevant to it.<br><br>On receipt of this command the controlled system will:<br>1. Update OBS_STATE to *Configuring*<br>2. Fan out the **Configure** command to its controlled sub-systems, including any sub-system configuration derived if the parameter was a JSON string.<br>3. Do any processing required to prepare for the start of the Scheduling Block.<br>4. Wait for all sub-systems to return.<br>5. Update any attributes required for synchronised start of the scan.<br>6. If all completes successfully, update OBS_STATE to *Ready* before returning successfully to the caller (a **Configured** transition). |
| ***Configured*** | This is an automatic transition after *Configuring* state has completed |

|  |  |
|---|---|
|  | successfully. |
| **Reconfigure** | This is the only transition not triggered by a TANGO command. It is triggered by a TANGO write_attributes() request to change a set of configuration attributes exposed by the sub-system.<br><br>On receipt of this request the sub-system will:<br>1. Update OBS_STATE to *Configuring*<br>2. Fan out the the transition in the form of write_attributes commands to its controlled sub-systems.<br>3. Do any processing required to prepare for the start of the Scheduling Block.<br>4. Wait for all sub-systems to return.<br>5. Update any attributes required for synchronised start of the scan.<br>6. If all completes successfully, update OBS_STATE to *Ready* before returning successfully to the caller (a **Configured** transition). |
| **Scan** | Command has a parameter which indicates the time (TAI) at which the Scan will start. |
| **End Observation** | Command has no parameter. This is a transition to indicate the end of the Scheduling Block. Sub-systems should transition to *Idle*, and may enter low power mode after a suitable time. |
| ***Scan Complete*** | This can be either an automatic or an externally triggered transition after the *Scanning* state is completes normally. Typically, at top level only one sub-system will automatically transition. The control system will then notify all the other sub-systems that the scan is complete by invoking this command. |
| **Pause** | Command has a single parameter which indicates the time (TAI) at which the scan should pause. As part of the transition the system should move to a position where the observation could resume from where it would be at that time. Pauses will not be triggered automatically, but will have to ultimately triggered by an operator action. |
| **Continue** | Command has a parameter which indicates the time (TAI) at which the Scan will restart. |
| **End Scan** | This command signifies that the scan has completed successfully, but manually by operator action. Otherwise it is synonymous to ***Scan Complete*** |
| **Abort** | Command has a single parameter and this is an asynchronous command that can be sent at any time. System should immediately transition to the *Aborted* state which is one in which the system is in a safe configuration and not generating observational data. The parameter (which could be optional) could indicate what level of safety is required - the two options being: |

| | |
|---|---|
| | 1. as near as possible to the current configuration (default), or<br>2. as safe as possible given the current conditions. |
| **Reset** | Command has no parameter. System should transition to *Idle* state. |
| ***Fatal Error*** | This is an internally triggered transition which indicates the system cannot continue with the current state transition, request or command. If a controlling system detects this transition it should immediately send an Abort transition to all other controlled sub-systems and transition to the Error state. |

## Engineering Functionality

Most of the detail in this document deals with control of typical observations where operation is automatic and controlled by a sub-array sequencer. In this case control will nearly always follow the hierarchy shown in Figure 1. The only exception might be:

1. The sequencer might not automatically transition from one state to another - it may be programmed to pause just before a scan, for example, so the operator can give a final check before taking data.
2. The Pause state is only entered as a result of a manual operation. At this point the operator might correct some minor setting before continuing.

However, during Engineering operations the control will largely not be automatic, and no such assumptions apply. In particular, the following should be supported:

- All transitions can be done manually - a typical implementation would be a button that sends the appropriate command to the sequencer, where it would be fanned out to the sub-systems.
- Automatic generation of a Scheduling Block Identifier. When the engineer initiates the **Configure** transition she is basically switching the system out of standby mode ready for operations. This might involve pressing a “Ready” button, which will trigger the sequencer to configure the system with a generated identifier.
- Any TANGO device can be manipulated directly at any time. Engineering will often be done via engineering screens that communicate directly to low-level TANGO devices. Nothing should prevent this low level control. Typically this will be done when the system is in *Ready* state.

