developer.skatelescope.org Documentation

Release 1.10.0

Marco Bartolini

Nov 22, 2023

CONTENTS

Library for retrieving and working with SKA Telescope Model information. What we are concerned with is enabling different SKA sub-systems to agree about information - such as shared assumptions about:

- the physical location of telescope receptors (i.e. dishes or stations), or
- configuration of the correlator and its connections to links, or
- internal configuration templates for sub-systems

This sort of information evolves relatively slowly and is in many cases too voluminous to be exchanged between systems in real time. On the other hand, especially for information characterising knowledge about the telescope, we will need to evolve it independently of the software development lifecycle.

For this purpose, this library provides:

- Means to access versioned telescope model data
- Schemas to check whether telescope model data is valid
- Ways for interpret and transform telescope model information

CHAPTER

ONE

INSTALLATION

Install using pip from the SKAO central artefact repository:

```
pip install --extra-index-url https://artefact.skao.int/repository/pypi-internal/simple ska-telmodel
```

1.1 SKA Telescope Model

Library for retrieving and working with SKA Telescope Model information. What we are concerned with is enabling different SKA sub-systems to agree about information - such as shared assumptions about:

- the physical location of telescope receptors (i.e. dishes or stations), or
- configuration of the correlator and its connections to links, or
- internal configuration templates for sub-systems

This sort of information evolves relatively slowly and is in many cases too voluminous to be exchanged between systems in real time. On the other hand, especially for information characterising knowledge about the telescope, we will need to evolve it independently of the software development lifecycle.

For this purpose, this library provides:

- Means to access versioned telescope model data
- Schemas to check whether telescope model data is valid
- Ways for interpret and transform telescope model information

1.1.1 Installation

Install using pip from the SKAO central artefact repository:

```
pip install --extra-index-url https://artefact.skao.int/repository/pypi-internal/simple ska-telmodel
```

1.2 Change Log

All notable changes to this project will be documented in this file. This project adheres to Semantic Versioning.

1.2.1 1.10.0

- Added new schema section for midcbf InitSysParams command
- Added schema and test cases for midcbf InitSysParams command

1.2.2 1.9.2

- Added new field z_pos to antenna_geojson structure
- Added documentation for station and antenna geojson

1.2.3 1.9.1

- Mid cbf scan_id changed from string to integer (SKB-254)
- Added tutorials to restructured documentation
- Support shortened "car:" URI scheme (defaults to "gitlab.com/ska-telescope/" prefix and "#tmdata" segment)

1.2.4 1.9.0

- Adds support for partial configuration with Target-offset parameters to enable 5-point calibration scans. (BTN-2052)
- Adds a new module for semantic validation of Low telescope configuration. (NAK-673)

1.2.5 1.8.2

- Bugfix: 'simpleeval' and 'astropy' are required in production, not only as development dependencies.

1.2.6 1.8.1

- Update receptor validation and example values to match ADR-32
- Added optional eb_id to CSP common schema

1.2.7 1.8.0

- Fixed semantic validation issue on receptor_id and fsp_id for AA0.5 schema.
- Add new "car://" backend type that behaves like "nexus://", but enforces that data comes from artefact repository
- Fix handling of the CAR_RAW_REPOSITORY_URL to fix behaviour in CI pipelines
- Added station_id to version 1.1 of the receptor schema
- Renamed station_name to station_label in version 1.1 of the receptor schema
- Added the various changes required to the update scripts
- Add documentation for Mid.CBF command schema

1.2.8 1.7.0

- Added new semantic validation support for AA0.5 schema

1.2.9 1.6.0

- Add schemas for Low CBF configuration commands
- Add receptor_id to version 2.1 of TMC release resource schema

1.2.10 1.5.0

- Add version 2.1 of SKA-MID assign,configure,release,scan schema to support standardised keys.

1.2.11 1.4.1

- PST schema updates following review

1.2.12 1.4.0

- Added telescope model data interface to query sources of truth on matters of static telescope information
- Added layout schemas in support to provide data for delay modelling. Including schemas for geocentric, geodectic and local positions, and fixed delays.
- Added delay model schema to CSP
- Refactored CSP version code for config to use common version check function

1.2.13 1.3.3

- SDP schema refactoring
- Implement SDP scan metadata required for multi-scan support (version 0.4, including new ReleaseResources schema)
- Add receive address propagation support for PSS & PST
- Introduced CSP schemas and examples: assignresources, scan, endscan, and releaseresources

1.2.14 1.3.2

- Using standard SKAO CI stages now
- Substantial internal code refactoring - build schemas incrementally
- Add PST (Pulsar Timing) configuration schemas to CSP
- Add PSS (Pulsar Search) configuration schemas to CSP

1.2.15 1.3.1

- Update values in example file for CSP Configure schema
- Enhance CSP Schema version check logic

1.2.16 1.3.0

- Add version 2.0 of CSP Configure schema to support standardised keys (ADR-35)
- Add version 2.0 of TMC schemas for SKA-Low to support standardised keys (ADR-35)

1.2.17 1.2.0

- Add version 0.3 of SDP schemas to support standardised keys (ADR-35)

1.2.18 1.1.0

- Introduce TMC configuration to the TMC SubArrayNode.Configure schema

1.2.19 1.0.0

- Introduced schema for TMC CentralNode and TMC SubArrayNode, currently just for SKA LOW.
- Introduced schema for MCCSController and MCCSSubarray

1.2.20 0.3.0

- Generate schema description into Sphinx documentation instead of using bootprint
- Replaces specialised validation routines by a general one that selects the schema by the URI.

1.2.21 0.2.0

- Implementation of changes in CSP configuration string according ADR-18
- Especially add stubs for PSS and PST configuration
- Rework version handling to use URIs as suggested by ADR-22

1.2.22 0.1.4

- Accept raw dictionaries instead of strings

1.2.23 0.1.3

- Added SDP schema verifications

1.2.24 0.1.2

- Added CSP schema verification

1.2.25 0.1.1

- Renamed *outputChannelOffset* to *fspChannelOffset*

1.2.26 0.1.0

- Initial release
- Added CSP interface generation

1.3 Getting Started

1.3.1 Installation

Install using `pip` from the SKAO central artefact repository:

```
$ pip install --extra-index-url https://artefact.skao.int/repository/pypi-internal/
↪simple ska-telmodel
```

1.3.2 Simple usage

List data

You can now use the command line utility to list default telescope model data:

```
$ ska-telmodel ls
instrument/mccs-configuration/station_export_w2.json
instrument/mccs-configuration/antenna_export_w2.json
instrument/ska1_low/layout/low-layout.json
instrument/ska1_low/layout/data.json
instrument/ska1_low/layout/README.md
[...]
```

You can achieve the same thing from Python as follows:

```
from ska_telmodel.data import TMData
for key in TMData(): print(key)
```

Retrieve data

You can easily retrieve data from the command line as well:

```
$ ska-telmodel cat instrument/ska1_low/layout/README.md

SKA Low layout
--------------
[...]
```

Again, the same can be achieved from Python:

```
from ska_telmodel.data import TMData
print(TMData()['instrument/ska1_low/layout/README.md'].get().decode())
```

For JSON or YAML data, you can especially retrieve it parsed:

```
print(TMData()['software/tango/ska_wide/Guidelines.yaml'].get_dict())
# -> [{'class': None, [...]
```

1.3.3 Data sources

Local directory

`ska-telmodel` has a number of default data sources built-in, which we have been querying above. However, you can override this. For instance, you can use a local directory as a source:

```
$ mkdir tmdata_demo
$ echo Test! > tmdata_demo/test.txt
$ ska-telmodel ls --sources=file://tmdata_demo
test.txt
$ ska-telmodel cat --sources=file://tmdata_demo test.txt
Test!
```

This works similarly from Python:

```
from ska_telmodel.data import TMData
tmdata = TMData(['file://tmdata_demo'])
print(tmdata['test.txt'].get().decode())
# -> Test!
```

A useful pattern is to use this to create a local copy of telescope model data (see *ska_telmodel.cli.cmd_cp()*).

