AA0.5 Single-Station Commissioning Modes

D. C. Price et. al

Thu 5th Sept, 2024

1 Introduction

Single-station commissioning modes were developed for the Aperture Array Verification System (AAVS[1]) program, and have been demonstrated on the Engineering Development Array (EDA2) and AAVS3 systems. Single-station commissioning modes have, and will continue to be, crucial for commissioning, testing, and calibrating SKA-Low stations.

Notably, single stations can be used for all-sky snapshot imaging. In December 2019, the LFAA CDR[2] panel recommended adding all-sky imaging functionality for station diagnostics **[RD9, §6h]**, which were added in ECP-1900003 **[RD10]**. This all-sky imaging capability now underpins sensitivity analyses and calibration approaches used in MCCS[3].

While the underlying functionality for all-sky imaging and other single-station commissioning modes will exist in AA0.5 and beyond, access to single-station modes will only be available via MCCS **[RD10]**; there are no plans for support via higher-level OSO[4]-like tools, such as scripting interfaces or databases **[RD2]**.

Nevertheless, Science Operations will need to routinely conduct single-station commissioning observations for verification testing during station roll-out, to support System Science and Science Commissioning teams, and to continually build up a sophisticated understanding of station characteristics for exquisite calibration. This motivates effort to support, document and simplify single-station commissioning modes. Indeed, §4.10 of the AAVS3 Status Review **[RD8]** recommends that:

> *"A library of observing scripts with the functionality of the existing AAVS suite, but calling MCCS for station control and data acquisition, should be developed."*

This document has two goals, both aimed toward enacting this recommendation:

1. To synthesize existing but out-of-date documentation on single-station commissioning modes, and update it for MCCS-controlled AA0.5.

2. To define a YAML[5] specification for single-station operational modes to make commissioning observations easier, and as a first step toward the AAVS3 Status Review recommendation above.

These goals are expanded upon in the following subsections.

1.1 Goal 1: Update and improve documentation of single-station modes

The first goal is *to provide a reference document for telescope operators with sufficient detail to conduct a single-station commissioning observation with AA0.5.* To this end, Section 2 details the ten single-station commissioning modes that can be accessed via MCCS.

A single-station user guide for operators. Previous documentation for single-station observations was written for AAVS **[RD3,4]**, which uses a deprecated data acquisition system that is being replaced by MCCS. While the MCCS observing modes are equivalent, these documents are nonetheless out of date. There is also useful information about single-station operational modes in firmware capability and control documents, e.g. **[RD5]**, where a telescope operator may not think to look. Here, we have tried to extract information from these documents, update them to reflect their MCCS implementation, and synthesize the information into a 'user guide' for telescope operators.

Supporting MCCS documentation. While we have focused on the needs for telescope operators, parts of this document could help support MCCS API[6] documentation. For example, the `ska-low-mccs-spshw` Python package provides the utilities to configure single-station observing modes and capture data, but does not go into detail about what the observing modes are.

1.2 Goal 2: Define a schema that describes a single-station observation

The second goal is *to define a schema that describes single-station operational modes to make commissioning observations easier.* It is designed to be both human-readable and machine-readable. This specification will allow exact and explicit communication of desired observations between Science Operations and System Science / Science Commissioning teams.

Supporting data lifecycle management. It is important that we do not lose the contextual metadata about an observation that enables us to work with the recorded data files. Learning from our AAVS3 experience, Science Operations is developing a lightweight observation request and management tool using Jira, to track AA0.5 single-station commissioning observations. We intend that this tool is also used to generate/store standardized descriptions of all observations undertaken. The ADR-55 data archival and organization approach used as an "initial standard" for AAVS3 **[RD6]** could be continued with AA0.5, enriched with context metadata from these observation descriptions.

Toward a single-station commissioning mode tool suite. This specification is also is a first step toward a simple single-station equivalent of the OSO software tools **[RD2]**. Operating a single station requires far less orchestration than operating the full SKA-Low, and not all the functionality of the OSO tools are needed. Nevertheless, MCCS only provides a low-level interface to control single

stations, meaning there is a gap in system functionality for running and tracking single-station observations. Long-term, single-station modes support could be added to the OSO tool suite and made available to end users via an Engineering Change Proposal (ECP[7]); but in the short term, development of simple OSO-like tools for single-station commissioning modes is pressing.

The proposed solution is that an observer writes an observation config file (or uses a GUI to generate one), which is then passed to a scheduler program that executes the observation—essentially a single-station equivalent to to the OSO *Observation Scheduling Tool* (OST) and *Observation Execution Tool* (OET). The config files could be archived simply in a folder, or a database, as a lightweight single-station *Observatory Data Archive* (ODA).

1.3 Reference Documents

Ref	Document ID	Title
RD0	SKA-TEL-SKO-0001722	Observatory Establishment and Delivery Plan
RD1	SKA-TEL-SKO-0000002	SKA1 System Baseline Design V2 (BDV2)
RD2		OSO Roadmap
RD3	SKA-TEL-LFAA-060054	AAVS1 Software Demonstrator Design Report
RD4		AAVS1 Software User Guide
RD5	Comoretto (2021)	LFAA Tile Beamformer Structure
RD6		ADR-55 Definition of metadata
RD7	SKA-TEL-SKO-0001818	SKA Data Products: A summary
RD8	SKAO-TEL-0002425	AAVS3 Status Review: Panel Report
RD9	SKA-TEL-SKO-0001007	LFAA CDR Panel Report
RD10	ECP-190003	Station Diagnostics for LOW
RD11	ADR-55	Definition of metadata for data management at AA0.5
RD12	ADR-22	Decide approach to versioning JSON schema
RD13	ADR-35	Standardise JSON key names: convention and common keys
RD14	ADR-49	Standard units and some standard keynames for use in JSON
RD15	ADR-37	How to represent time
RD16	`ska-oso-pdm`	SKA Project Data Model (PDM)
RD17		OSO Project Data Model refactor

Information in this memo is based on the above reference documents (RDs).

