LMC harmonization through Telescopes
III Meeting - Edinburgh

CSP.LMC Prototype Development: Progresses and Proposals

C.Baffa, E.Giani, M.Vela Nuñez

Edinburgh 4-6 July 2016

Updates on Tango Prototype

Most progresses in Engineering UI (in Marina's presentation)

Some work on harmonization => Some of the results from ANT in the process of being incorporated

Some experimenting

<u>Changes from Madrid underlined</u>

Basic Assumptions & Requirements

1) Use of Tango as control framework

2) Most, if not all, TM interactions flow through CSP.LMC

3) TM is agnostic about the detailed hardware structure

4) TM sends coherent and complete commands to CSP.LMC

- CSP.LMC performs syntactic and minimum safety checks

5) TM sends detailed configurations for scan programming (EICD) as compounded settings for parameters or compounded commands:

6) TM sends immediate and timed commands.

- the implementation of command queues on all Master nodes
- Each command identified by an ID
- A structured system of acknowledge for immediate and delayed commands

Inside our Tango Classes

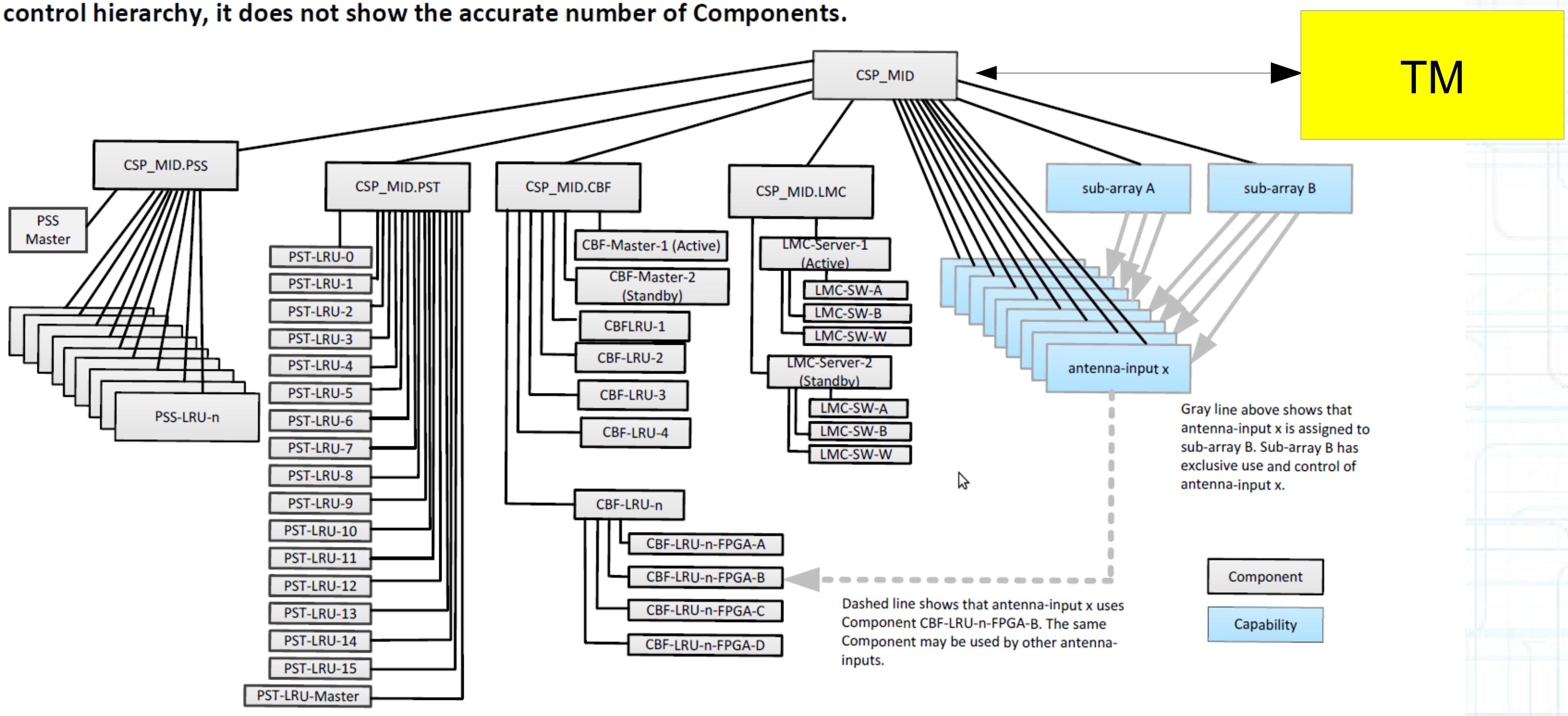
To define the Tango Classes of the CSP.LMC prototype we started from the two ICD documents:

