OSKAR Station Simulator: Data Flow and Interface Specification

This document describes the data transfer and interface between each major component of the OSKAR station simulator (see OSKAR station simulator: conceptual diagram).

A description of the data, the quantity and the rate are described at each interface.

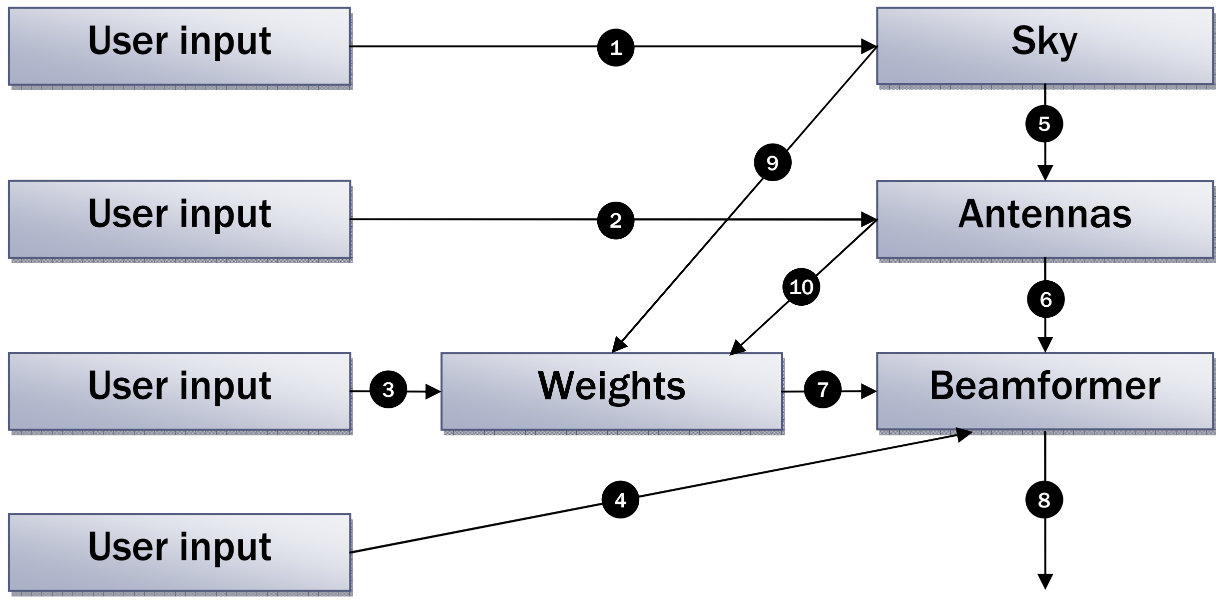

Figure 1: Data flow description of the simulator. Blocks represent discrete elements of the code. Data flow elements have numbers referring to sections in the document.

A. Introductory notes

1.1 Data Flow and Interface summary

This section provides a brief summary of all the data links in Figure 1.

1.1.1 User input to sky simulation

For the sky simulation, the user must supply: a flag specifying the simulator's mode of operation; the parameters of the observation (including the station position, start time, sample rate, frequency channels, etc); a list of celestial sources and their properties (amplitude and phase for each polarisation); and a set of time- and direction-dependent corruptions specifying amplitude and phase offsets for each polarisation.

1.1.2 User input to antenna simulation

For the antenna simulation, the user must supply: a list of all antenna positions in the station, relative to the station phase centre; non-directional amplitude and phase corruptions (to include effects such as cross coupling, broken antennas and polarisation mixing, for each antenna, time, channel, and polarisation); and the antenna element pattern (amplitude and phase offsets to indicate how the sky is modified by the antenna response).

1.1.3 User input to weights generation

For the weights generation, the user must supply: the level configuration structure (see Figure 2); the levels at which to form complete beams; the algorithm choice for each level; correction options; the correction source; optional apodisation weights; a beam configuration database to specify the required beams at each level; and the parameters of the observation (including the station position, start time, sample rate, frequency channels, etc).