Gitlab & CAR sources

You can also use any Gitlab directory as a source:

```
$ ska-telmodel ls --sources=gitlab://gitlab.com/ska-telescope/ska-telmodel?master#tmdata/
↪software
UserWarning: gitlab://gitlab.com/ska-telescope/ska-telmodel?master#tmdata/software not␣
↪cached in SKA CAR - make sure to add tmdata CI!
  warnings.warn(warning)

tango/dsh/DishManager.yaml
tango/ska_wide/Guidelines.yaml
tango/ska_wide/SKABaseDevice.yaml
tango/ska_wide/SKAMaster.yaml
```

This is useful for development, but as the warning indicates should **not** be used seriously, as Gitlab will eventually start blocking these kinds of requests. A better approach is to use the SKAO central artefact repository (CAR) as the source:

```
$ ska-telmodel ls --sources=car:ska-telmodel?master
instrument/ska1_mid/validation/mid-validation-constants.json
software/tango/ska_wide/Guidelines.yaml
software/tango/ska_wide/SKAMaster.yaml
software/tango/ska_wide/SKABaseDevice.yaml
software/tango/dsh/DishManager.yaml
```

Using the `car:` URI we are now referring to an archive artefact, typically mirroring the contents of a repository (see *Adding a New Gitlab Data Source*).

Dynamic sources

The source URIs given above point to dynamic branches (`master`), which means that the results of queries against telescope model data might change. For instance we can go:

```
$ echo Test! > tmdata_demo/test.txt
$ git switch -c my_test_branch
$ git add tmdata_demo/test.txt; git commit -m "Telescope model data test"; git push my_
↪test_branch origin
$ export SKA_TELMODEL_SOURCES=gitlab://gitlab.com/ska-telescope/ska-telmodel?my_test_
↪branch#tmdata_demo
$ ska-telmodel ls
test.txt
$ ska-telmodel cat test.txt
Test!
$ echo Test2! > tmdata_demo/test.txt
```

(continues on next page)

(continued from previous page)

```
$ git add tmdata_demo/test.txt; git commit -m "Telescope model data test 2"; git push my_
↪test_branch origin
$ ska-telmodel cat test.txt
Test!
$ ska-telmodel cat --update test.txt
Test2!
```

Note that the result of our query changed - albeit only after we passed `--update`, which forced a refresh of the cache. A CAR data source would have the same behaviour if a new package was uploaded by a CI pipeline.

In Python we would achieve the equivalent as follows:

```
from ska_telmodel.data import TMData
sources = ['gitlab://gitlab.com/ska-telescope/ska-telmodel?my_test_branch#tmdata_demo']
tmdata = TMData(sources, update=True)
print(['test.txt'].get().decode())
```

Pinning dynamic sources

This dynamic behaviour might be useful in development, but when running code in testing or production, we would like more reproduceability. This is why it is a good idea to "pin" dynamic sources to a specific version. One approach is to refer to a fixed "tag":

```
$ ska-telmodel --sources=car:ska-telmodel?1.5.0 ls
software/tango/ska_wide/Guidelines.yaml
software/tango/ska_wide/SKAMaster.yaml
software/tango/ska_wide/SKABaseDevice.yaml
software/tango/dsh/DishManager.yaml
```

Now we are effectively referring to a "telescope model data release", which is permanently stored in the CAR and will always give the same result. Note that every repository publishing telescope model data might have its own independent version history, and there's especially no connection to the version of the telescope model data library.

Another approach is to "pin" sources, which resolves them to hashes:

```
$ export `ska-telmodel -U pin`
Using car:ska-telmodel-data?~9d576afb2f8980bab1fea5d82fa80ddfa91fba21
Using car:ska-telmodel?~719f0146df1de15dfaaa1780847de656ce35c29a
Using car:mccs/ska-low-mccs?~6d98ac66b188d9943b2af19e3e5f2f317da384e8
$ echo $SKA_TELMODEL_SOURCES
car:ska-telmodel-data?~9d576afb2f8980bab1fea5d82fa80ddfa91fba21,car:ska-telmodel?~
↪719f0146df1de15dfaaa1780847de656ce35c29a,car:mccs/ska-low-mccs?~
↪6d98ac66b188d9943b2af19e3e5f2f317da384e8
```

In Python we would achieve the same as follows:

```
from ska_telmodel.data import TMData
sources = TMData(update=True).get_sources(pinned=True)
print(sources)
# -> ['car:ska-telmodel-data?~9d576afb2f8980bab1fea5d82fa80ddfa91fba21', 'car:ska-telmodel?
↪~719f0146df1de15dfaaa1780847de656ce35c29a', 'car:mccs/ska-low-mccs?~
↪6d98ac66b188d9943b2af19e3e5f2f317da384e8']
```

At this point we would be able to pass `sources` to a different component (e.g. a configured sub-system):

```
# Set telescope model data to use, issue call to other component
config['sources'] = tmdata.get_sources(pinned=True)
config['layout_key'] = 'instrument/ska1_low/layout/data.json'
otherComponent.Command(json.dumps(config))
```

Now another component (e.g. Tango device) could get the data pointed at as follows:

```
def Commnand(self, config_str):
    config = json.loads(config_str)
    tmdata = TMData(config['sources'])
    layout = tmdata[config['layout_key']]
```

At this point we could be sure that the second piece of code has exactly the same view of telescope model data - regardless of any updates to telescope model data that might have happened in the meantime.

Permanently adding or changing files

In *Dynamic sources* we used a Gitlab source to quickly add a file, but this is not how you would add files to telescope model data permanently. As explained in the last section, to add data long-term we want to make them part of telescope model data "releases" persisted in the central artefact repository (such as `car:ska-telmodel?1.5.0`).

The idea is that **any** SKAO repository can release such telescope model data packages, similar to how any repository can publish (say) Python packages. For instance, the following repositories currently publish telescope model data:

- https://gitlab.com/ska-telescope/ska-telmodel - telescope model data directly associated with the telescope model library (data for semantic validation)
- https://gitlab.com/ska-telescope/ska-telmodel-data - shared information about the telescope, such as receptor positions
- https://gitlab.com/ska-telescope/mccs/ska-low-mccs - MCCS configuration information

You can view the information coming from these repositories as usual:

```
$ ska-telmodel --sources=car:mccs/ska-low-mccs?master ls
instrument/mccs-configuration/station_export_w2.json
instrument/mccs-configuration/antenna_export_w2.json
```

To add your own information, you need to:

1. Identify the repository to add the information to. If your telescope model data does not fit into an existing repostory that publishes telescope model data, check *Adding a New Gitlab Data Source* for how to set up a new repository to publish telescope model data.
2. Add the data to the `tmdata` folder in the repository, e.g. using a merge request. Make sure you choose a good path within it, because it will be global, see *Data*. Once merged, you should be able to see your file using `ska-telmodel --sources=car:ska-your-repo?main` (assuming your main branch is called `main`, otherwise `master`)
3. Optional: Release your repository (i.e. create a tag) to create a versioned telescope model data package, which can then be accessed using `ska-telmodel --sources=car:ska-your-repo?a.b.c` where `a.b.c` is the release version.

Further information

For more in-depth guides, check *Usage Guide*. There is also an SKAO Slack channel for helping users and developers of the SKA telescope model - `#help-telmodel`.

1.4 Usage Guide

1.4.1 Data

Telescope model data is stored as *objects* identified as *keys*. Each key takes the following form:

```
[domain]/([sub-domain]/)*[name].[type]
```

Where

- `[domain]/` specifies the coarse top-level telescope model data domain.
- `[sub-domain]/` provides further hierarchical data sub-categories
- `[name]` associates a name with the telescope model data.
- `[type]` identifies the file type, which is used to identify the kind of file contents. The library currently supports `json` and `yaml`.

Names should be chosen for being self-describing and stable long-term. Top-level domains:

```
environment/...    # Environment telescopes are deployed in
instrument/...     # Telescopes and their equipment
software/...       # Software deployed to the telescopes
```

To retrieve a particular piece of data from the telescope model, csimply construct an *ska_telmodel.data.TMData* object and use the `[]` operator to access:

```
from ska_telmodel.data import TMData
tmdata = TMData()
print(tmdata['instrument/ska1_low/layout/low-layout.json'].get_dict())
```

This works because the telescope model comes with a number of default sources that will be checked for matching telescope model data. *ska_telmodel.data.TMObject.get_dict()* automatically parses and converts JSON and YAML documents, but you can also get the raw data using *ska_telmodel.data.TMObject.get()*, or open or copy the contents as a file using *ska_telmodel.data.TMObject.open()* or *ska_telmodel.data.TMObject.copy()* respectively.