- from the EICD we have derived the attributes and methods common to all elements, sub-elements and capabilities => we have defined few *abstract* classes
- from the IICD we have derived the attributes and methods specific to each sub-element and capability

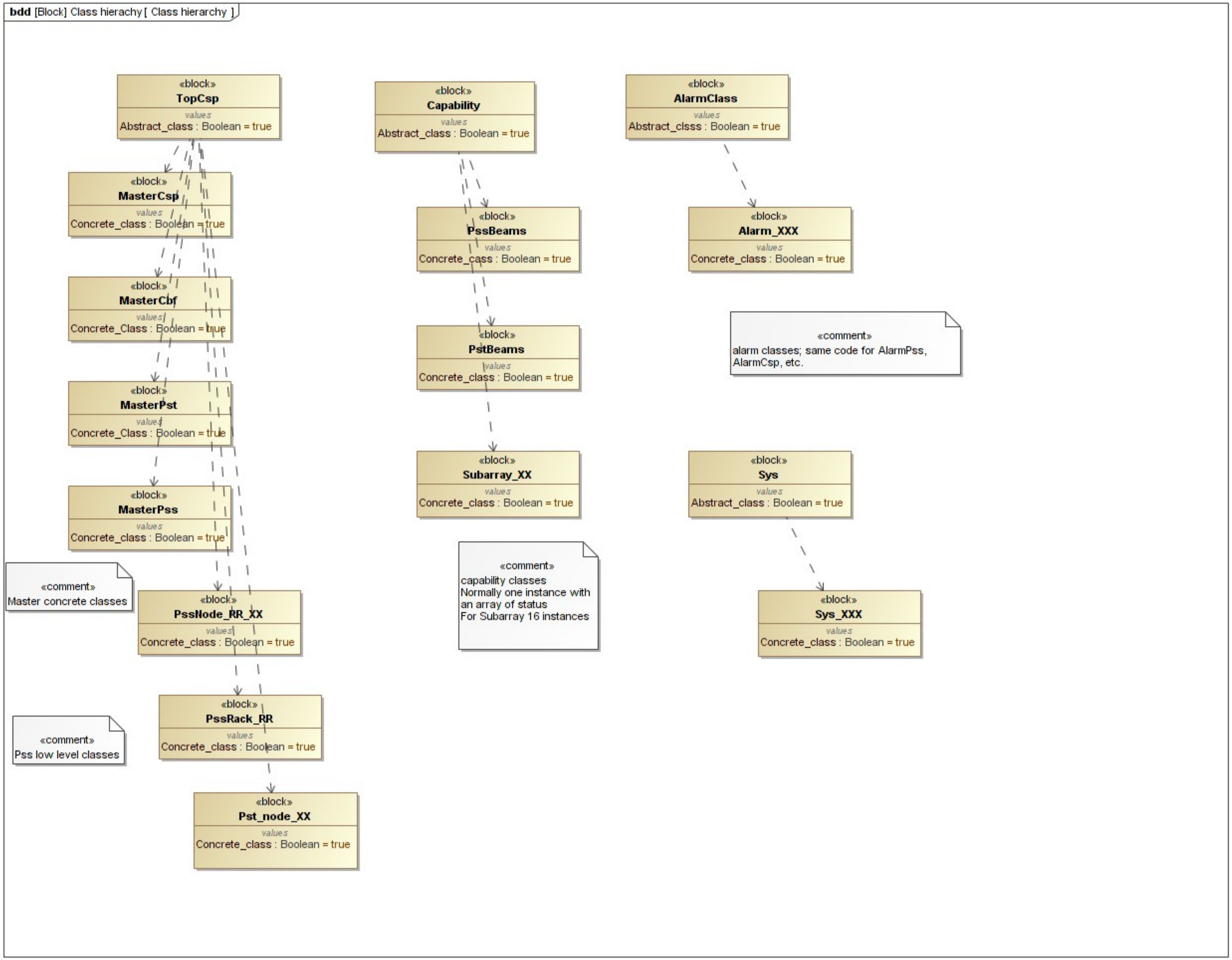
CSP Detailed Structure

From S.Vrcjc SKA ICD SKA Document

Figure 7-1 Monitor and control hierarchy for CSP Mid– this diagramhas been provided as an example of the monitor and control hierarchy, it does not show the accurate number of Components.