1.1.4 User input to beamformer

For the beamforming module, the user must only specify the list of levels at which to form complete beams.

1.1.5 Sky simulation to antenna simulation

The sky simulation module must pass a buffer of data to the antenna simulation module. The buffer consists of a list of sources with positions in local horizon coordinates for each time-sample and channel, and their corresponding amplitude and phase offsets in both horizontal and vertical polarisations.

1.1.6 Antenna simulation to beamformer

The antenna simulation module must pass a buffer of data to the beamformer module. The buffer consists of a set of complex antenna signals for each time-sample and channel, in both horizontal and vertical polarisations.

1.1.7 Weights generation to beamformer

The weight generation module must pass a buffer of complex weights to the beamformer module. The time indexing of the buffer occurs at a different (slower) rate to the channelised antenna signal sample rate. The weights should be applied to each element of each detector at each level in the station hierarchy.

1.1.8 Beamformer to (next stage) output processing

The beamformer will produce a buffer of complex beam amplitudes for each channelised time sample.

1.1.9 Sky simulation to weights generation

The sky simulation module passes the direction-dependent corruptions to the weights generator, in the case that these are required to apply corrections.

1.1.10 Antenna simulation to weights generation

The antenna simulation module passes the non-directional corruptions to the weights generator, in the case that these are required to apply corrections.

1.2 Typographical conventions

- Round brackets ( ) provide additional information on listed parameters.
- Square brackets [ ] describe the parameterisation of data structures.

1.3 Polarisation

Polarisation will be parameterised by horizontal and vertical components of the amplitude and phase of the channelised source signals.

1.4 Data transfer

Many of the options and data will be passed to the back-end simulator via human-readable text files. The actual interfaces between modules would therefore consist only of file pointers, rather than transferring large amounts of data explicitly.

1.5 Data representation

The OSKAR Simulator will be developed using fixed-precision floating-point arithmetic, because of the restrictions imposed by the current available hardware. However, we are well aware that the real SKA may use lower bit depth, possibly varying bit-depth in its various stages. The effects of using lower-precision arithmetic than the hardware default could be studied, although at much increased computational cost.

1.6 Data type conventions

Data types have the following sizes (assuming 8 bits per byte):

- Boolean value (bool): ~ 1 byte
- Single-byte value (char): 1 byte
- Integer value (int): 4 bytes
- Single-precision floating-point value (float): 4 bytes
- Double precision floating-point value (double): 8 bytes

1.7 Data rate assumptions

In the examples, we assume the simulator is working with the following parameters:

- A sky model of 100 point sources in 1000 channels
- Only 10 active channels
- A post-channelised sample rate of 100 Hz
- 65536 antennas per station
- 2 levels of beamforming (tile, station)
- 10 beams per level
- 10 minute (600 second) observation