## Correlation with other global views and similar concepts

The table below tries to correlate the Observing Control State of a sub-array with various other concepts.

| ***Sub-Array Availability*** | ***Operational State*** | ***User perception(?)*** | ***Accounting Category*** | ***Observing Control State*** | ***TANGO State?*** |
|---|---|---|---|---|---|
| Operational | Science Operations | Science observation | Setup | Configuring | MOVING |
| | | | | Ready | ON |
| | | | Science | Scanning | RUNNING |
| | | Calibration observation | Setup | Configuring | MOVING |
| | | | | Ready | ON |
| | | | Calibration | Scanning | RUNNING |
| | | Observation overheads | Other(?) | Idle | STANDBY |
| | | | | Paused | ON |
| | | | | Aborted | STANDBY |
| | System Level Engineering | Engineering | Engineering | Configuring | MOVING |
| | | | | Ready | ON |
| | | | | Scanning | RUNNING |
| | | | | Idle | STANDBY |
| | | | | Paused | ON |
| | | | | Aborted | STANDBY |
| Not Operational | Sub-array reconfiguration | Engineering | Engineering | Undefined | UNKNOWN |
| | Sub-array with no hardware | Irrelevant concept(?) | None/Other | Undefined | UNKNOWN |
| | Weather | Weather | Weather | Idle | Any |
| | Utility | Fault | Fault | Idle | OFF |
| | Component Level Engineering | Engineering | Engineering | Idle | Any |
| | System Fault | Fault | Fault | Fault | FAULT |

### Interaction with the TANGO Device State

Note that the TANGO state column in the above table is just a (potentially provocative) suggestion. The assignments may depend on technical details, such as the alarm behavior in MOVING, RUNNING and STANDBY and FAULT states. It is also assumed that the device automatically transitions through the TANGO INIT state to the *Idle*/STANDBY state.

## Synchronisation

This could be implemented in a number of ways. A simple proposal would be for each system to have an attribute that is set before the completion of any “Configured” or “Paused” transition. This would indicate a minimum time needed to start scanning. For example:

| ***Attribute Name*** | ***Attribute Type*** | ***Description*** | ***Access*** |
|---|---|---|---|
| SCAN_DELAY | Double | Minimum time in seconds for sub-system to transition from current state to scanning. Only defined when OBS_STATE is Paused or Ready. Negative values are undefined. | Read only |

Any controlling system will return the maximum value returned from any controlled sub-system. The sequencer will use this as a guide to set the start time for the next scan.

# Aggregation

Aggregation has been a topic of much discussion, since the right thing to do when aggregating is often context sensitive.

## Control Aggregation

The proposed control aggregation strategy is defined above - any low level fault that stops an observation continuing should be propagated to top level and all other controlled sub-systems aborted. This however, means that the barrier to declare a fault condition is high.

The use of this control aggregation strategy is only mandated to the levels described in the first (Overview) diagram, but individual elements could propagate it further if they see it is warranted.

### Monitoring Aggregation

Any form of monitoring aggregation will not have any direct effect on an observation in progress - this will only happen as a result of a change in the observing state. However, developers should generate summary attributes and summary screens of their sub-systems showing the important information.

Problems with the underlying sub-systems that need the attention of the operator should be signalled by the alarm system. Alarms should conform to the IEC 62682 standard. Controlling systems should propagate any alarms in the controlled sub-systems through a suitable alarm aggregation TANGO device. This should, at minimum, indicate whether there are any sub-system alarms, and whether there are any unacknowledged sub-system alarms.

TODO: This could either just be through a 3 level enum (NO_ALARMS, ACKNOWLEDGED_ALARMS, UNACKNOWLEGED_ALARMS) or something more complicated.

## Component level states

In this context, I consider that components are low level systems which are either:

- beneath the control hierarchy described above, or
- don’t participate in it (i.e. pure monitoring systems like weather stations or SaDT).

The current “SKA Control Model” document describes many potential component level states - the difference between that document and this is that it tries to define how to aggregate these component level states and I propose that they are largely not aggregated - except to the level described above. The aggregation is largely through the monitoring and display hierarchy, not the control hierarchy described above.