Taxonomy of the prototype classes



The prototype structure -1

- The prototype structure is modeled on the CSP architecture: Each M&C entity is implemented as a Tango Device Servers running one or more Tango Devices

  - ✓ One Tango Device Server for CSP Element
  - ✓ One Tango Device Server for each SubElement (CBF, PSS and PST)
  - ✓ Each Device Server runs on a separate PC (Master Node)

- The prototype will implement some M&C functionalities as Tango Devices.
- <u>The prototype uses the Tango-ANT System Logging</u>

We see two alternative approaches to sub-array implementation:

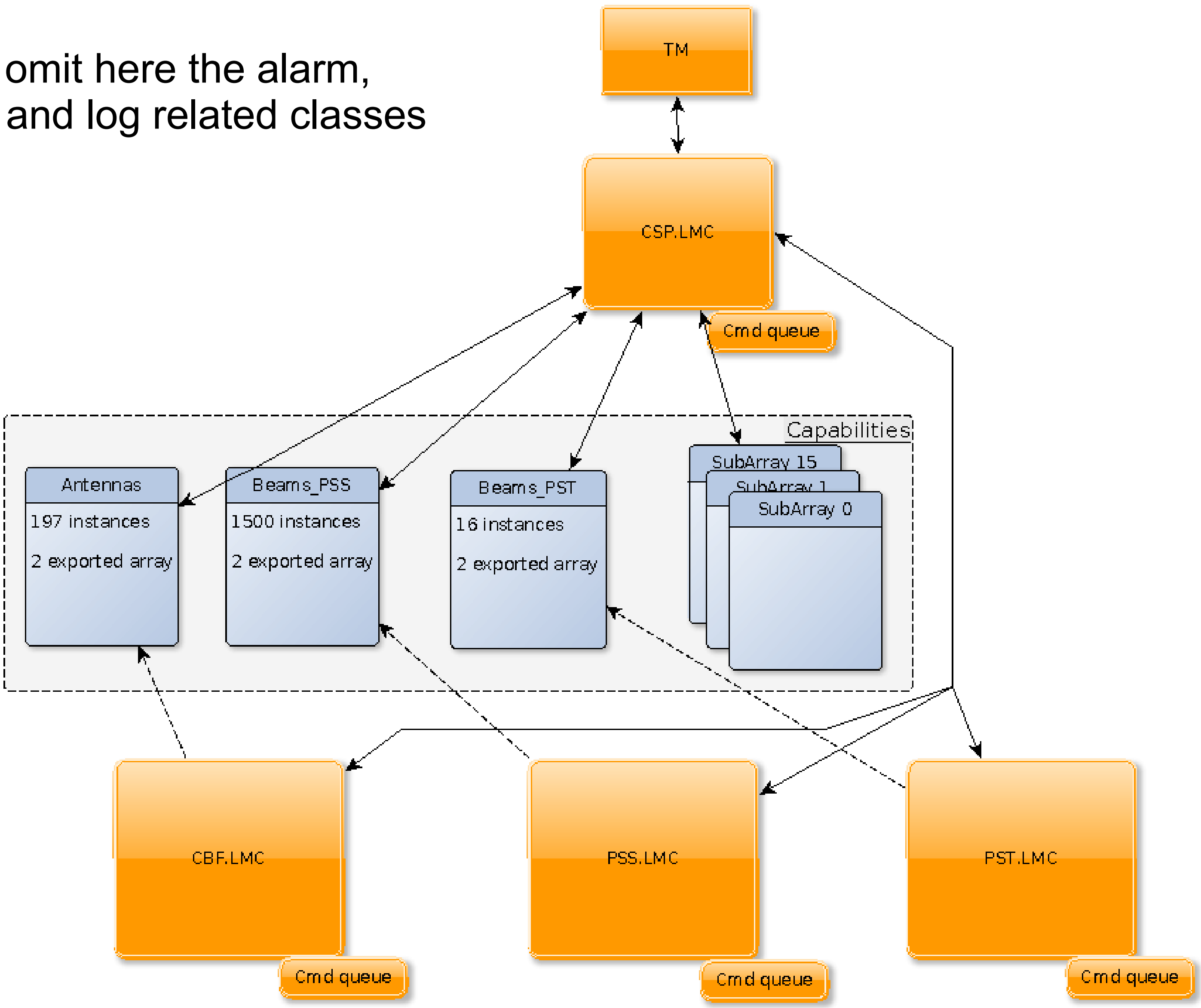
a) Implementation of subarrays as 16 separate telescopes which share a pool of hardware resources