Contents

Notes

[1] AAVS: Aperture Array Verification Systems
[2] LFAA CDR: Low Frequency Aperture Array Critical Design Review
[3] MCCS: Monitor, Control & Calibration System
[4] OSO: Observatory Science Operations
[5] YAML: Yet Another Markup Language
[6] API: Application Programming Interface
[7] ECP: Engineering Change Proposal

2 Observing Modes

This document defines ten 'single station' modes for commissioning observations (Table 1):

Table 1: Summary of single-station commissioning modes

ID	Mode	MCCS ID	
	ADC sample capture		
1	synchronous	`RAW_DATA`	§2.2.1
2	asynchronous	`RAW_DATA`	§2.2.2
*	transient buffer dump	`ANTENNA_BUFFER`	§2.2.3
	F-engine (coarse channel) voltages		
3	single fixed channel	`CONTINUOUS_CHANNEL_DATA`	§2.3.1
4	frequency sweep	`CHANNEL_DATA`	§2.3.2
	Correlator		
5	single fixed channel	`CORRELATOR_DATA`	§2.4.1
6	frequency sweep	`CORRELATOR_DATA`	§2.4.2
	Station beamformer		
7	voltage beam	`RAW_STATION_BEAM`	§2.5.1
8	power beam	`STATION_BEAM_DATA`	§2.5.2
	Bandpass monitoring		
9	antenna bandpass (PSD)	`INTEGRATED_CHANNEL_DATA`	§2.6.1

* *Not yet implemented*

The observing modes in the table correspond to receiver modes in MCCS that are derived from the `aavs-system` package[1] (see MCCS ID column). As we have grouped 'sub-modes' together, and to align with more common terminology, we do not use the MCCS IDs elsewhere within this document.

ADC sample capture

Primarily for engineering tests, this mode captures a few microseconds of data, direct from the ADCs. Using **synchronous** capture mode, 5.12μs of data may be captured. This is extended to 40.96μs if **asynchronous** mode is used, however this means antenna data streams are not time-aligned. The **transient buffer dump** mode will capture a much larger time period and is currently in development.

F-engine (coarse channel) capture

These modes capture voltage-level data from a coarse channel to disk. The SKA-Low signal processor implements an **F-engine**, which breaks the antenna data streams into 512 **coarse channels** (0.78125 MHz channel spacing). Currently, only a single channel can be captured at once. The channel can be automatically scanned across channels (**sweep**), or can be kept at a **fixed** frequency.

[1] https://gitlab.com/ska-telescope/aavs-system

Correlator modes

The correlator modes take the data stream from a coarse channel for all antennas, and forms the cross-correlation between all antenna pairs (i.e. the visibility matrix). Data are integrated for 0.283–2.265 s. As with the F-engine capture modes, the firmware can automatically **sweep** across channels, or output can be at a **fixed** frequency.

Station beamformer modes

SKA-Low stations use frequency-domain beamforming to combine antennas and form a phased-array beam. This mode can capture wide bandwidths (multiple channels), at the expense of higher data rates. The **voltage station beam** can be recorded to disk, or the **power station beam**, which is equivalent to a dynamic spectrum (data are squared and time-averaged to form power spectral densities, PSD), can be recorded instead.

Bandpass monitoring mode

Primarily for station health monitoring, this mode captures the **bandpasses** (measurements of power spectral density, PSD) for each antenna within the station. The bandpass covers 0–400 MHz and has 0.78125 MHz resolution (equal to the coarse channel bandwidth).

2.1 Nomenclature overview

An observation mode can be described by its observing parameters used to configure the observation, and the type of data that is produced:

Observing parameters: A number of **observing parameters** (Table 2) are set by physical design (e.g. the number of antennas in the array), or by firmware implementation (e.g. the number of coarse channels). There are also a number of tunable, or mode-specific parameters, which are summarized in Table 3. Tables with **blue backgrounds** list parameters that are fixed due to the station's physical design and/or the TPM firmware (see Appendix A.1). Tables with **pink backgrounds** list parameters that may have different values across modes.

Output data frames: Each observing mode is described in terms of its output data frame: an N-dimensional array (aka a 'tensor') whose **frame shape** is set by observing parameters. All data within a frame has the same **frame datatype**. There are two **frame modes**: some modes deliver only a single frame; other modes deliver a continuous stream of frames. In general, frames are contiguous in time over the observing period, but some modes (M3) allow frames to be skipped to throttle the data rate, and some modes (M4, M6) sweep through frequency channels one at a time.

Table 2: Observational parameters that are common across all modes. These are set by physical design and/or the TPM firmware.

Parameter		Value
Station / antenna		
N_{ant}	Number of antennas in array	256
N_{baseline}	Number of baselines (including autocorrelations)	32896
N_{pol}	Number of polarizations	2
N_{stokes}	Number of polarization products (XX, YY, XY, YX)	4
ADC		
τ_{sampling}	ADC sampling rate	1.25 ns
$\nu_{\text{ADC bw}}$	ADC digitized bandwidth, $1/2\tau_{\text{sampling}}$	400 MHz
F-engine		
N_{coarse}	Number of coarse channels	512
f_{pfb}	Filterbank oversampling factor	32 / 27
ν_{spacing}	Coarse channel spacing	.78125 MHz
ν_{cbw}	Coarse channel bandwidth, $\nu_{\text{spacing}} \times f_{\text{pfb}}$	.925925 MHz
τ_{coarse}	Coarse channel sample rate, $1/\nu_{\text{cbw}}$	1.08 μs

Table 3: A summary of tunable or mode-specific parameters used in this document.

Parameter	Description	Modes
Data array		
N_{frame}	Number of frames	all
N_{samples}	Number of samples to capture per frame	1–4
N_{int}	Number of time integration steps per frame	6
Channel selection		
N_{chan}	Number of channels to capture	3,5,7
N_{sweep}	Number of channels to capture in sweep modes	4,6
i_{chan}	Start channel ID**	3–7
Time		
t_{start}	Observation start time	all
t_{obs}	Observation length	all
t_{int}	Integration time, $N_{\text{int}} \times \tau_{\text{coarse}}$	5,6,8,9
N_{int}	Number of time samples to integrate	5,6,8,9
$t_{\text{frame period}}$	How often a frame is written to disk, in seconds	3

2.2 ADC Sample Capture

This mode captures a few microseconds of data, direct from the ADCs. Using synchronous capture mode, 5.12μs of data may be captured. This is extended to 40.96μs if asynchronous mode is used, however this means antenna data streams are not time-aligned. This observing mode is primarily for engineering and monitoring purposes.