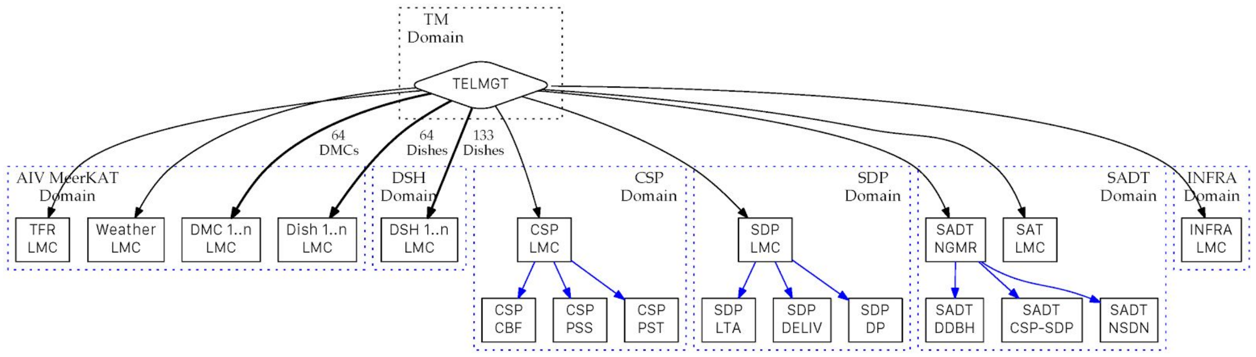
However, elements can do their own Engineering Control aggregation to the level that they consider appropriate, but that this does not not propagate up into Telescope Manager. Engineering level functionality can be done through engineering level devices.

Note that it may also make sense to aggregate alarms both by sub-array, and by element. (i.e. there is a top-level DISH alarm aggregation for all dishes with its own drill down, and an aggregation by sub-array).

# Appendix. Background Information

## SKA MID Control Hierarchy

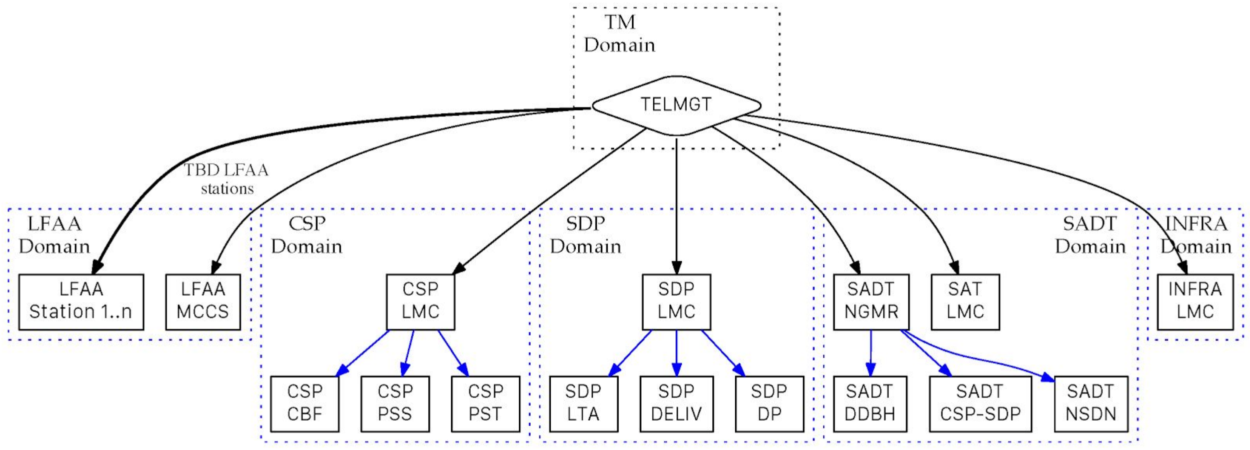
The SKA MID Control Hierarchy from a System Engineering domain point of view is represented as follows:



It is important to note that these boxes don't really represent anything specific yet from a control functionality viewpoint, they are just names. Also, the following top level states were defined in a draft document circulated by the MID States and Modes RT.

## SKA Low Control Hierarchy

The SKA LOW Control Hierarchy from a System Engineering domain point of view is represented as follows:



## SKA Observatory of sub-array states

The following state set of state definitions were used as input to the States and Modes resolution team.

| ***State*** | | ***Definition*** |
|---|---|---|
| Operational | Observing | At least one sub-array is in the SO_Calibrating or SO_Observing mode (see sub-array modes for definitions), for the purpose of science observations. |
| | Standby | System is functionally available for science observations, but is not used for observing. Sub-arrays could be in SO_Standby or SO_Configuring mode, or no sub-arrays could be configured. |
| Not Operational | Weather | Observing is not possible due to poor weather (e.g. wind stow conditions). |
| | Utility | Observing is not possible due to utility problems (e.g. power failure). |
| | Engineering / Maintenance | System is not available for any science observations, due to upgrades, scheduled maintenance, off-line calibration, software updates or testing. |
| | System Fault | System is not available for any science observations, due to a critical system fault. |