B. The station configuration

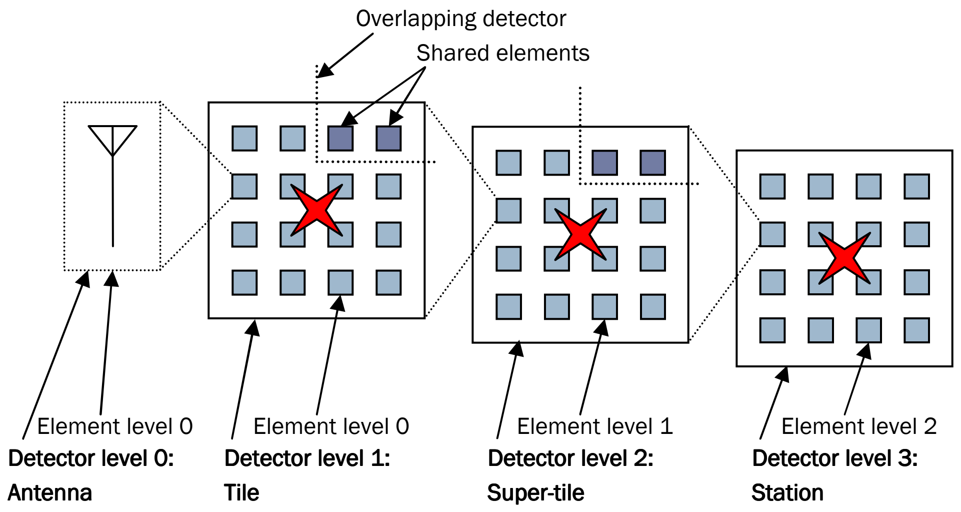

Figure 2: The station configuration in terms of element and detector hierarchy. Each detector phase-centre is marked in red

The station configuration specifies how elements are grouped into detectors for each level in the beamformer hierarchy, as shown in Figure 2. For example, individual antennas (elements and detectors at level 0) may be grouped into tiles (detectors at level 1). Tiles (elements at level 1) may be grouped into super-tiles (detectors at level 2), and super-tiles (elements at level 2) may be grouped into a station (the final detector, at level 3). In general, each element may belong to more than one detector, so, for example, different logical tiles may share a single antenna. The station configuration database must start at the antenna level (level 0), and end at the station level (level *n*, where *n* is specified by the user).

1 User input to sky simulation

1.1 Data description

- Mode of operation flag
  - Full simulation
  - Diagnostic
- Observation parameters structure
  - Channel to frequency mapping
  - Frequency channels in use (observation channels)
  - Start sidereal time
  - Length of observation
  - Sample rate
  - Station position on Earth
  - Sky grid resolution for diagnostic mode
- Source map/structure [source ID]
  - Source type (e.g. various presets)
  - Celestial coordinate position (RA, Dec)
  - Amplitude [polarisation, frequency channel]
  - Phase [polarisation, frequency channel)]
- Direction-dependent corruptions maps [time]
  - Corruption map type (e.g. gridded, point sources)
  - Corruption map timestamp
  - Corruption map/structure [position ID]
    - Local coordinate position (azimuth, elevation)
    - Amplitude offset [polarisation, frequency channel]
    - Phase offset [polarisation, frequency channel]

1.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Observation parameters	struct parameters	(	(
	float frequency[channel ID]	4 * #[channels] +	4 * 1000 +
	int channel ID[obs. channel ID]	4 * #[obs. channels])	4 * 10 +
	double start sidereal time	8 +	8 +
	double length of observation	8 +	8 +
	int sample rate	4 +	4 +
	float station position[2] (longitude, latitude)	4 * 2 +	8 +
	float diagnostic grid resolution[2] (azimuth, elevation)	4 * 2)	8) = 4 kB
Source map	struct sources[source ID]	#[sources] * (	100 * (
	char source type	1 +	1 +
	float position[2] (RA, Dec)	4 * 2 +	4 * 2 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[channels]	4 * 2 * 1000 +
	float phase[polarisation][channel]	4 * #[polarisations] * #[channels])	4 * 2 * 1000) = 1.6 MB
Direction-dependent corruptions	struct corruptions[time]	#[times] * (	600 * (
	float time stamp	4 +	4 +
	char corruption map type	1 +	1 +
	struct corruption data[position ID]	#[positions] * (	32000 * (
	float position[2] (azimuth, elevation)	4 * 2 +	4 * 2 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase[polarisation][channel]	4 * #[polarisations] * #[obs. channels]))	4 * 2 * 10)) = 3.3 GB

Data is provided by the user prior to starting the simulator, and is loaded only once. Any corruptions or sources which are time-variable must therefore be parameterised by a time varying model/function at this stage.