Parameter		Value
τ_{sampling}	ADC sampling rate	1.25 ns
$\nu_{\text{ADC bw}}$	ADC digitized bandwidth	400 MHz

2.2.1 Synchronous capture (M1)

Captures 4096 ADC samples ($t_{\text{obs}} = 5.12\mu s$, 1.25 ns sampling period) from all antennas. Data are captured synchronously across antennas.

Parameter		Allowed values
N_{frame}	Number of frames	1
N_{samples}	Number of samples	4096

Frame shape: $(N_{\text{ant}}, N_{\text{pol}}, N_{\text{samples}}) = (256, 2, 4096)$
Frame type: 16-bit signed integer (12 bits occupied)
Frame size: 4 MiB per capture
Frame mode: single

2.2.2 Asynchronous capture (M2)

Captures 32768 ADC samples ($t_{\text{obs}} = 40.96\mu s$, 1.25 ns sampling period) from all antennas. Data are captured asynchronously across antennas, meaning that the data streams are not time aligned.

Parameter		Allowed values
N_{frame}	Number of frames	1
N_{samples}	Number of samples	32768

Frame shape: $(N_{\text{ant}}, N_{\text{pol}}, N_{\text{samples}}) = (256, 2, 32768)$
Frame type: 16-bit signed integer (12 bits occupied)
Frame size: 32 MiB per capture
Frame mode: single

2.2.3 Transient buffer dump (M2)

Large transient buffer for raw voltage data using RAM. This mode has been implemented in the TPM firmware, but has not been tested via MCCS.

v1.6 TPM hardware has 8 GB RAM per FPGA. For 800 MB/s per antenna (8-bit resample), that's 12.8 GB/s, so 8 GB would be 0.625 s buffer. v2.0.2 may have 32 GB, so potentially a few seconds.

2.3 F-engine voltage capture

These modes capture voltage-level data from a coarse channel to disk. The 512 'coarse' channels are spaced $400/512 = 0.78125$ MHz apart. The data are oversampled by a factor of 32/27, so each channel has a bandwidth of ≈ 0.9259 MHz (corresponding sample rate is $1/\nu_{\text{bw}} = 1.08\mu s$). These modes have a high data output rate, so are limited by disk write speed.

Parameter		Value
N_{coarse}	Number of coarse channels	512
ν_{bw}	Coarse channel bandwidth	$.925\overline{925}$ MHz
ν_{spacing}	Coarse channel spacing	.78125 MHz
τ_{coarse}	Coarse channel sample rate	1.08 μs

2.3.1 Fixed frequency (M3)

Captures complex voltages from a single coarse channel. Writes in terms of *frames*.

Parameter		Allowed values
N_{frame}	Number of frames	$1-\infty$
N_{samples}	Number of samples to capture per frame	524288*
N_{chan}	Number of channels to capture	1
i_{chan}	Start channel ID	0–511
$t_{\text{frame period}}$	How often a frame is written to disk, in seconds	0–120*

*Needs verifying

Frame shape: $(N_{\text{chan}}, N_{\text{ant}}, N_{\text{pol}}, N_{\text{samples}})$
Frame type: 8-bit complex signed integer
Frame size: 256 MiB per frame (assuming $N_{\text{samples}} = 524288$).
Frame mode: continuous stream — but not necessarily *contiguous* in time
Data rate: 452 MiB per second, per channel (if $t_{\text{frame period}} = \tau_{\text{coarse}}/N_{\text{samples}}$)

2.3.2 Frequency sweep (M4)

Step through N_{sweep} coarse channels, one at a time, and send N_{samples} for each channel. This mode was designed for calibrating the station.

Parameter		Allowed values
N_{samples}	Number of samples to capture	1024*
N_{frame}	Number of frames	1
N_{sweep}	Number of channels to capture	1–512
i_{chan}	Start channel ID	0–511

*Needs verifying

Frame shape: $(N_{\text{sweep}}, N_{\text{ant}}, N_{\text{pol}}, N_{\text{samples}})$
Frame type: 8-bit complex signed integer
Frame size: 32 MiB per sweep (assuming $N_{\text{sweep}} = 512$ channels, $N_{\text{samples}} = 128$).
Frame mode: single

2.4 Correlator modes

Correlator modes compute the complex visibilities $V_{pq} = \langle v_p(t) v_q^*(t) \rangle$ for a pair of antennas, p and q, where $\langle\rangle$ denotes time averaging. The correlator ingests the same data as the F-engine voltage capture modes in §2.3, but computes outputs time-integrated visibilities instead of voltages.

Baselines The visibility matrix V_{pq} is Hermitian ($V_{pq} = V_{qp}^*$), so the number of unique entries (corresponding to antenna baselines) given by

$$N_{\text{baselines}} = \underbrace{\frac{N_{\text{ant}}^2 - N_{\text{ant}}}{2}}_{\textit{cross correlations}} + \underbrace{N_{\text{ant}}}_{\textit{auto correlations}} \quad ; \tag{1}$$

for $N_{\text{ant}} = 256$, the number of baselines is $N_{\text{baselines}} = 32896$. As such, the correlator computes and outputs only the lower triangular matrix where p ≤ q.

Integration time The correlator integrates (time-averages) N_{int} samples together, resulting in integration times

$$t_{\text{int}} = N_{\text{int}} \times \tau_{\text{coarse}} \tag{2}$$

where $\tau_{\text{coarse}} = 1.08\,\mu$s. N_{int} values should be within 2^{18}–2^{22}, with corresponding integration times 0.283–2.265 s. Integrations times smaller than 0.283 s are not supported.

Parameter		Values
N_{baseline}	Number of baselines (including autocorrelations)	32896
N_{stokes}	Number of polarization products (XX, YY, XY, YX)	4

2.4.1 Fixed frequency (M5)

Continually records a time stream of correlated data frames ($N_{\text{frame}} \leq \infty$) from a single coarse channel.

Parameter		Allowed values
N_{frame}	Number of frames	1–∞
N_{chan}	Number of channels to capture	1
i_{chan}	Start channel ID	1–511
t_{int}	Corresponding integration time	0.283–2.265 s

Frame shape: $(N_{\text{chan}}, N_{\text{baselines}}, N_{\text{stokes}})$
Frame type: 32-bit complex float (8 bytes)
Frame size: 1.0 MiB per frame
Frame mode: continuous stream
Data rate: 0.43–3.55 MiB per second (for t_{int} in 0.283–2.265 s)

2.4.2 Frequency sweep (M6)

Step through N_{sweep} coarse channels, one at a time, and record time-averaged visibilities for each channel.