b) A single hardware pool which can be organized in up to 16 subarrays.

→ In our prototype we have implemented model b).

The prototype structure - 2

We omit here the alarm,
sys and log related classes



The prototype structure - 3

- There will be one top-level CSP and three sub-element Master Nodes: 4 pc based on COTS hardware and SO (Linux)
- In each master node will run one or more Tango Devices servers

On the CSP Master node:

- The CSP.LMC Tango Device
- The CSP.SYS Tango Device
- A command handler device (scheduler)
- An Alarm Handler Device
- Formulaconf Device
- The Capability Device(s)
- The SubArray Device(s)
- The element logging system

The prototype structure -4

- On the three SubElement Master nodes:

  The SubElement LMC (CBF.LMC, PSS.LMC, PST.LMC)

  The SubElement SYS

  The Alarm Handler

  A command handler device (scheduler)

  A limited device logging system

  These devices can run in a single Tango Device Server (as a multi-class device) or can run in separate Tango Device Servers.

  → Alternative: single server (Box) or independent servers?

  We still use Box servers – alternative not tested

Arrays of State/Mode Attibutes

We need to use, report and summarize many SKA Status variables

For a complex physical device as CBF, PSS and PST we need to report the logical states of a large number of devices, from tenths to thousands

Most efficient handling by means of arrays

Tango still do not implement arrays of enum, → SKA Status Variables will be implemented as array of shorts

Prototype main points

- Tentative Naming Schema
- State/Mode Variables
- **Parameter setting, *setParam***
- Capability/SubArray strategy
- Scenarios execution analysis
- Alarm implementation
- Initialization strategy

Parameters Setting 1

At startup:

Succession of defaults:
1. Tango library default
2. Class default
3. Tango database
4. Hardwired code

At set-up:
Use of **setParam** command?
A number of alternatives

For special/engineering purpose a remote setAttribute(s) command

Parameters Setting 2

In a hierarchy of device server how to set up many parameters up to few levels below?

A number of alternatives:

1) For special/engineering purpose a remote setAttribute(s) command
2) For few parameters (up to 10? 50?) or forwarded or replicated attributes
3) For many parameters (from 100 to over 1000) set-up by means of:
   a) Blob transmission by pipe
   b) Parameters blob transmission by setParam command
   c) Blob transmission by future Tango REST interface

SetParam Command 1

setParam accepts attribute settings and general commands

setParam can have a complex command structure inside
It its simplest form *setParam* command just set different attributes to specific values In this case :

```
Command:  setParam   From: TM     Destination: CSP.LMC (cspMaster).
Argument: Json  String {
{
    "scanTime":  "34.12" , // scan (integration) time
    "beamBw":  "2",
    "accelerationRange" : "0",
    "DispersionMeasure":  "300",...
}
```

SetParam Command 2

In a more complex implementation, the attributes values can be specified also for lower level devices:

```
Command:  setParam   From: TM     Destination: CSP.LMC (cspMaster).
Argument: Json  String {
"CSP" : {
   "scanTime":  "34.12" , // scan (integration) time
   "beamBw":  "2", …
}
"PSS" : {
   "accelerationRange" : "0",
   "DispersionMeasure":  "300",...
}
```

Json argument versus structured Pipe:
efficiency, flexibility, easier to maintain