To get an idea what is contained in a particumlar part of the telescope model data tree, simply iterate over it (equivalent to *ska_telmodel.cli.cmd_ls()*):

```
from ska_telmodel.data import TMData
tmdata = TMData()
for key in tmdata['instrument']:
    print(key)
```

Logically, the `[]` operator constructs a sub-set of all telescope model data. If the key is a valid object name (i.e. has an extension, so contains a `'.'`) this subset is assumed to contain only a single object, and the `[]` operator will return a *ska_telmodel.data.TMObject* instance.

1.4.2 Data Sources

Telescope model data can be retrieved from a list of sources, which can be specified to the *ska_telmodel.data.TMData* constructor, using the SKA_TELMODEL_SOURCES environment variable or left to in-built ska_telmodel.data.sources.DEFAULT_SOURCES. Each source is represented as an URI that specifies the source of truth for some portion of telescope model data.

The following telescope model data backends are currently supported:

- mem://?[key1]=[value1]&[key2]=[value2] (see *ska_telmodel.data.backend.MemoryBackend*)
- file://[absolute path] (see *ska_telmodel.data.backend.FilesystemBackend*)
- gitlab://[gitlab server]/[project name]?[branch]#[directory] (see *ska_telmodel.data.backend.GitlabBackend*)
- car://[gitlab server]/[project name]?[branch]#[directory] (see *ska_telmodel.data.backend.CARBackend*)

The simplest example would be to utilise *ska_telmodel.data.backend.MemoryBackend* to set a key directly:

```
from ska_telmodel.data import TMData
tmdata = TMData(['mem://?test.txt=test_data'])
print(tmdata['test.txt']) # -> b"test_data\n"
```

This can also be configured using environment variables:

```
import os
from ska_telmodel.data import TMData
os.environ['SKA_TELMODEL_SOURCES'] = 'mem://?test.txt=test_data'
tmdata = TMData()
print(tmdata['test.txt']) # -> b"test_data\n"
```

You would typically do this from outside your program, see the documentation for *ska_telmodel.cli.cmd_pin()* and *ska_telmodel.cli.cmd_cp()* for examples.

A more complex example would be to retrieve data from Gitlab using *ska_telmodel.data.backend.GitlabBackend*:

```
from ska_telmodel.data import TMData
gl_uri = 'car://gitlab.com/ska-telescope/ska-telmodel?master#tmdata'
tmdata = TMData([gl_uri])
print(tmdata['software/tango/dsh/DishManager.yaml'])
```

This will retrieve data directly from the telescope model library repository.

Note that external telescope model data sources using *ska_telmodel.data.backend.GitlabBackend* or *ska_telmodel.data.backend.CARBackend* will cache data locally in order to prevent repeated requests to servers. This means that if we reference a Gitlab branch (like master in the example), the telescope model data in the cache might go out of sync with the server.

This is intentional, as it means that we provide a consistent view of telescope model data as long as possible. It is generally best to use "pinned" sources (see *ska_telmodel.cli.cmd_pin()*), but in day-to-day usage, you can simply use the -U flag as documented in *Command Line Usage* or (less preferably) the update option to *ska_telmodel.data.TMData* to occassionally refresh the cache as needed. The library will occassionally check for and warn about stale caches.

1.4.3 Adding a New Gitlab Data Source

If you want others to be able to view data in your GitLab repository using *ska_telmodel.data.backend.GitlabBackend* or *ska_telmodel.data.backend.CARBackend*, first you will need to place the data you wish to export in a top level dictionary in your repo named `tmdata`. For example:

```
/tmdata/instrument/mccs_configuration/config_file_low.json
/tmdata/instrument/mccs_configuration/config_file_mid.json
```

Important to note:

- Try to use a directory structure that is compatible with domains (see above) and is reasonably likely to remain stable.
- Currently only .json and .yaml files are accepted, and you should have schemas associated with them.

Next add telescope model data support to your top-level `Makefile` as documented in https://developer.skao.int/projects/ska-cicd-makefile/en/latest/README.html :

```
include .make/tmdata.mk
```

At this point you should be able to verify that `make tmdata-package` will result in both a `tmtree.json` and a `tmdata.tar.gz` file getting created in `build/tmdata`. Next add the packaging and publishing stage to your Gitlab pipeline by adding the following lines to the `.gitlab-ci.yml` file as documented in https://developer.skao.int/projects/templates-repository/en/latest/README.html :

```
- project: 'ska-telescope/templates-repository'
  file: 'gitlab-ci/includes/tmdata.gitlab-ci.yml'
```

Now once you merge these changes into the `main` branch, others will be able to access this data by specifying your repository as the source:

```
$ ska-telmodel --sources=car:mccs/ska-low-mccs?main ls
instrument/mccs_configuration/config_file_low.json
instrument/mccs_configuration/config_file_mid.json
```

Branches other than `main` will also work, just adjust the URL accordingly. However by default the GitLab pipeline will only upload the TMData package to the artefact repository on the main branch as well as tags.

If you want the data to be accessible without passing command line parameters, make a merge request to the ska-telmodel repository (https://gitlab.com/ska-telescope/ska-telmodel) that adds your repository address the `src/ska_telmodel/data/source.py` file. This makes your telescope model data available "globally":

```
$ ska_telmodel ls
[...]
instrument/mccs_configuration/config_file_low.json
instrument/mccs_configuration/config_file_mid.json
```

1.4.4 Schemas

Schemas check JSON-like objects for conformance, e.g. nested dictionaries containing primitives and lists. They especially have a JSON schema representation - though *ska_telmodel.schema.validate()* will generally implement more thorough checks.

All schemas are identified by an URI of the form:

```
https://schema.skao.int/ska-[subsystem]-[interface]/[major].[minor]
```

The entire URI should be lower-case alphanumerical. The [subsystem] identifies the leading party for maintaining the schema, and [interface] the concrete interface implemented. Depending on context, this might either be data produced or consumed by the sub-system in question.

Versioning should follow semantic versions: Changes in minor version indicate backwards-compatible changes such as adding new fields or otherwise introducing additional accepted schemas. Changes that break backwards compatibility should change the major version.

You can use the URIs with *ska_telmodel.schema.validate()* to validate data:

```
from ska_telmodel.data import TMData
from ska_telmodel.schema import validate

uri = "https://schema.skao.int/ska-telmodel-layout-location/0.0"
layout_dict = TMData()['instrument/ska1_low/layout/low-layout.json'].get_dict()
validate(uri, layout_dict)
```

Furthermore you can use *ska_telmodel.schema.example_by_uri()* to retrieve examples of certain schemas (which are replicated in the schema section of this documentation).

1.5 Command Line Usage

The library provides the ska-telmodel command line utility that can be used to perform basic data retrieval and validation tasks. Usage examples:

```
ska-telmodel [-vULS<uris>] cat [<key>]
ska-telmodel [-vULS<uris>] cp [-R] <key> [<path>]
ska-telmodel [-vULS<uris>] ls [<prefix>]
ska-telmodel [-vULS<uris>] pin
ska-telmodel [-vULS<uris>] validate [-tR] <key/path>
ska-telmodel help [<command>]

Options:
  -L, --local        Equivalent to "--sources=file://."
  -R, --recursive    Copy / validate keys or files recursively
  -S <uris>, --sources <uris>
                     Set telescope model data sources of truth
                     (','-separated list of URIs)
  -t, --strict       Strict validation mode
  -U, --update       Update source list
  -v, --verbose      Verbose mode
```

See *Data Sources* for explanations about telescope model data sources.

ska_telmodel.cli.**cmd_cat**(*args*, *data*)

Retrieves and prints the telescope model data identifed by the given key to stdout.