Parameter		Allowed values
N_{frame}	Number of frames	1
N_{int}	Number of integration timesteps	1 or 2
N_{sweep}	Number of channels to capture	1–512
i_{chan}	Start channel ID	0–511
t_{int}	Corresponding integration time	0.283–2.265 s

Frame shape: $(N_{\text{sweep}}, N_{\text{int}}, N_{\text{baselines}}, N_{\text{pol}})$
Frame type: 32-bit complex float
Frame size: 256 MiB per frame (if $N_{\text{sweep}} = 512$)
Frame mode: single

2.5 Beamformer modes

The AAVS stations use frequency-domain beamforming to combine antennas and form a phased-array beam. If we consider the output of a coarse channel to be a voltage stream, $v(t)$, then the *voltage beam* from N_{ant} antennas is given by multiplying each stream with a complex weight $w = Ae^{i\theta}$, then summing the weighted streams together. To account for polarization the TPMs also apply a 2×2 Jones calibration matrix:

$$\begin{pmatrix} b_x(t) \\ b_y(t) \end{pmatrix} = \sum_{p}^{N_{\text{ant}}} w_p \begin{pmatrix} c_{xx} & c_{xy} \\ c_{yx} & c_{yy} \end{pmatrix}_p \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix}_p \tag{3}$$

where the subscript p is antenna number, $(v_x, v_y)_p$ are the voltage streams for X and Y polarizations, (b_x, b_y) are the calibrated beams for the two polarizations, w_p is the weight for beam steering (i.e. geometric delay term), and the C_p matrix corrects for everything else (polarization, ionospheric phase, receiver complex gain, station tapering, etc). Frequency-domain beamforming theoretically allows for each channel to be pointed independently, but this functionality is not implemented for single-station commissioning.

Antenna flagging When beamforming, bad antennas need to be flagged so that they do not impact the station performance. This means that a list of bad antennas is needed, and must be passed at run-time. Currently this must be handled by the user.

Parameter		Allowed values
F_{ant}	Boolean antenna flag array	0 or 1

Phase Weights The beam is steered using a set of phase weights applied to each incoming antenna stream. The phase weights are calculate to point in a given direction, in RA/DEC or ALT/AZ coordinates.

Parameter		Allowed values
RA	Right ascension	0-24 hr
DEC	Declination	-90 to +90 deg
ALT	Altitude	0-90 deg
AZ	Azimuth angle	0-360 deg

Note that the beam steering is controlled separately from the data capture (beam coefficients can be updated during the observation, without interrupting the data stream), whereas antenna flags cannot be changed during capture.

2.5.1 Voltage station beam (M7)

A single station beam is formed, and voltage stream is written to disk. This is one of the higher data rate modes for single-station observations. Bandwidths of up to 75 MHz (96 channels) are supported; it is possible to receive the full 300 MHz bandwidth, but this should be considered experimental.

Parameter		Allowed values
N_{beam}	Number of beams	1
i_{chan}	Start channel ID	1–511
N_{chan}	Number of coarse channels	1–96

Frame shape: $(N_{\text{beam}}, N_{\text{chan}}, N_{\text{pol}})$
Frame type: 8-bit complex signed integer (4 bytes)
Frame size: 256 B per frame (for $N_{\text{chan}} = 32$)
Frame mode: continuous stream
Data rate: 226 MiB per second (for $N_{\text{chan}} = 32$)

2.5.2 Power station beam (M8)

The power station beam detects (i.e. squares) the voltage data and then averages across time. This is a measure of total power. The time-averaging significantly decreases the data volume written to disk. As with the voltage beam mode, up to 75 MHz (96 channels) are supported, and power data written to disk.

Parameter		Allowed values
N_{beam}	Number of beams	1
t_{int}	Integration time	$\geq 1.08\,\mu$s
i_{chan}	Start channel ID	1–511
N_{chan}	Number of coarse channels	1–96

Frame shape: $(N_{\text{beam}}, N_{\text{chan}}, N_{\text{pol}})$
Frame type: 64-bit float (8 bytes)
Frame size: 512 B per frame (for $N_{\text{chan}} = 32$)
Frame mode: continuous stream
Data rate: 512 B – 452 MiB per second ($1.08\mu s < t_{\text{int}} \leq 1\,s$)

2.6 Bandpass monitoring

2.6.1 Antenna bandpass (M9)

The antenna bandpass mode is primarily used for monitoring. Antenna bandpasses (i.e. measurements of power spectral density, PSD), are generated from detection and integration of the F-engine's coarse channels.

Parameter		Value
N_{coarse}	Number of coarse channels	512
N_{ant}	Number of antennas	256
N_{pol}	Number of polarizations	2

Parameter		Allowed values
t_{int}	Integration time	0.5s

Frame shape: $(N_{\text{coarse}}, N_{\text{ant}}, N_{\text{pol}})$
Frame type: 32-bit float (4 bytes)
Frame size: 1 MiB per frame
Frame mode: single

3 Observation configuration schema

In Section 2, we defined and detailed the single-station commissioning modes of operation and their observing parameters. Here, we define a schema to fully describe a single-station observation using these modes. This schema allows exact and explicit communication of desired observations between Science Operations and System Science / Science Commissioning teams. For example, a correlator mode observation would look like:

```yaml
obs_config:
  station: S9-1
  mode: correlator
  sub_mode: fixed
  intent: All-sky monitoring, December 2023
  notes: Additional notes if needed (optional)
scan_config:
  utc_start: 2023-12-25 12:30:00.00
  utc_start_format: iso
  obs_duration: 60
  obs_duration_unit: min
frequency_config:
  n_channel: 1
  start_channel: 64
time_config:
  time_resolution: 0.5
  time_resolution_unit: s
```

By design, it *only* includes observing parameters; it does *not* specify data location, telescope initialization, calibration parameters, etcetera. Here, we have written the schema in YAML[2] for readability, as it would stored within a metadata file, `ska-data-product.yaml`, as defined in ADR-55 **[RD11]**.