SetParam Command 3

In the most complete implementation, attribute setting can be mixed with command execution on a hierarchy of devices. setParam argument for a a complete beams PSS parameter set-up. We wrote a simulator to produce such strings.

```
Command:  setParam           From: TM  Destination: CSP.LMC (cspMaster).
Argument: Json  String {
"activationTime": "10:31:00", // should be a Unix time
"sourceId": "TM",
"commandId" : "123456",        // identifies this execution
"GlobalValues": { // init of internal variables common to all subsystems
          "subArrayId": "4",
          "ObservingMode": "2", // PSS
          "scanId":  "AB45-34",          // We store scanId for subArray 4
          "numberOfBeams": "500"
}
"CSP" : {
          // CSP specific parameters
          "PSSBeamID" : ["AB45-34/1", "AB45-34/2", … "AB45-34/500"] // 500 values
          "PSSPointingCoord" : [ … ] // 500 values
          "PSSDestinationAddress" : ["10.1.1.1:4000", … "10.1.50.10:4000"] // 500 values
}
"CBF.Master": {
           "setSubArray":{    // specialized command
                    "scanTime":  "34.12" ,         // scan (integration) time
                    "subArrayObsMode":  "2", // PSS
                    "numberOfChannels":  "4096", // PSS
                    "beamBw":  "2", // PSS
                    "bitPerSample":  "8", // PSS
                    "Filter Banks Parameters" : { … ,}      // many hardware related parameters
                    "Delay Model Parameters" : { … },
                    "commandId" : "123456/2",     // identifies this execution
          }
          "setBeams":{        // specialized command
                    "numberOfChannels":  "4096",
                    "PSSBeamID" : ["AB45-34/1", "AB45-34/2", … "AB45-34/500"] // 500 values
                    "Beam Ponting Parameters" : { … },      // many hardware related parameters
                    "commandId" : "123456/3",     // identifies this execution
          }
}
"PSS.Master": {
           "setSubArray":{    // specialized command
                    "subArrayId":  "4", // in the  fourth slot we host subArray 4
                    "scanTime":  "34.12" ,         // scan (integration) time
                    "subArrayObsMode":  "2", // PSS
                    "beamBw":  "2",
                    "accelerationRange" : "0",
                    "DispersionMeasure":  "300",
                    "programming Parameters" : { … }        // many hardware related parameters
                    "commandId" : "123456/4",     // identifies this execution
          }
           "setBeams":{        // specialized command
                    "beamBw":  "2",
                    "accelerationRange" : "0",
                    "DispersionMeasure":  "300",
                    "PSSBeamID" : ["AB45-34/1", "AB45-34/2", … "AB45-34/500"] // 500 values
                    "programming Parameters" : { … }        // many hardware related parameters
                    "commandId" : "123456/5",     // identifies this execution
          }

}
}
```

SKA SQUARE KILOMETRE ARRAY CSP

CENTRAL SIGNAL PROCESSOR

PSS EICD Parameters

Table 9-2 Legend for Table 9-3

Mode	
All	All
PT	Pulsar Timing
DS	Dynamic Spectrum (proposed - TBC)
FT	Flow Through (proposed - TBC)
When is the parameter required (When):	
SCfg	Scan Configuration - to be received in advance of the 'activation time' specified in the message.
SStart	At the beginning of the scan.
Intg	To be received after each N-second integration.
MC	May change, i.e. may be updated during an observation (scan).
Implementation (support) Priority (P) :	
H	High
M	Medium
L	Low
C	Not known if required until calibration model of the telescope is defined.
D	To be defined (TBD)

Table 9-3 CSP_Mid.PST - Observing Mode Parameters

CSP_Mid.PST Parameter (Keyword)	Type	Units, valid values or range (all TBD)	Mode	When	Description	P
Activation Time	String	UTC	All	SCfg	Date & Time when to start PST re-configuration	H
PST Beam ID	TBD	TBD	All	SCfg	Identifier assigned by CSP_Mid.LMC (or TM) used to identify beam configuration. CSP_Mid.PST selects which PST server (node) to use for this this scan and PST-Beam ID, and provides 'mapping' from the PST Beam ID provided by TM to PST Node ID (Capability ID).	H
CSP-PST-Beam ID	TBD	TBD	All	SCfg	Identifier of the Capability CSP-PST-Beam to be used for this configuration. Note: It is better to use an abstract Capability ID than the identifier of the PST node (or server) used by the Capability. In the future, a single server may be able to process data from more than one beam.	H
Scan ID	64 bits	TBD	All	SCfg	64-bit Scan ID inserted in the CBF output data.	H