Usage:

```
ska-telmodel [-vUs<uris>] cat [<key>]
```

Use `ska-telmodel ls` to obtain a list of valid keys.

How exactly the object is retrieved depends on the backend and the state of the cache. For a GitLab backend, the typical behaviour is to download a tarball either from the SKAO central artefact repository, or from GitLab directly. The latter should be avoided and will generate a warning.

ska_telmodel.cli.**cmd_cp**(*args*, *srcs*)

Retrieves specified telescope model data, and copies it to the given path.

Usage:

```
ska-telmodel [-vUs<uris>] cp [-R] <key> [<path>]
```

If `-R` is given, the key can be a key directory, in which all keys that start with `<path>/` will be copied. Note that you can especially give the empty string (`""`) as `<key>`, in which case all available telescope model data will be copied.

This is especially useful for serving telescope model data either partially or completely from storage. For instance:

```
$ ska-telmodel cp -UR "" tmdata
$ export SKA_TELMODEL_SOURCES=file://$(pwd)/tmdata
```

Would completely mirror the telescope model to the given location.

ska_telmodel.cli.**cmd_ls**(*args*, *data*)

List telescope model keys with a particular prefix

Usage:

```
ska-telmodel [-vUs<uris>] ls [<prefix>]
```

ska_telmodel.cli.**cmd_pin**(*args*, *data*)

Generates a "pinned" telescope model data source list, where all URIs replaced such that they will uniquely identify the contents of the telescope model data repository.

Usage:

```
ska-telmodel [-vUs<uris>] pin
```

After pinning, the source list precisely identifies the contents of the all telescope model data. For instance, this will replace GitLab URIs like `gitlab://gitlab.com/grp/proj#path` with `gitlab://gitlab.com/grp/proj?[commit]#path`, therefore baking in the exact commit referenced. You can set pinned sources in the environment as follows:

```
$ export $(ska-telmodel pin -U)
$ export $(ska-telmodel pin -US [custom sources])
```

This will especially prevent the ska-telmodel tool from infrequently (once a day) re-checking whether cached telescope model data contents is still current. The `-U` flag forces the cache refresh, which is generally a good idea before pinning.

ska_telmodel.cli.**cmd_validate**(*args*, *srcs*)

Validates given keys (or files) against applicable schemas from the telescope model library

Usage:

```
ska-telmodel [-vUs<uris>] validate [-tlR] [<key/path>]*
```

If -R is given, the key can be a key directory, in which all keys that start with <path>/ will be copied. Note that you can especially give the empty string ("") as <key>, in which case all available telescope model data will be copied.

This is especially useful for serving telescope model data either partially or completely from storage. For instance:

```
$ ska-telmodel cp -R "" tmdata
$ export SKA_TELMODEL_SOURCES=file://$(pwd)/tmdata
```

Would completely mirror the telescope model to the given location.

1.6 Schema Development

The Telescope Model is developed jointly by all teams working on the SKA telescope. To make this work, all changes will have to be tested thoroughly and pass a code review via merge request.

Testing should ensure that all code paths are checked, i.e. we want to reach 100% coverage. We also aim to minimise regressions of any kind. This means that most code and data should be versioned *within* the Telescope Model, with old behaviour staying supported until a sufficient depreciation period has passed.

1.6.1 Adding a new schema (version)

To add a new interface, you will have to adjust a number of places in the library. For a new SKA interface <interface> with <elem> as the leading sub-system, do the following steps:

1. Add this:

   ```
   <ELEM>_<INTERFACE>_PREFIX = "https://schema.skao.int/ska-<elem>-<interface>/"`
   ```

   to src/ska_telmodel/<elem>/version.py. This is the interface namespace URI.

2. Add a get_<elem>_<interface>_schema(version: str, strict: bool) function to src/ska_telmodel/<elem>/schema.py, returning an appropriate Schema object. Consult https://pypi.org/project/schema/ for how to write such schemas. Please add documentation as far as possible, this will be put both into the JSON schema as well as the documentation.

3. Adjust schema_by_uri in src/ska_telmodel/schema.py to call get_<elem>_<interface>_schema for schemas starting with <ELEM>_<INTERFACE>_PREFIX so that your schema can be found.

4. Add a documentation file docs/src/ska_<elem>_<interface>.rst with a line along the lines of

   ```
   .. ska-schema:: https://schema.skao.int/ska-<elem>-<interface>/<ver>
   ```

   to ensure documentation is generated

If you just want to add new schema version, skip steps (1) and (3) and extend existing definitions in the remaining steps.

1.6.2 Adding a new example

It is a good idea to always provide an up-to-date example for every schema version. Assuming the schema is defined, the steps are fairly similar:

1. Add a `get_<elem>_<interface>_example(version :str)` function to `src/ska_telmodel/<elem>/examples.py`, returning an `dict`. If you have multiple examples, you can add a `str` parameter to select the appropriate one.
2. Adjust `example_by_uri` in `src/ska_telmodel/schema.py` to call `get_<elem>_<interface>_example` for schemas starting with `<ELEM>_<INTERFACE>_PREFIX` so that your example can be found.
3. Add your example to `docs/src/ska_<elem>_<interface>.rst` by adding a line like

   ```
   .. ska-schema-example:: https://schema.skao.int/ska-<elem>-<interface>/<ver>
   ```

   inside the `.. ska-schema` block of the appropriate version

1.6.3 Last steps

1. Import the newly added `<ELEM>_<INTERFACE>_PREFIX` from version,`get_<elem>_<interface>_schema`from schema and `get_<elem>_<interface>_example` from examples into `src/ska_telmodel/<elem>/__init__.py` file.
2. Finally add tests in `test_<elem>_schemas.py` to ensure test coverage. This is especially easy if you add an example to the schema (see above sub-section).

1.6.4 Code Style

This project uses automated code formatting using the Black Code Formatter, isort as well as custom bowler refactoring rules .

To ensure that all code is formatted as required, run the following before you commit:

```
$ pip install black isort bowler # if needed
$ make python-format
```

1.7 Semantic Validation

1.7.1 Semantic vs Syntactic validations

Semantic validation and syntactic validation are two types of validation techniques used in software development to ensure that data entered into a system is accurate and conforms to the requirements of the system.

Syntactic validation checks the syntax of the input data and ensures that it adheres to the prescribed format. It checks whether the data entered is structured correctly and follows the expected syntax rules. For example, if an input field is supposed to accept only numerical data, a syntactic validation would ensure that only numerical characters are entered and reject any non-numeric characters.

Semantic validation, on the other hand, checks the meaning of the input data and ensures that it is valid in the context of the system. It checks whether the input data conforms to the business rules and logic of the system.

For example, if a system requires a date to be entered, a semantic validation would ensure that the date entered is valid, such as it's not a future date or a date that has already passed.

In summary, syntactic validation checks the structure of the data, while semantic validation checks the meaning of the data. Both types of validation are important to ensure the accuracy and integrity of data entered into a system.

1.7.2 Introduction

Here we have created 'Framework for semantic validation of observing setups'. This framework provides semantic validation which helps to prevent the users from making errors in their setups. This framework is supporting both MID and LOW schema validation.

For creating this framework there are some requirements and architecture have already provided. These are as follows:

- Configuration Schemas (Mid)
- Configuration Schemas (Low)
- Semantic Validation architecture AA0.5

1.7.3 JSON validator file

Two seperate JSON files have been created for Mid & Low schemas to store all the parameters present in assign & configure resources along with its business rules and errors.

- Reference of JSON validator file (Mid)
- Reference of JSON validator file (Low)

Created a seperate `constant` file to maintain all telvalidation constant. From there we are importing JSON validator file in `semantic_validator` for both Mid & Low schemas.