3.1 Specification Overview

The observation config schema roughly corresponds to a scheduling block (SB) within the SKA operation execution model **[RD1, §6.4.4.2]**: it can be considered an atomic unit of observation planning that contains all observing attributes necessary for it to be executed. It is designed to be both human-readable and machine-readable. Each YAML file consists of:

[2] YAML: Yet another markup language

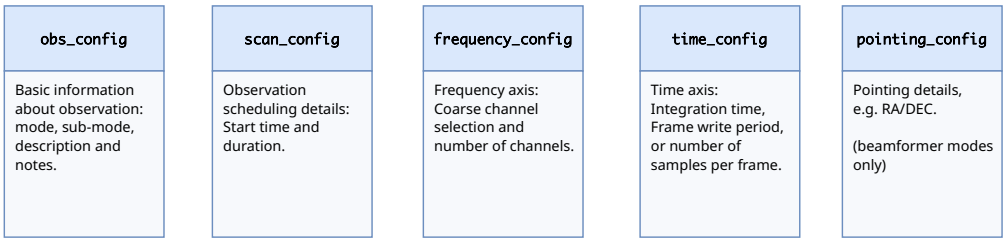


Figure 1: The primary YAML containers used to create an observation config. Only the `obs_config` and `scan_config` containers are required by all modes.

Containers The YAML file is comprised of a set of 'containers', which contain configuration information for logical groupings of metadata (Figure 1). All modes require a `obs_config` container and a `scan_config` container. Other containers may be needed, depending on the mode.

Attributes Each container consists of a number of key-value pairs, known as attributes. Some attributes are mandatory, while others are optional. Attribute IDs follow a `camel_case` convention, e.g. `utc_start`.

Units Any attribute `XX` with corresponding units must be accompanied by an attribute `XX_unit`, with a SI unit string as defined in Astropy's Units and Quantities documentation. This approach allows for unambiguous and flexible parsing of quantities.

Unit shorthand Any attribute `XX` with corresponding units `XX_unit` can be written in one line by explicitly adding SI units at the end. For example, `obs_duration: 1 hr`.

Sky and Time Coordinates As with units, sky and time coordinates are parsed using Astropy Sky Coordinates and Time and Date representations.

Convenience attributes Some containers allow alternative approaches to configuration. These alternatives are 'convenience attributes', which when executed are converted into the base attribute. For example, the start time in `scan_config` can be set via `utc_start` or `lst_start`. However, when executed, `lst_start` is converted into the base attribute: a `utc_start` time. Some rounding issues may occur in this conversion process—particularly when selecting frequency ranges—which must be handled by the observation execution tool.

3.2 Common containers

3.2.1 The `obs_config` container

The YAML file starts with a `obs_config` container, that contains basic information about the observation (i.e. provides context). Three mandatory attributes must be present:

Attribute	Description
`mode`	Observing mode, must be one of `correlator`. `beamformer`, `adc-capture`, or `channel-voltages`
`sub_mode`	Observing mode subtype, e.g. `sweep`. Depends on selected `mode`
`intent`	Simple description of the observation, e.g. `"sun observations for calibration"`
`notes`	Any additional notes, e.g `"poor RFI conditions expected"` (optional)

3.2.2 The `scan_config` container

This container configures the observational time period. Observations must define a start time with `utc_start`. The convenience attribute `lst_start` allows an upcoming LST to be used instead. Another convenience attribute `timed_start` allows an observation to be started immediately (`now`), or after a specified time period (e.g. 1 hour) has elapsed (measured relative to when the execution tool receives it).

For continuous modes (M3, M5, M7), an observation length with `obs_duration` must also be set. Corresponding unit attributes (`XX_unit`), or time format (`XX_format`) must also be set.

Attribute	Description
`utc_start`	UTC start time.
`utc_start_format`	UTC start format. Must be one of `iso, isot, unix, mjd, jd, gps`, as defined in astropy documentation
`obs_duration`	Length of observation, e.g. 60 m
`obs_duration_unit`	Unit must be one of `us, ms, s, min, h`.
Convenience attributes	
`lst_start`	Local Sidereal start time. E.g. 12.50 or "12:30:00"
`lst_unit`	LST unit. Must be one of `hourangle, deg, rad`.
`timed_start`	`now`, or start after specified time period. E.g 1 min
`timed_unit`	Unit must be one of `us, ms, s, min, h`

3.2.3 The `frequency_config` container

This container configures the frequency and observing bandwidth of the observation, and is required by all modes except ADC sample capture (M1, M2). This must be configured with a start channel (`start_channel`) and number of channels (`n_channel`).

Attribute	Description
`start_channel`	Index of first channel in selection (0–511).
`n_channel`	Number of channels in selection.
Convenience attributes	
`start_frequency`	Start frequency for observing band
`start_frequency_unit`	One of `Hz, kHz, MHz, GHz`
`center_frequency`	Central frequency for observing band
`center_frequency_unit`	One of `Hz, kHz, MHz, GHz`
`obs_bandwidth`	Observing bandwidth
`obs_bandwidth_unit`	One of `Hz, kHz, MHz, GHz`

Notes The channel can also specified by using the convenience attributes `start_frequency` (or `center_frequency`) and `obs_bandwidth`, to specify the start (or center) frequency and the observing bandwidth. These are converted into `start_channel` and `n_channel` via:

$$\texttt{start\_frequency} = \texttt{center\_frequency} - \texttt{obs\_bandwidth}/2 \tag{4}$$

$$\texttt{start\_channel} = \text{round}\left(\frac{\texttt{start\_frequency}}{\nu_{\text{spacing}}}\right) \tag{5}$$

$$\texttt{n\_channel} = \texttt{obs\_bandwidth}/\nu_{\text{spacing}} \tag{6}$$

where ν_{spacing}=0.78125 MHz is the coarse channel spacing.

3.3 ADC capture modes (M1, M2)

ADC capture modes require the following containers:

`obs_config`

`scan_config`

Allowed configuration values are shown in the table below.

Attribute	Description
`obs_config:`	
`mode`	`adc-capture`
`sub_mode`	Must be set to either `synchronous` (M1) or `asynchronous` (M2).

Notes `scan_config` does not require `obs_duration` as ADC capture modes receive either 4096 samples (synchronous, M1) or 32768 samples (asynchronous, M2). Antenna transient buffer data is not currently supported.

3.4 Channel voltage capture modes (M3, M4)

Channel voltage capture modes require the following containers:

`obs_config`
`scan_config`
`frequency_config`
`time_config`

Allowed configuration values are shown in the table below.