2 User input to antenna simulation

2.1 Data description

- Antenna (*x*, *y*) position within station [antenna ID]
- Non-directional corruptions [antenna ID, time] (e.g. cross coupling, broken antennas, polarization mixing)
  - Amplitude gain [polarisation, frequency channel]
  - Phase offset [polarisation, frequency channel]
  - Polarisation mixing percentage
- Antenna element pattern [position ID]
  - Local coordinate position (azimuth, elevation)
  - Amplitude gain [polarisation, frequency channel]
  - Phase offset [polarisation, frequency channel]

2.2 Data format and rate

Name	Type	Amount of data / bytes	Example
List of antennas	float position[2][antenna ID] (x, y)	4 * 2 * #[antennas]	4 * 2 * 65536 = 524 kB
Non-directional corruptions	struct corruption[antenna ID][time]	#[antennas] * #[times] (	65536 * 600 * (
	double timestamp	8 +	8 +
	char corruption type	1 +	1 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase [polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float polarisation mixing percentage	4)	4) = 6.8 GB
Antenna element pattern	struct antenna pattern[position ID]	#[positions] * (	32000 * (
	float position[2]	4 * 2 +	4 * 2 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase[polarisation][channel]	4 * #[polarisations] * #[obs. channels])	4 * 2 * 10) = 5.4 MB

Data is provided by the user prior to starting the simulator, and is loaded only once.

3 User input to weights generation

3.1 Data description

- Level configuration structure
    - Level of detector
        - Element ID
        - Element position relative to detector phase-centre
- Form complete beams or partial beams [level]
- Algorithm choice [level]
- Correction options
    - Type (all, directional, non-directional, none)
    - Correction source (external filename, ignored if local)
- Apodisation weights options (weight modification)
    - Type (function, user specified)
    - Amplitude [radius from phase centre]
- Beam configuration [level, beam ID, channel]
    - Beam type (beam or null)
    - Position type (local or celestial)
    - Position (RA, Dec or azimuth, elevation)
- Observation parameters structure
    - Channel to frequency mapping
    - Frequency channels in use (observation channels)
    - Start sidereal time
    - Length of observation
    - Sample rate
    - Station position on Earth
    - Sky grid resolution for diagnostic mode

3.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Level configuration structure	struct level configuration[level]	#[levels] * (	2 * (
	int number of detectors	4 +	4 +

	struct detector identifier[detector ID]	#[detectors] * (	256 * (
	float detector phase centre[2]	4 * 2 +	4 * 2 +
	int number of elements	4 +	4 +
	float element relative position[2] [element ID]	4 * 2 * #[elements] +	4 * 2 * 256 +
	int element's detector originator identifier[element ID]	4))	4)) = 1 MB
Levels at which to form complete beams	char value[level]	1 * #[levels]	3 B
Algorithm choice	char value[level]	1 * #[levels]	3 B
Apodisation weights	struct weights	(	(
	char type	1 +	1 +
	float radius[radius ID]	4 * #[radius positions] +	4 * 10 +
	float amplitude[radius ID]	4 * #[radius positions])	4 * 10) = 80 B
Beam configuration	struct beam[level][ID][channel]	#[levels] * #[beams per level] * #[channels] * (	2 * 10 * 10 * (
	char beam type	1 +	1 +
	char position type	1 +	1 +
	float position[2]	4 * 2)	4 * 2) = 2 kB
Direction-dependent corruptions	struct corruptions[time]	#[times] * (	600 * (
	float time stamp	4 +	4 +
	char corruption map type	1 +	1 +
	struct corruption data[position ID]	#[positions] * (	32000 * (
	float position[2] (azimuth, elevation)	4 * 2 +	4 * 2 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase[polarisation][channel]	4 * #[polarisations] * #[obs. channels]))	4 * 2 * 10)) = 3.3 GB
Non-directional corruptions	struct corruption[antenna ID][time]	#[antennas] * #[times] (	65536 * 600 * (
	double timestamp	8 +	8 +
	char corruption type	1 +	1 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +

	float phase [polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float polarisation mixing percentage	4)	4) = 6.8 GB
Observation parameters	struct parameters	(	(
	float frequency[channel ID]	4 * #[channels] +	4 * 1000 +
	int channel ID[obs. channel ID]	4 * #[obs. channels])	4 * 10 +
	double start sidereal time	8 +	8 +
	double length of observation	8 +	8 +
	int sample rate	4 +	4 +
	float station position[2] (longitude, latitude)	4 * 2 +	8 +
	float diagnostic grid resolution[2] (azimuth, elevation)	4 * 2)	8) = 4 kB

Data is provided by the user prior to starting the simulator, and is loaded only once.