Below are the commands to import JSON validator files.

```
from ska_telmodel.data import TMData

from .constant import (
    LOW_VALIDATION_CONSTANT_JSON_FILE_PATH,
    MID_VALIDATION_CONSTANT_JSON_FILE_PATH,
)

# For Mid
semantic_validate_data = tm_data[
        MID_VALIDATION_CONSTANT_JSON_FILE_PATH
    ].get_dict()

# For Low
semantic_validate_data = tm_data[
        LOW_VALIDATION_CONSTANT_JSON_FILE_PATH
    ].get_dict()
```

1.7.4 Adding a new parameter in JSON validator file

Steps to add a new parameter in JSON validator file

- **Locate the appropriate place in the JSON structure:**
  - Identify the parent key or object where the new parameter should be added.
  - Determine the desired position for the new parameter within the parent key's object.
- **Add a new key-value pair representing the parameter:**
  - Structure of parameter should be parent-child.
  - Specify the name of the parameter as the key, this key represents the parent_key and it should contain dictionary.
  - Add additional key-value pairs within the parent_key object for the rule and error message. In this you can specify the business rule & error message to validate the specific key.

Example

If a user wants to add any new parameter in JSON validator file so he can take reference of this example:

```
"scan": {
        "tmc": {
            "scan_id": [
                {
                    "rule": "scan_id == 1",
                    "error": "Invalid input for scan_id"
                }
            ]
        }
    },
```

Let's take scan command as a dummy key which is currently not present in the JSON file.

Here under scan there is a dictionary which has a key named "tmc" so scan.tmc will be the parent_key and under tmc we have a "scan_id" child key containing a list which should contain appropriate rules and error messages.

1.7.5 General structure

This framework has created very dynamically and user friendly. If user wants to access this framework from CDM or Jupyter Notebook then he just has to import telvalidation package from import statement and call `semantic_validate` function and pass the appropriate parameters to this function. If validation fails then the end user will get the list of errors.

This framework can be access by below command:

```
from ska_telmodel.telvalidation.semantic_validator import semantic_validate
```

- Location of this framework

There are some steps of this framework these are as follows:

- **Step 1** It checks the parameter in the JSON validator document which is present in tmdata package.
- **Step 2** There is a `validate_json` function which takes two parameters JSON file & config as a dictionary. It is present in `src/ska_telmodel/telvalidation/oet_tmc_validators`. Here we are using an eval term to evaluate the business rules present in the JSON file and based on that it raises custom errors. All the

custom errors are stored in a list named `error_msg_list`. At the end this function returns a list containing all the error messages.

`ska_telmodel.telvalidation.oet_tmc_validators.`**`validate_json`**(*semantic_validate_constant_json: dict, command_input_json_config: dict, error_msg_list: list, parent_key: str*) → list

Parameters

- **semantic_validate_constant_json** – json containing all the parameters along with its business semantic validation rules and error message.
- **command_input_json_config** – dictionary containing details of the command input which needs validation. This is same as for ska_telmodel.schema.validate.
- **parent_key** – temp key to store parent key, means if same semantic validation key present in 2 places this will help to identify correct parent.

Returns error_msg_list: list containing all combined error which arises due to semantic validation.

- **Step 3** There is one more function `semantic_validate` which takes mainly three argument config, interface and raise_semantic. It is present in `src/ska_telmodel/telvalidation/schema`.

  This function first checks for the interface, if the interface is not present then a warning message is logged, indicating that the `interface` is missing from the config. Additionally, a SchematicValidationError exception is raised with the same message.

  This framework allowed interface only for two commands that are `assignresources` & `configure`. If a user provides an incorrect or unsupported interface value, for example if user passes the interface for the scan command, the code will not be able to find a matching validation schema based on that interface. As a result, the `validate_json` function will not be called, and the `msg_list` variable will remain empty.

  Also this function is not supporting low telescope schema validation currently.

`ska_telmodel.telvalidation.semantic_validator.`**`semantic_validate`**(*config: dict, tm_data: ska_telmodel.data.frontend.TMData, interface: Optional[str] = None, raise_semantic: bool = True*)

Parameters

- **config** – dictionary containing details of the command which needs validation. This is same as for ska_telmodel.schema.validate. If command available as json string first convert to dictionary by json.loads.
- **tm_data** – telemodel tm data object using which we can load semantic validate json.
- **interface** – interface uri in full only provide if missing in config
- **raise_semantic** – True(default) would need user to catch somewhere the SchematicValidationError. Set False to only log the error messages.

Returns msg: if semantic validation fail returns error message containing all combined error which arises else returns True.

1.7.6 Target visibility validation

There are ra and dec parameters in configure resources, to validate these parameters we have created a separate module named `coordinates_conversion` which converts Right Ascension and Declination to Azimuth and Altitude. This module contains a function `ra_dec_to_az_el` which has logic for this conversion. This function has been imported in the `validate_target_is_visible` function which is present in the `oet_tmc_validators` module.

`ska_telmodel.telvalidation.oet_tmc_validators.`**`validate_target_is_visible`**(*ra_str: str*, *dec_str: str*, *telescope: str*, *target_env: str*, *tm_data*, *observing_time: datetime.datetime = datetime.datetime(2023, 11, 22, 9, 17, 42, 553153)*) → str

Check the target specific by ra,dec is visible during observing_time at telescope site

Parameters

- **`ra_str`** – string containing value of ra
- **`dec_str`** – string containing value of dec
- **`telescope`** – string containing name of the telescope
- **`observing_time`** – string containing value of observing_time
- **`target_env`** – string containing the environment value(mid/low) for the target
- **`tm_data`** – telemodel tm dataobject using which we can load semantic validate json.

This is the main function for conversion.

`ska_telmodel.telvalidation.coordinates_conversion.`**`ra_dec_to_az_el`**(*telesc: str*, *ra: float*, *dec: float*, *obs_time: str*, *el_limit: float*, *tm_data: ska_telmodel.data.frontend.TMData*, *time_format: str = 'iso'*, *if_set: bool = False*, *time_scale: str = 'utc'*, *coord_frame: str = 'icrs'*, *prec: float = 0.0001*, *max_iter: int = 200*) → list

Returns the az el in degrees from ra dec at given time for the telescopes [az el info_isvisible]

Index 0 azimuth in degrees

Index 1 elevation in degrees

Index 2 info_isvisible is True if src visible above/at el_limit given time else False

Parameters

- **`telesc`** – “mid” for Mid or “low” for Low Telescope
- **`ra`** – Right ascension in degrees with decimal places for arc min,arc sec also covert to degrees. Eg 123d30’ input 123.5 . In case of RA in hh mm sec please also convert to degrees.
- **`dec`** – Declination in degrees with decimal places.

- **obs_time** – str containing time when source position in terms of azimuth, elevation should be calculated. Eg ‘2023-04-18 20:12:18’
- **time_format** – str to choose from available Time.FORMATS. Default “iso”
- **time_scale** – str to choose from available Time.SCALES Default “utc”
- **coord_frame** – str to choose from available Astronomical Coordinate Systems
- **el_limit** – float specifying elevation in degree below which our telescope cannot observe the source
- **prec** – float for precision limit in degrees to match elevation with given el_limit. default: 0.0001 degrees i.e. <1 arcsecond
- **max_iter** – int to specify upto how many iterations can root finder use before it stops or reaches required precision. Default is 200. Only set higher if suggested by message. There is also a seperate message if it is determined that root finder is not able to converge starting from given time
- **tm_data** – telemodel tm data object using which we can load semantic validate json.

1.8 API reference

1.8.1 ska_telmodel.data

class ska_telmodel.data.**TMData**(*source_uris: Optional[list[str]] = None*, *prefix: str = ''*, *update: bool = False*, *backend_pars: dict = {}*)

Represents a tree of telescope model data.

Data is retrieved from specified `sources` (or using default sources if not passed). Depending on backend, this might cause data to be loaded from remote locations, such as the SKAO central artefact repository or Gitlab.

Objects of this class provide a hierarchical `dict`/`h5py`-like interface. For instance, you can print all objects with keys starting with `instrument/layout` as follows:

```
layouts = tmdata['instrument/layout']
for key in layouts:
   print(f"Data for {key}: ", layouts[key].get())
```

This works because `__getitem__()` will redirect to *get_subtree()* or *get()* depending on whether a valid key is passed (i.e. it has an extension). The *TMObject* object can then be used to access the underlying telescope model data.