CSP_Mid.PST Parameter (Keyword)	Type	Units, valid values or range (all TBD)	Mode	When	Description	P
Type of observation to be performed (OBSMODE)	string	PULSAR_TIMING	All	SCfg	The observing mode that the given PST server should operate in. In the future, two more Observing Modes may be added (an ECP has been proposed to add Observing Modes): Dynamic Spectrum and Flow Through.	H
Observer ID (OBSERVER)	string		All	SCfg	Observer in charge of observations.	L
Project ID (PROJID)	string		All	SCfg	Project that the observations are for.	L
Pointing ID (PNT_ID)	string		All	SCfg	ID for sub-array pointing.	H
Sub-array ID (SUBARRAY_ID)	string		All	SCfg		H
Source (SRC_NAME)	string		All	SCfg	Name or ID of source	H
ITRF (ITRF)	double[3]	[m,m,m]	All	SCfg	ITRF coordinates of the telescope delay centre.	H
Receiver ID (FRONTEND)	string		All	SCfg	Receiver name or ID (instrument).	H
Number of polarization channels (NRCVR)	int	[1, 2]	All	SCfg		H
Native polarization of feed (FD_POLN)	string	[LIN, CIRC]	All	SCfg		H
Feed handedness (FD_HAND)	int	[-1,+1]	All	SCfg	Code for sense of feed. FD_HAND = +1 for XYZ forming RH set with Z in the direction of propagation. Looking up into the feed of a prime-focus receiver (or at the sky), for FD_HAND = +1, the rotation from A (or X) to B (or Y) is counter clockwise or in the direction of increasing Feed Angle (FA) or Position Angle (PA). For circular feeds, FD_HAND = +1 for IEEE LCP on the A (or X) probe. See van Straten et al. (2010), PASA, 27, 104 for a full description of the polarisation conventions adopted for PSRFITS and PSRCHIVE RD14 AND RD15.	H
Feed angle (FD_SANG)	double	Degrees [-180,180]	All	SCfg	Feed angle of the E-vector for an equal in-phase response from the A(X) and B(Y) probes, measured in the direction of increasing FA or PA (clockwise when looking down on a prime focus receiver) and in the range +/- 180 deg.	H

CSP_Mid.PST Parameter (Keyword)	Type	Units, valid values or range (all TBD)	Mode	When	Description	P
Oversampling ratio (OVRSAMP)	int[2]		All	SCfg	Numerator and denominator for the oversampling ratio.	H
Beam major axis (BMAJ)	double	Deg.	All	SCfg		L
Beam minor axis (BMIN)	double	Deg.	All	SCfg		L
Beam position angle (BPA)	double	Deg.	All	SCfg		L
Frame of coordinates (COOR_MD)	string	[GALACTIC, EQUATORIAL, ECLIPTIC]	All	SCfg	Frame of coordinates.	H
Coordinate epoch (EQUINOX)	double	JD or MJD	All	SCfg		H
STT_CRD1	string	hh:mm:ss.sss or ddd.ddd	All	SStart	X component of starting coordinates in COORD_MD frame.	H
STT_CRD2	string	hh:mm:ss.sss or ddd.ddd	All	SStart	Y component of starting coordinates in COORD_MD frame.	H