4 User input to beamformer

4.1 Data description

- Form complete beams or partial beams [level]

4.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Levels at which to form complete beams	char value[level]	1 * #[levels]	3 B

Data is provided by the user prior to starting the simulator, and is loaded only once.

5 Sky simulation to antenna simulation

5.1 Data description

- Sky buffer
  - Header (flat):
    - Real time-stamp of start of buffer
    - Sample rate (time increment per sample)
    - Number of time samples in buffer
    - Start source ID in the buffer
    - Number of sources in the buffer
    - Start channel ID in the buffer
    - Number of channels in the buffer
  - Data (three dimensions: source, time, channel):
    - Position (local coordinates: azimuth, elevation)
    - Amplitude (horizontal and vertical polarisations)
    - Phase (horizontal and vertical polarisations)

5.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Sky data	struct sky buffer	(	(
	double timestamp (for start of buffer)	8 +	8 +
	int sample rate	4 +	4 +
	int number of time samples	4 +	4 +
	int start source ID	4 +	4 +
	int number of sources	4 +	4 +
	int start channel ID	4 +	4 +
	int number of channels	4 +	4 +
	struct data[time ID][source ID][channel ID]	#[times in buffer] * #[sources in buffer] * #[channels in buffer] (	100 * 100 * 10 * (
	float position[2] (azimuth, elevation)	4 * 2 +	4 * 2 +
	float amplitude[polarisation]	4 * #[polarisations] +	4 * 2 +
	float phase[polarisation]	4 * #[polarisations]))	4 * 2))

			= 2.4 MB

The actual layout of this data structure depends on how the functions will be parallelised.

The amount of data is 24 bytes per source per channel per time-sample, with an additional 32 byte header per buffer.

For a 100 Hz sample rate on the sky, 100 sources and 10 channels, the data rate is approximately 24 * 100 * 100 * 10 = 2.4 MB / sec.

6 Antenna simulation to beamformer

6.1 Data description

- Antenna buffer
  - Header (flat):
    - Real time-stamp of start of buffer
    - Sample rate (time increment per sample)
    - Number of time samples in buffer
    - Start antenna ID in the buffer
    - Number of antennas in the buffer
    - Start channel ID in the buffer
    - Number of channels in the buffer
  - Data (three dimensions: antenna, time, channel):
    - Amplitude (horizontal and vertical polarisations)
    - Phase (horizontal and vertical polarisations)

6.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Antenna data	struct antenna buffer	(	(
	double timestamp (for start of buffer)	8 +	8 +
	int sample rate	4 +	4 +
	int time samples	4 +	4 +
	int start antenna ID	4 +	4 +
	int number of antennas	4 +	4 +
	int start channel ID	4 +	4 +
	int number of channels	4 +	4 +
	struct data[time ID][antenna ID][channel ID]	#[times in buffer] * #[antennas in buffer] * #[channels in buffer] (	100 * 65536 * 10 * (
	float amplitude[polarisation]	4 * #[polarisations] +	8 +
	float phase[polarisation]	4 * #[polarisations]))	8)) = 1.1 GB

The actual layout of this data structure depends on how the functions will be parallelised.

The amount of data is 16 bytes per antenna per channel per time-sample, with an additional 32 byte header per buffer.