Parameters

- **source_uris** – List of telescope model data sources. If not passed, defaults to `SKA_TELMODEL_SOURCES` enviroment variable, then in-built `DEFAULT_SOURCES`.
- **prefix** – Key prefix for sub-tree selection
- **update** – Update cached data sources (if any)
- **backend_pars** – Extra parameters to specific backend (types)

get(*key: str*) → *ska_telmodel.data.frontend.TMObject*

Returns the telescope model object with the given key

Parameters **key** – Key to retrive. Must be a valid telescope model key (i.e. have a file type extension)

Returns *TMObject* object

Raises `KeyError` if object doesn't exist

get_sources(*pinned: bool = False*) → list[str]

Returns list of source URIs

Parameters **pinned** – Attempt to return URIs that will continue to refer to this specific version of telescope model data. E.g. for GitLab URIs, this replaces tags or branches by the concrete commit hash.

Returns list of sources

get_subtree(*prefix: str*) → *ska_telmodel.data.frontend.TMData*

Returns clone of *TMData* object with given prefix

Note that no checking is done whether any keys with the given prefix exist.

Parameters **prefix** – Prefix to narrow scope to. Must be a valid telescope model prefix

Returns *TMData* object using prefix

class ska_telmodel.data.**TMObject**(*source: ska_telmodel.data.backend.TMDataBackend, key: str*)

Represents a telescope model data object. Provides a number of ways to access the data.

Parameters

- **source** – Backend to use to retrieve object data
- **key** – Key associated with object

copy(*dest: str*)

Copy object data to a file.

Parameters **dest** – Path of destination file

get() → bytes

Access data at given key as raw bytes

Returns Raw object data

get_dict(***kwargs*) → dict

Access object as a dictionary

Will only work if the key ends with a known extension – e.g. `.json` or `.yaml`.

Parameters **kwargs** – Extra parameters to `[json/yaml].load`

Returns Parsed dictionary

open() → IO[bytes]

Access object data as a read-only file object

Parameters **key** – Key to query

Returns File-like object

1.8.2 ska_telmodel.schema

Support for validating and generating examples for SKA telescope model schemas.

`class ska_telmodel.schema.SchemaUri(version: str)`

Convenience class for manipulating version URIs.

Parameters `version` – Interface URI

`property major_minor: Tuple[int, int]`

Get the major and minor parts of the version.

Returns tuple of major and minor versions

`property prefix: str`

Get the prefix.

Returns prefix

`property version: str`

Get the version.

Returns version

`ska_telmodel.schema.example_by_uri(version: str, *args) → dict`

Generates an example for a particular schema

Parameters

- `version` – Interface URI
- `args` – Extra parameters depending on interface (strings)

Returns Dictionary

`ska_telmodel.schema.schema_by_uri(version: str, strict: int = 1, **kwargs) → schema.Schema`

Looks up interface schema based on interface identifier

Parameters

- `version` – Interface URI
- `strict` – Strictness level

Returns Interface schema

`ska_telmodel.schema.validate(version: Optional[str], config: dict, strictness: int = 1)`

Validate a dictionary against schema

Will automatically determine the schema to check against

Parameters

- `version` – Interface with version
- `config` – Dictionary to validate
- `strictness` – Strictness level (0: permissive warnings, 1: permissive errors + strict warnings, 2: strict errors).

  Note that with strictness level 2, a lot of generally harmless schema violations will cause an exception to be raised. This is generally inadvisable in production consumer code ("be liberal in what you accept"!).

Raises `SchemaError` – Raised if the object fails permissive checks at strictness level 1. At strictness level 2, raised if the object fails any schema check.

1.9 Internals

1.9.1 ska_telmodel._common

class ska_telmodel._common.**TMSchema**(*schema: Optional[Any] = None, error=None, ignore_extra_keys: bool = False, name: Optional[str] = None, description: Optional[str] = None, as_reference: bool = False, version: Optional[str] = None, strict: bool = False*)

Wrapper on top of schema.Schema for incremental schema build-up.

ska_telmodel._common.**get_unique_id_schema**(*strict: bool, type_re: str = '[a-z0-9]+'*) → schema.Schema

Return schema for unique identifier.

Parameters `type_re` – Restricts ID type(s) to accept.

ska_telmodel._common.**interface_uri**(*prefix: str, *versions: int*) → str

Make an URI from the given prefix and versions

Parameters

- `prefix` – Schema URI prefix. Must end in '/'
- `versions` – Components of the version

ska_telmodel._common.**mk_if**(*cond: bool*) → Callable[[Any], Any]

Generate schema combinator to conditionally activate a part.

ska_telmodel._common.**split_interface_version**(*version: str*) → Tuple[int, int]

Extracts version number from interface URI

Parameters `version` – Version string

Returns (major version, minor version) tuple

1.9.2 ska_telmodel.channel_map

Tools for working with JSON compressed channel maps.

The SKA is meant to have a large number of channels, which means that any type of per-channel configuration might become very cumbersome to transfer and reason about. To prevent such issues we are using a simple run-length encoding to "compress" the representation. The idea is that if we write:

```
[ [0,0], [200,1], [400, 3] ]
```

We essentially mean the dictionary:

```
{ 0: 0, 1: 0, ..., 199:0, 200:1, ..., 399:1, 400: 3, ...}
```

Furthermore runs of numbers are supported, by adding an increment:

```
[ [0,0,1], [200,1] ]
```

Means:

```
{ 0: 0, 1: 1, 2:2, ..., 199:100, 200:1, ...}
```

ska_telmodel.channel_map.**channel_map_at**(*channel_map: List[list]*, *channel: int*, *make_entry: bool = False*) → Any

Query a value from a channel map

Parameters

- **channel_map** – Queried map
- **channel** – Channel ID to query
- **make_entry** – Return an channel map entry (including increment) instead of just the value

Returns Value from map

ska_telmodel.channel_map.**shift_channel_map**(*channel_map: List[list]*, *channel_shift: int*) → List[list]

Shift a channel map by some channel distance

Parameters

- **channel_map** – Channel map to use
- **channel_shift** – Shift to apply

ska_telmodel.channel_map.**split_channel_map**(*channel_map: List[list]*, *first_channel: int*, *channel_group_steps: int*, *rebase_groups: Optional[int] = None*, *minimum_groups: int = 0*) → List[List[list]]

Split a channel map using a constant channel step length

Parameters

- **channel_map** – Channel map to split. Each entry is expected to have the start channel in the first field, and mapped data in the remaining entries
- **first_channel** – First channel to appear in the map
- **channel_group_steps** – Chunks to split the channel map into
- **rebase_groups** – Start every group at given channel index (None: left as-is)
- **minimum_groups** – Minimum number of groups to return

Returns List of channel maps

ska_telmodel.channel_map.**split_channel_map_at**(*channel_map: List[list]*, *channel_groups: List[int]*, *rebase_groups: Optional[int] = None*) → List[List[list]]

Split a channel map at certain points

Parameters

- **channel_map** – Channel map to split. Each entry is expected to have the start channel in the first field, and mapped data in the remaining entries
- **channel_groups** – Boundaries between channel groups. The *n*-th returned channel map will cover channels *channel_groups[n]..channel_groups[n+1]-1*. Needs to have at least two entries.
- **rebase_groups** – Start every group at given channel index (None: left as-is)

Returns List of channel maps

1.9.3 ska_telmodel.data

class ska_telmodel.data.backend.**CARBackend**(*uri: str*, **args*, ***kwargs*)

Represents data in (a mirror of) the SKA central artefact repository. Permissible URI formats:

```
car:[project name]?[branch]#[directory]
car://[gitlab server]/[project name]?[branch]#[directory]
```

So for instance:

```
car:ska-telmodel?master
car://gitlab.com/ska-telescope/ska-telmodel?master#tmdata
```

The source of truth might still be Gitlab, yet this backend will only work with artefacts that have been uploaded to the CAR. The short form URI will be expanded into the long form automatically.

classmethod backend_name() → str

Returns the name of the backend.

Will be used for the scheme in URIs to identify the backend type of a telescope model data source.

get_uri(*pinned: bool*) → str

Returns URI for this telescope model data backend

Parameters **pinned** – Attempt to return an URI that will continue to refer to this specific version of telescope model data

Returns URI identifying data source

class ska_telmodel.data.backend.**FilesystemBackend**(*uri: str*, *update: bool = False*)

Retrives data from a locally accessible file system. URI format:

```
file://[absolute path]
```

Note that changes to the file system are outside of our control. Consistency must be ensured externally.

classmethod backend_name() → str

Returns the name of the backend.