Attribute	Description
`obs_config:`	
`mode`	`channel-voltages`
`sub_mode`	Must be set to either `fixed` (M3) or `sweep` (M4).
Fixed frequency (M3)	
`time_config:`	
`samples_per_frame`	Time samples to capture per frame
`samples_per_frame_unit`	Must be one of `samples, us, ms, s, min, h`
`frame_write_period`	How often a frame is written (fixed mode only)
`frame_write_period_unit`	Must be one of `samples, us, ms, s, min, h`
Frequency Sweep (M4)	
`time_config:`	
`samples_per_frame`	Time samples to capture per frame
`samples_per_frame_unit`	Must be one of `samples, us, ms, s, min, h`

Notes Channel voltage capture modes require a `frequency_config` container. Currently, the `fixed` submode requires that `n_channel=1`. The `sweep` submode will treat `n_channel` as the number of channels to sweep across.

The `fixed` mode is often run with `samples_per_frame=524288`, whereas the `sweep` mode normally uses a lower value of `samples_per_frame=128`.

3.5 Correlator modes (M5, M6)

Correlator modes require the following containers:

`obs_config`
`scan_config`
`frequency_config`
`time_config`

Allowed configuration values are shown in the table below.

Attribute	Description
`obs_config:`	
`mode`	`correlator`
`sub_mode`	Must be set to either `fixed` (M5) or `sweep` (M6).
Fixed frequency (M5)	
`time_config:`	
`time_resolution`	Time resolution / integration time (per frame). Must be between 0.283–2.264 seconds
`time_resolution_unit`	Must be one of `us, ms, s, min, h`.
Frequency Sweep (M6)	
`time_config:`	
`time_resolution`	Time resolution / integration time (per frame). Must be between 0.283–2.264 seconds
`time_resolution_unit`	Must be one of `us, ms, s, min, h`.
`n_int`	Number of integration timesteps to capture for each channel. Must be between 1 or 2

Notes For correlator modes, the `sub_mode` attribute must be set to either `fixed` (M5) or `sweep` (M6). Correlator modes must also have a `time_config` container set, to configure the integration time used in the correlator.

3.6 Station beamformer modes

The station beamforming modes require the following containers:

`scan_config`
`frequency_config`
`pointing_config`
`antenna_flags`
`time_config` - *power beam mode only*.

Allowed configuration values are shown in the table below.

Attribute	Description
`obs_config:`	
`mode`	`beamformer`
`sub_mode`	Must be set to either `voltage` (M7) or `power` (M8).
`frequency_config:`	
`start_channel`	Index of first channel in selection (0–511).
`n_channel`	Number of channels in selection.
`antenna_flags`	List of antenna flags (can use `!include` constructor)
Power beam (M8)	
`time_config:`	
`time_resolution`	Time resolution / integration time (minimum 1.08 μs).
`time_resolution_unit`	Must be one of `us, ms, s, min, h` .

Notes Details of the `pointing_config` container and `antenna_flags` container are provided in the subsections below.

3.6.1 The `pointing_config` container

There are five specified pointing modes:

- `celestial` - this corresponds to a sky coordinate on the celestial sphere. The ICRS frame is used by default, but different frames (e.g. FK5, FK4) can also be specified, as long as these are supported by astropy.
- `altaz` - point the telescope at a fixed altitude and azimuth angle.
- `zenith` - point the telescope at zenith, e.g. to perform a drift scan. This is equivalent to an `altaz` pointing at 90° altitude.
- `solarsys` - point the telescope toward a solar system body.

- `tle` - track an Earth satellite using its Two-Line Element (TLE) ephemeris.

Allowed `pointing_config` values are shown in the table below.

Attribute	Description
`pointing_config:`	
`tracking`	One of `celestial`, `altaz`, `solarsys`, `tle` or `zenith`
Celestial	
`ra`	Right ascension, e.g. `17:45:00` or `17.75`
`ra_unit`	One of `hourangle, deg, rad`.
`dec`	Declination, e.g. `-32:45:00` or `-32.75`
`dec_unit`	One of `hourangle, deg, rad`.
`frame`	*Optional.* If not set, defaults to `icrs`
Alt-Az	
`alt`	Pointing altitude, e.g. `0-90` degrees
`alt_unit`	One of `hourangle, deg, rad`.
`az`	Pointing azimuth, e.g. `0-360` degrees
`az_unit`	One of `hourangle, deg, rad`.
Solar system objects	
`body`	Name of solar system body One of `sun, moon, mercury, venus, mars` `jupiter, saturn, uranus, neptune`
`ephemeris`	*Optional.* JPL ephemeris to use. Defaults to `de432s`
Earth satellites	
`tle1`	First line of TLE.
`tle2`	Second line of TLE.

3.6.2 The `antenna_flags` container

The beamformer requires an `antenna_flags` container, which has a list of flagged antennas by ID. A value of 1 means the antenna is flagged (i.e. to be excluded from beamforming):

```
antenna_flags:
  - 0: 0
  - 1: 0
  - 2: 1
  ...
```

To make this easier, you can reference an existing flags file with the `!include` constructor:

```
antenna_flags: !include path/to/flags.yaml
```

3.7 Example YAML configurations

3.7.1 Correlator: Fixed-frequency (M5)

Run the correlator for 60 minutes, selecting channel 64, with 0.5 s time integrations:

```yaml
obs_config:
  mode: correlator
  sub_mode: fixed
  intent: All-sky monitoring, December 2023
scan_config:
  utc_start: 2023-12-25 12:30:00.00
  utc_start_format: iso
  obs_duration: 60
  obs_duration_unit: min
frequency_config:
  n_channel: 1
  start_channel: 64
time_config:
  time_resolution: 0.5
  time_resolution_unit: s
```

3.7.2 Correlator: Frequency sweep (M6)

Run a frequency sweep across channel 50–150 MHz (channels 64–192), with 2×0.5 s time integrations per channel, using the abridged unit shorthand:

```yaml
obs_config:
  mode: correlator
  sub_mode: sweep
  intent: Calibration using EEPs and sun
scan_config:
  utc_start: 2023-12-25 12:30:00.00
  utc_start_format: iso
frequency_config:
  start_frequency: 50 MHz
  obs_bandwidth: 100 MHz
time_config:
  time_resolution: 0.5 s
  n_int: 2
```

3.7.3 Beamformer: Power beam (M8)

Run the beamformer for 5 minutes, selecting channel 100-131 (25 MHz band), with 0.5 s time integrations:

```
obs_config:
  mode: beamformer
  sub_mode: power
  intent: Sun observations
scan_config:
  utc_start: 60216.0
  utc_start_format: mjd
  obs_duration: 5
  obs_duration_unit: min
pointing_config:
  tracking: celestial
  ra: 17:00:00
  ra_unit: hourangle
  dec: 100:00:00
  dec_unit: deg
frequency_config:
  n_channel: 32
  start_channel: 100
time_config:
  time_resolution: 0.5
  time_resolution_unit: s
```

Equivalently, this could be rewritten to include units on the same line as their attribute:

```
obs_config: (as above)
scan_config:
  utc_start: 60216.0
  utc_start_format: mjd
  obs_duration: 5 min
pointing_config:
  tracking: celestial
  ra: 17:00:00 hourangle
  dec: 100:00:00 deg
frequency_config: (as above)
time_config:
  time_resolution: 0.5 s
```

4 Station data products

TODO: This section is under development.