For a 100 Hz sample rate on the sky, 65536 antennas and 10 channels, the data rate is approximately 16 * 100 * 65536 * 10 = 1.1 GB / sec

7 Weights generation to beamformer

7.1 Data description

- Weights buffer:
    - Header
        - Timestamp (start of weights buffer)
        - Number of times in a buffer for tracking
        - Level identifier
        - Detector identifier
        - Start observation channel ID
        - Number of channels in buffer
        - Start beam ID
        - Number of beams in buffer
    - Data
        - Amplitude [element ID, beam ID]
        - Phase [element ID, beam ID]

7.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Weights buffer	struct weights buffer [level]	#[levels] * (	2 * (
	float time stamp	4 +	4 +
	int buffer length	4 +	4 +
	int obs. start channel ID	4 +	4 +
	int number of channels	4 +	4 +
	int start beam ID	4 +	4 +
	int number of beams	4 +	4 +
	struct detector weights[detector ID]	#[detectors] * (	256 * (
	float amp[element ID][beam ID][channel ID][time]	4 * #[elements] * #[beams] * #[channels] * [times] +	4 * 256 * 10 * 10 * 100 +
	float phase[element ID][beam ID][channel ID][time]	4 * #[elements] * #[beams] * #[channels] * #[times]	4 * 256 * 10 * 10 * 100)) = 10 GB

The beamformer receives a data rate of 10 GB per update. Assuming we update the weights for every level at each update, and the update rate for weights is 1 Hz (note that tracking directions are updated at 100 Hz), the data rate into the beamformer from the weights is 10 GB per second.

Alternatively the data rate can be parameterised by the rate at which we want to track, as 100MB per second per tracking update.

8 Beamformer to next-stage (output) processing

8.1 Data description

- Beams buffer[level]
  - Timestamp
  - Length of the buffer
  - Observation channel start ID
  - Observation channel end ID
  - Beams[channel ID][detector ID][beam ID][polarisation]

8.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Output beams buffer	struct beam[level]	#[levels] * (	2 * (
	float timestamp	4 +	4 +
	int length of buffer (time samples)	4 +	4 +
	int obs. channel start ID	4 +	4 +
	int obs. channel end ID	4 +	4 +
	float beam amplitude[channel ID] [detector ID][beam ID][polarisation][time]	4 * #[channels] * #[detectors] * #[beams] * #[polarisations] * #[times]	4 * 10 * 256 * 10 * 2 * 1 +
	float beam phase[channel ID] [detector ID][beam ID][polarisation][time]	4 * #[channels] * #[detectors] * #[beams] * #[polarisations] * #[times])	4 * 10 * 256 * 10 * 2 * 1) = 820 kB

For each timestamp, the beamformer will output 100 kB of channelised beam data. Assuming a channeliser sample rate of 100Hz this corresponds to a data rate of 10MB/s out of the beamformer.

9 Sky processing to weights generation

9.1 Data description

- Buffer of correction/corruption data
    - Time stamp (start of sky buffer)

9.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Direction-dependent corruptions	struct corruptions[time]	#[times] * (	600 * (
	float time stamp	4 +	4 +
	char corruption map type	1 +	1 +
	struct corruption data[position ID]	#[positions] * (	32000 * (
	float position[2] (azimuth, elevation)	4 * 2 +	4 * 2 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase[polarisation][channel]	4 * #[polarisations] * #[obs. channels]))	4 * 2 * 10)) = 3.3 GB

10 Antenna processing to weights generation

10.1 Data description

- Buffer of correction/corruption data
    - Time stamp (start of antenna buffer)

10.2 Data format and rate

Name	Type	Amount of data / bytes	Example
Non-directional corruptions	struct corruption[antenna ID][time]	#[antennas] * #[times] (	65536 * 600 * (
	double timestamp	8 +	8 +
	char corruption type	1 +	1 +
	float amplitude[polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float phase [polarisation][channel]	4 * #[polarisations] * #[obs. channels] +	4 * 2 * 10 +
	float polarisation mixing percentage	4)	4) = 6.8 GB