Will be used for the scheme in URIs to identify the backend type of a telescope model data source.

copy(*key: str*, *dest: str*)

Write key contents to a file.

Raises *KeyError* if the key does not exist

Parameters

- **key** – Key to query
- **dest** – Path of destination file

exists(*key: str*) → bool

Check whether a given key exists.

Parameters **key** – Key to query

Returns True if key exists

get(*key: str*) → bytes

Get the data stored with the given key

Parameters **key** – Key to query

Returns Bytes stored at key

get_uri(*pinned: bool*) → str

Returns URI for this telescope model data backend

Parameters **pinned** – Attempt to return an URI that will continue to refer to this specific version of telescope model data

Returns URI identifying data source

list_keys(*key_prefix: str = ''*) → collections.abc.Iterable[str]

List children keys

Yields all keys with prefix "{key_prefix}/" in ascending order

Parameters **key_prefix** – Path to query

open(*key: str, binary: bool = True*) → IO[bytes]

Access data at given key as a file-like object

Raises *KeyError* if the key does not exist

Parameters **key** – Key to query

class ska_telmodel.data.backend.GitlabBackend(*uri: str, update: bool = False, gl: gitlab.Gitlab = None, try_nexus: bool = True, nexus_url: str = None, env=None*)

Represents data in a GitLab repository. URI format:

```
gitlab://[gitlab server]/[project name]?[branch]#[directory]
```

So for instance:

```
gitlab://gitlab.com/ska-telescope/ska-telmodel?master#tmdata
```

Would refer to data contained in the `ska-telmodel` repository itself.

Repositories accessed in this way should make sure to activate the `tmdata` standard continuous integration stages (see https://gitlab.com/ska-telescope/templates-repository) to ensure that telescope model data is cached in the SKAO central artefact repository. Once that has been done, this library will never actually query GitLab directly.

Furthermore, this backend will cache all loaded data locally, including resolved Gitlab references (like `master` in the example above). This especially means that once instantiated, the version of data will be "pinned" even between different instances (and processes). Use the `update` parameter to *ska_telmodel.data.TMData* or *GitlabBackend* respectively to refresh the local cache.

classmethod backend_name() → str

Returns the name of the backend.

Will be used for the scheme in URIs to identify the backend type of a telescope model data source.

copy(*key: str, dest: str*)

Write key contents to a file.

Raises *KeyError* if the key does not exist

Parameters

- **key** – Key to query
- **dest** – Path of destination file

exists(*key: str*) → bytes

Check whether a given key exists.

Parameters **key** – Key to query

Returns True if key exists

get(*key: str*) → bytes

Get the data stored with the given key

Parameters **key** – Key to query

Returns Data stored at key, or None if it doesn't exist

get_uri(*pinned: bool*) → str

Returns URI for this telescope model data backend

Parameters **pinned** – Attempt to return an URI that will continue to refer to this specific version of telescope model data

Returns URI identifying data source

list_keys(*key_prefix: str = ''*) → collections.abc.Iterable[str]

List children keys

Yields all keys with prefix "{key_prefix}/" in ascending order. Exception is if the path is empty, in which case all available keys are listed.

Parameters **key_prefix** – Path to query

open(*key: str*) → IO[bytes]

Access data at given key as a file-like object

Raises *KeyError* if the key does not exist

Parameters **key** – Key to query

class ska_telmodel.data.backend.**MemoryBackend**(*uri: str*, *update: bool = False*)

Represents in-memory data. URIs should look as follows:

```
mem://?[key1]=[value1]&[key2]=[value2]
```

This will directly set the given telescope model data keys to the given values. Useful for testing, and overriding single values in telescope model data.

classmethod backend_name() → str

Returns the name of the backend.

Will be used for the scheme in URIs to identify the backend type of a telescope model data source.

get(*key: str*) → bytes

Get the data stored with the given key

Parameters **key** – Key to query

Returns Bytes stored at key

get_uri(*pinned: bool*) → str

Returns URI for this telescope model data backend

Parameters **pinned** – Attempt to return an URI that will continue to refer to this specific version of telescope model data

Returns URI identifying data source

list_keys(*key_prefix: str* = '') → collections.abc.Iterable[str]

List children keys

Yields all keys with prefix "{key_prefix}/" in ascending order

Parameters **key_prefix** – Path to query

class ska_telmodel.data.backend.**TMDataBackend**(*uri: str*, *update: bool* = *False*)

Base class for telescope model data backends

Sub-classes should override `backend_name()`, then utilise `telmodel_backend()` to register the telescope model data backend. A minimal implementation should furthermore provide `list_keys()` and `get()`.

abstract classmethod backend_name() → str

Returns the name of the backend.

Will be used for the scheme in URIs to identify the backend type of a telescope model data source.

copy(*key: str*, *dest: str*)

Write key contents to a file.

Raises *KeyError* if the key does not exist

Parameters

- **key** – Key to query
- **dest** – Path of destination file

exists(*key: str*) → bool

Check whether a given key exists.

Parameters **key** – Key to query

Returns True if key exists

abstract get(*key: str*) → bytes

Get the data stored with the given key

Parameters **key** – Key to query

Returns Data stored at key, or None if it doesn't exist

get_uri(*pinned: bool*) → str

Returns URI for this telescope model data backend

Parameters **pinned** – Attempt to return an URI that will continue to refer to this specific version of telescope model data

Returns URI identifying data source

abstract list_keys(*key_prefix: str* = '') → collections.abc.Iterable[str]

List children keys

Yields all keys with prefix "{key_prefix}/" in ascending order. Exception is if the path is empty, in which case all available keys are listed.

Parameters **key_prefix** – Path to query

open(*key: str*) → IO[bytes]

Access data at given key as a file-like object

Raises *KeyError* if the key does not exist

Parameters **key** – Key to query

classmethod valid_key(*key: str*) → bool

Check whether this is a valid key we could store data for

For this to be valid, it needs to: * Have every path segment start with a letter * Have no dot in directory names, and a dot in file name

Returns Validity of key

classmethod valid_prefix(*key: str*) → bool

Check whether argument could be a valid prefix to a key

For this to be valid, it needs to: * Have every path segment start with a letter * Have no dot in directory names, and a dot in file name

Returns Validity of key

1.9.4 ska_telmodel.csp

ska_telmodel.csp.config.**add_midcbf_visibility_receive_addresses**(*csp_config: dict*, *scan_receive_addrs: dict*, *csp_interface_version: str*, *sdp_interface_version: str*) → dict

Add SDP visibility receive addresses into mid-cbf configuration

Parameters

- **csp_config** – CSP input configuration
- **scan_receive_addrs** – SDP receive addresses for scan
- **csp_interface_version** – CSP interface version to assume
- **sdp_interface_version** – SDP interface version to assume

Returns New CSP configuration

ska_telmodel.csp.config.**add_pss_receive_addresses**(*csp_config: dict*, *scan_receive_addrs: dict*, *csp_interface_version: str*, *sdp_interface_version: str*) → dict

Add SDP visibility receive addresses into pulsar search configuration

Parameters

- **csp_config** – CSP input configuration
- **scan_receive_addrs** – SDP receive addresses for scan
- **csp_interface_version** – CSP interface version to assume
- **sdp_interface_version** – SDP interface version to assume

Returns New CSP configuration

ska_telmodel.csp.config.**add_pst_receive_addresses**(*csp_config: dict, scan_receive_addrs: dict, csp_interface_version: str, sdp_interface_version: str*) → dict

Add SDP visibility receive addresses into pulsar timing configuration

Parameters

- **scan_trype** – Scan type executed
- **csp_config** – CSP input configuration
- **sdp_receive_addrs** – SDP receive addresses for scan

Returns New CSP configuration

ska_telmodel.csp.config.**add_receive_addresses**(*scan_type: str, csp_config: dict, scan_receive_addrs: dict, csp_interface_version: str, sdp_interface_version: str*) → dict

Add SDP visibility receive addresses into CSP configuration

Parameters

- **csp_config** – CSP input configuration
- **scan_receive_addrs** – SDP receive addresses for scan
- **csp_interface_version** – CSP interface version to assume
- **sdp_interface_version** – SDP interface version to assume

Returns New CSP configuration

ska_telmodel.csp.config.**get_fsp_channel_offset**(*csp_config_in: dict*) → int

Determines first channel ID within an FSP

ska_telmodel.csp.config.**get_fsp_output_channel_offset**(*fsp_config: dict, fsp_id: str, fsp_ch_offset: str*) → int

Determines the FSP channel offset. Either read from the dictionary or reconstructed.