The AAVS2 scripts output data to **scratch data formats** (4.1), which are mainly stored in HDF5 files. These scratch formats contain a minimal amount of metadata, and need to be converted into **science data formats** (4.2) like UVFITS or CASA MeasurementSets before analysis. This conversion entails data quality checks, flagging, optional calibration, and inclusion of metadata such as antenna positions.

For long-term storage, data should be converted into an **archival data format** (4.3). Storing data in the science data format is a reasonable choice in many instances; however, an archival format should minimize file size and the number of files (many small files are bad for distributed file systems). The archival format should well documented, and ideally be self-describing and readable with common software packages.

4.1 Scratch Data Formats

- `stationbeam_integ_0_20230926_27350_0.hdf5`
- `correlation_burst_100_20230922_31858_0.hdf5`
- `channel_cont_0_20221130_16922_0.hdf5`
- `channel_0_1_1694045777.415959.dada`

The file structure of a HDF5 file can be printed to screen using the `h5ls` utility:

```
$ h5ls -r -v filename.hdf5
```

4.1.1 Common datasets

All HDF5 files will contain a `root` group, and `observation_info` group, in which observation details are stored as HDF5 attributes. It will also contain a `sample_timestamps` group:

Name	Description
/root	Root group for metadata (stored as attributes)
/observation_info	Observation information (stored as attributes)
/sample_timestamps/data	Dataset with unix timestamps along time axis

4.1.2 Common metadata (attributes)

Metadata are stored as attributes attached to the `observation_info` and `root` groups:

Attribute	Description	Datatype	Example
/observation_info			
description	Text description of observation	str	""
software_version	git commit ID of software	str	"58e710eb0..."
/root			
channel_id	Channel index (ID)	int	0
data_mode	?	str	""
data_type	dtype of dataset	str	"double"
date_time	UTC datetime, YYYYMMDD_HHMMSS	str	"20230926_27350"
n_antennas	Number of antennas in file	int	16
n_baselines	Number of baselines (unused?)	int	0
n_beams	Number of beams in file	int	1
n_blocks	Number of blocks (unused?)	int	9532
n_samples	Number of time samples (unused?)	int	1
n_stokes	Number of polarization products	int	4
station_id	Station ID	int	0
tile_id	Tile ID within station	int	0
timestamp	Unix start timestamp (UTC)	double	1.695710e+09
ts_end	Unix stop timestamp (UTC)	double	1.695711e+09
ts_start	Unix start timestamp (UTC)	double	1.695710e+09
tsamp	Integration time (s)	double	1.13246
type	?	int	5

4.1.3 Correlation burst

Correlated data `correlation_burst_0_20230926_27350_0.hdf5`. Each file contains all TPMs.

Name	Description
/root	Root group for metadata (stored as attributes)
/observation_info	Observation information (stored as attributes)
/correlation_matrix/data	Inter-antenna cross correlations dataset **axes:** `(time, channel, baselines, stokes)` **dtype:** `complex64`
/sample_timestamps/data	Dataset with unix timestamps for each datum

4.1.4 Continuous Channel

Continuous channel data `continuous_channel_0_20230926_27350_0.hdf5`. Each file corresponds to one TPM output.

Name	Description
/root	Root group for metadata (stored as attributes)
/observation_info	Observation information (stored as attributes)
/chan_/data	Inter-antenna cross correlations dataset **axes:** `(sample, antpol)` **dtype:** `compound (int8 real, int8 imag)`
/sample_timestamps/data	Dataset with unix timestamps for each datum

4.1.5 Station beam integ

Station power beam data `stationbeam_integ_0_20230926_27350_0.hdf5`.

Name	Description
/root	Root group for metadata (stored as attributes)
/observation_info	Observation information (stored as attributes)
/polarization_0/data	Dataset for polarization X **axes:** `(time, channel)` **dtype:** `float64`
/polarization_1/data	Dataset for polarization Y **axes:** `(time, channel)` **dtype:** `float64`
/sample_packets/data	Dataset with packet information
/sample_timestamps/data	Dataset with unix timestamps for each datum

4.1.6 Raw burst

Raw burst `raw_burst_0_20230926_27350_0.hdf5`. Each file corresponds to one TPM output.

Name	Description
/root	Root group for metadata (stored as attributes)
/observation_info	Observation information (stored as attributes)
/raw_/data	ADC sample data **axes:** `(antpol, sample)` **dtype:** `int8`
/sample_timestamps/data	Dataset with unix timestamps for each datum

4.1.7 Station beam DADA file header

DADA files consist of a 4096B header, followed by the data payload. Each header metadata item is encoded as an 80-character ASCII line. While there is no official requirement for DADA keywords, the pulsar processing toolkit DSPSR can convert DADA files into PSRCHIVE files if a specific set of keywords is found in the header.

4.2 Conversion to Science Data Formats

A Python package, `ska-ost-low-uv`, is currently in development to convert scratch data into science and archival formats.