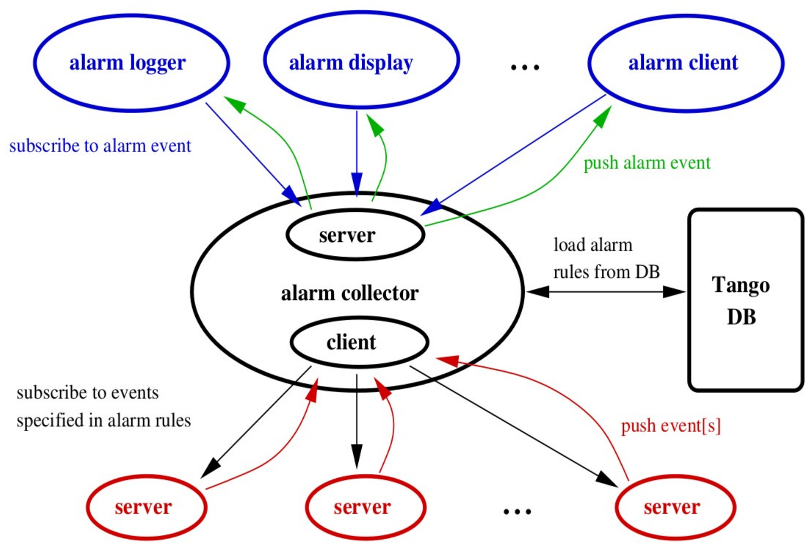
CSP_Mid.PST Parameter (Keyword)	Type	Units, valid values or range (all TBD)	Mode	When	Description	P
Track mode (TRK_MODE)	string	TRACK, SCANGC, SCANLA	All	SCfg	For 'TRACK' the beam axis tracks a fixed point on the sky; for 'SCANGC' the beam axis tracks at a uniform rate along a great circle on the sky; for 'SCANLAT' the beam axis tracks at a uniform rate along a line of constant latitude or declination (depending on COORD_MD).	H
STP_CRD1	string	hh:mm:ss.sss or ddd.ddd	All	Intg	X component of the final coordinates in COORD_MD frame. Required at the end of each sub-integration (not the end of each observation). Required for non-tracking observations only. This could be every 1 second to every hour, but would only be needed in the case of non-tracked scans. Currently there is no use case for non-tracked scans with the PST, so this is unlikely to be used.	H
STP_CRD2	string	hh:mm:ss.sss or ddd.ddd	All	Intg	Y component of the final coordinates in COORD_MD frame. Required at the end of each sub-integration (not the end of each observation). See STP_CRD1 for more comments.	H
De-dispersion measure (DM)	double	pc cm^-3 [0:10,000]	PT, DS	SCfg	De-dispersion measure for coherent/incoherent de-dispersion.	H
Rotation measure (RM)	double	rads m^-2 [-?:+?]		SCfg	Rotation measure for coherent de-faraday rotation. To de-faraday rotate low frequency signals while maintaining high time resolution may imply that we need to perform coherent rotation measure correction. This is not a requirement at the moment but may be implicitly derived from existing requirements - To Be Defined (TBD).	D
Maximum length of observation (SCANLEN_MAX)	float	Seconds [30:43200]	All	SCfg	Maximum length of observation.	H
Pulsar ephemeris (EPHEMERIS)	ASCII text		PT	SCfg	Pulsar ephemeris for pulsar being observed. The ephemeris files should be of the order a few kilobytes.	H
Pulsar phase predictor (PREDICTOR)	ASCII text		PT	SCfg	Pulsar phase predictor generated from ephemeris (T2 format). The predictor received from the SDP must have enough coefficients to represent the phase of the pulsar to better than 1 ns RMS over the full bandwidth of the scan and up to SCANLEN_MAX. More information related to predictors is provided below this table.	H
Output frequency channels (OUTNCHAN)	int	[1:2250]	PT, DS	SCfg	The number of output frequency channels.	H
Output phase bins (OUTNBIN)	int	[64:2048]	PT	SCfg	The number of phase bins in output.	H
The integration time for each output bin (OUTTBIN)	double	Seconds [0.00032:60]	DS	SCfg	The integration time for each output bin.	H

Prototype main points

- Tentative Naming Schema
- State/Mode Variables
- Parameter setting, *setParam*
- Capability/SubArray strategy
- Scenarios execution analysis
- **Alarms implementation**
- Initialization strategy

Alarms Handling

We plan to use Tango C++ Alarm System
From thousands to millions of attributes:
We definitively need both
a fast Implementation and *a hierarchical approach*



Alarms in Tango are out-of-limits exception.
<u>The Alarm device driver and Formulaconf device driver can convert this scenario to a complete Alarm System</u>

General Suggestions

- A centralized catalog with short **Executive Summaries** of main points (1-2 paragraphs) and and reference documents together with responsible people
- Easier access to other consortia documents
- SKA wide coding standards (LSST style)
- SKA wide code repository and politics

Comments and Suggestions?

Thank you!