4.2.1 Conversion to UVFITS

4.2.2 Conversion to SDP Visibilities

4.2.3 Conversion to PSRCHIVE

4.3 Archival data formats

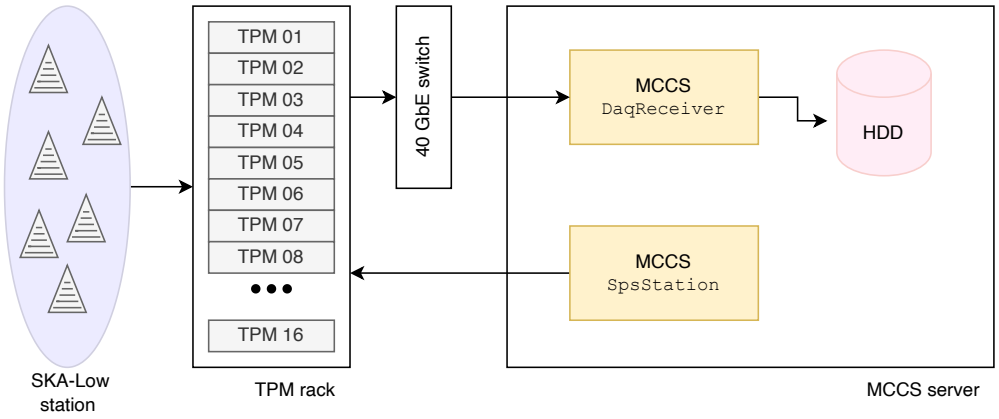


Figure 2: A simplified block diagram of an SKA-Low station being used as a single station. The TPMs are configured using the `SpsStation` class in the `ska-low-mccs-spshw` Python package. The TPMs output antenna data over high-speed Ethernet; these data are captured and processed using the `DaqReceiver` class and written to disk.

A Theory of Operation

TODO: Finish updating for AA0.5

From a simple operations perspective, an SKA-Low station consists of:

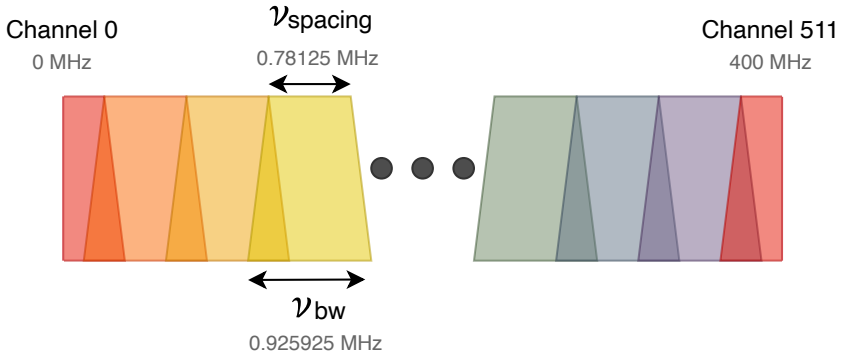
- **256× dual-polarization antennas** (512 ant-pols) and corresponding amplifiers / filters.
- **16× Tile-Processing Modules (TPMs)**, which do analog-to-digital conversion, and apply initial digital signal processing. Each TPM processes 32 ant-pol inputs (i.e. 16 dual-pol antennas), and outputs data as a stream of UDP Ethernet packets.
- **1× 40 GbE network switch** that transfers antenna data streams from the TPMs to the data capture server over high-speed Ethernet.
- **1× Data capture server** that provides monitor & control of the TPMs, and also runs a data capture pipeline to process incoming antenna data streams.

A basic block diagram of an SKA-Low station is shown in Figure 2.

SKA-Low stations are controlled with a system called MCCS. For single-station commissioning modes, the station can be configured using the `ska-low-mccs-spshw` Python package. This package contains a `SpsStation` class, which is used to instruct the TPMs to send data, and a `DaqReceiver` class, which captures data.

A.1 TPM Overview

Digitization The TPMs digitize the incoming voltages using analog-to-digital (ADCs) converters. The ADCs run at 800 Msamples/s (400 MHz Nyquist bandwidth), resulting in a 1.25 ns sampling period. The ADCs provide 12-bit signed real-valued data, but these are stored as 16-bit integers in output data (12-bit data are horrible to work with).



F-engine A primary function of the TPMs is to act as an 'F-engine' to split the incoming antenna data streams into 512 frequency channels. The TPMs apply an oversampled polyphase filterbank to split the antenna data streams into 512 'coarse' channels, spaced $400/512 = 0.78125$ MHz apart. The data are oversampled by a factor of 32/27, so each channel has a bandwidth of $32/27 \times 400/512 \approx 0.9259$ MHz. Data from each channel may be output as 8+8 bit complex-valued data.

Partial beamforming The TPMs are also used for beamforming: phase-coherent summation of all antenna signals to form a 'beam'. As each TPM can only process a subset of antennas (16 dual-pol / TPM), each TPM combines its antenna signals to form 'partial beams', which can be summed together in a secondary step to form the 'station beam'. Partial beams are output as 16+16 bit complex-valued data.

The TPMs implement a frequency-domain beamformer, meaning a complex phase weight is applied to each channel before summation (as opposed to a time delay). These phases are generated using a delay generation module on the TPM. After this is computed, a per-antenna calibration is applied to correct for receiver gains, a polarization effects, the ionosphere, optional tapering and to zero-out bad antennas. The partial beam is formed by summing phase-corrected and calibrated antenna signals together.

As per SKA-low specifications, the beamformers are able to form 48 independent beams with different pointing directions, frequency tunings, and sub-array modes. These options are not currently exposed in the `SpsStation` control class.

A.2 Tile-level data products

While the output of the TPMs is not a final data product, it is nonetheless useful to understand what data types the TPMs can deliver. As per Table 6-1 in **[RD3]**, there are seven different packet output modes; of these, there are three main types:

1. **ADC samples:** raw samples from the ADCs, before the data are channelized in the F-engine. Data array shape is (N_{sample}, $N_{\text{ant}} = 16$, $N_{\text{pol}} = 2$) of real-valued signed integer data. Only a short burst of ADC samples can be recorded.

2. **F-engine voltages:** voltage-level complex data streams from one (or more) coarse channels. Antenna streams are not combined. Data array shape is (N_{channel}, N_{sample}, $N_{\text{ant}} = 16$, $N_{\text{pol}} = 2$) of 8+8 bit complex signed integer data.

3. **Partially beamformed data:** the summed voltage-level complex data stream. Data array shape is (N_{sample}, $N_{\text{beam}} = 1$, $N_{\text{pol}} = 2$) of 16+16 bit complex signed integer data.

These tile-level data products are captured by the server, and form the basis of all station-level data products.

A.3 Summary

To recap, TPMs are configured using the `SpsStation` class in the `ska-low-mccs-spshw` Python package. The TPMs output antenna data over high-speed Ethernet; these data are captured and processed using the `DaqReceiver` class and written to disk.