Parameters

- **fsp_config** – FSP configuration structure
- **fsp_id** – Position of FSP in configuration
- **fsp_ch_offset** – Name of FSP channel offset field

ska_telmodel.csp.examples.**get_csp_assignresources_example**(*version: str*) → dict

Generate example of CSP assignresources argument

Parameters **version** – Version URI of configuration format

ska_telmodel.csp.examples.**get_csp_config_example**(*version: str, scan: Optional[str] = None*) → dict

Generate examples for CSP configuration strings

Parameters

- **version** – Version URI of configuration format
- **scan** – Includes SDP receive addresses for a scan? *None* means that this is "template" configuration as passed to TMC. Valid parameters: cal_a, science_a

`ska_telmodel.csp.examples.`**`get_csp_delaymodel_example`**`(`*version: str*`)` → dict

Generate example of CSP delay model argument

Parameters **`version`** – Version URI of configuration format

`ska_telmodel.csp.examples.`**`get_csp_endscan_example`**`(`*version: str*`)` → dict

Generate example of CSP endscan argument

Parameters **`version`** – Version URI of configuration format

`ska_telmodel.csp.examples.`**`get_csp_releaseresources_example`**`(`*version: str*`)` → dict

Generate example of CSP releaseresources argument

Parameters **`version`** – Version URI of configuration format

`ska_telmodel.csp.examples.`**`get_csp_scan_example`**`(`*version: str*`)` → dict

Generate example of CSP scan argument

Parameters **`version`** – Version URI of configuration format

csp.validators module defines constants and functions for validating CSP fields in schemas.

`ska_telmodel.csp.validators.`**`validate_integration_factor`**`(`*integration_factor: int*`)` → bool

Checks if the integration_factor is valid.

Parameters **`integration_factor`** – Integration time for correlation products

Returns True if integration_factor is valid

`ska_telmodel.csp.version.`**`check_csp_interface_version`**`(`*version: str, allowed_prefixes: Union[str, List[str]] = ['https://schema.skao.int/ska-csp-assignresources/', 'https://schema.skao.int/ska-csp-configure/', 'https://schema.skao.int/ska-csp-scan/', 'https://schema.skao.int/ska-csp-endscan/', 'https://schema.skao.int/ska-csp-releaseresources/', 'https://schema.skao.int/ska-csp-delaymodel/']*`)` → str

Check CSP interface version.

Checks that the interface URI has one of the allowed prefixes. If it does, the version number is returned. If not, a ValueError exception is raised.

Parameters

- **`version`** – CSP interface URI
- **`allowed_prefixes`** – allowed URI prefix(es)

Returns version number

`ska_telmodel.csp.version.`**`csp_config_versions`**`(`*min_ver=None, max_ver=None*`)`

Returns a list of CSP configuration interface version URIs

Parameters

- **`min_ver`** – Tuple of minimum version to return
- **`max_ver`** – Tuple of maximum version to return

ska_telmodel.csp.version.**normalize_csp_config_version**(*csp_interface_version: Union[int, str], csp_config: Optional[dict] = None*)

Provides a standard interface version for configure

Parameters

- **csp_interface_version** – External guess at the interface version
- **csp_config** – Example configuration to derive version from

Returns Canonical URI of interface version

1.9.5 ska_telmodel.sdp

Define schemas for SDP commands.

Miscellaneous schemas that probably should be moved somewhere else.

ska_telmodel.sdp.common.**ALL_RECEPTORS = Or(Regex('^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$'), Regex('^[ENS]([1-9]|1[0-6])-[1-6]$'), Regex('^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\\.\\S+)?$'), Regex('^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$'), Regex('^MKT0([0-5][0-9]|6[0-3])$'))**

All receptors.

ska_telmodel.sdp.common.**LOW_CORE = Regex('^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$')**

LOW core receptors, 1-224

ska_telmodel.sdp.common.**LOW_DIRS = Regex('^[ENS]([1-9]|1[0-6])-[1-6]$')**

LOW east/north/south receptors.

ska_telmodel.sdp.common.**LOW_FS = Regex('^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\\.\\S+)?$')**

LOW FS 1-512, plus optional substations.

ska_telmodel.sdp.common.**MID_MKT = Regex('^MKT0([0-5][0-9]|6[0-3])$')**

MID Meerkat, 000-063.

ska_telmodel.sdp.common.**MID_SKA = Regex('^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$')**

MID SKA, 001-133.

ska_telmodel.sdp.common.**get_beam_function_pattern**(*strict: bool*)

Get pattern for SDP beam functions

As used for SDP configuration - i.e. basically a kind of data that the SKA SDP needs to receive.

Returns A string pattern suitable for use in schemas

ska_telmodel.sdp.common.**get_receptor_schema**(*strict: bool*) → schema.Schema

Return schema for receptors.

Parameters **strict** – check names if set

Returns schema

`ska_telmodel.sdp.examples.`**`get_sdp_assignres_example`**(*version: Union[int, str]*) → dict

Generate example of SDP assign resources argument.

Parameters **`version`** – SDP assign resources version

Returns Example dictionary

`ska_telmodel.sdp.examples.`**`get_sdp_configure_example`**(*version: Union[int, str], scan_type: str = 'science'*) → dict

Generate example of SDP configure argument.

Parameters

- **`version`** – SDP configure version
- **`scan_type`** – Scan type to configure. "new_calibration" declares a new scan in-line.

Returns Example dictionary

`ska_telmodel.sdp.examples.`**`get_sdp_recvaddrs_example`**(*version: Union[int, str]*) → dict

Generate example of SDP receive addresses map.

Parameters **`version`** – SDP receive addresses version

Returns Example dictionary

`ska_telmodel.sdp.examples.`**`get_sdp_releaseres_example`**(*version: Union[int, str]*) → dict

Generate example of SDP release resources argument.

Parameters **`version`** – SDP release resources version

Returns Example dictionary

`ska_telmodel.sdp.examples.`**`get_sdp_scan_example`**(*version: Union[int, str], scan_id: int = 1*) → dict

Generate example of SDP scan argument.

Parameters

- **`version`** – SDP scan version
- **`scan_id`** – Scan ID to start

Returns Example dictionary

Define processing blocks schemas.

Defines receive addresses schema.

Used for checking SDP strings for conformance.

`ska_telmodel.sdp.version.`**`CALL_SIG`**

Call signature for schemas.

alias of `Callable[[Union[int, str], bool], schema.Schema]`

`ska_telmodel.sdp.version.`**`PREFIXES_TYPE`**

The type af allowed prefixes.

alias of `Union[str, Sequence[str]]`

`class` `ska_telmodel.sdp.version.`**`SchemaFactory`**(*prefix: Optional[str] = None, allowed_prefixes: Optional[Union[str, Sequence[str]]] = None*)

Get the right schema for a type based on its version.

get_schema(*version: ska_telmodel.sdp.version.SdpVersion*, *strict: bool*) → schema.Schema

Get the schema for this version.

If strict, an exact match is required. Otherwise, the last minor version matching the major version is accepted. It is assumed that a version is of the form version.subversion.

Parameters

- **version** – SDP version object
- **strict** – whether strict or not

Returns the matching schema

register(*version: str*, *func: Callable[[Union[int, str], bool], schema.Schema]*) → None

Register a function to create the schema.

Parameters

- **version** – the short version number (not the URI).
- **func** – function to create the schema

register_all(*versions: Iterable[str]*, *func: Callable[[Union[int, str], bool], schema.Schema]*) → None

Register a function to create the schema for multiple versions.

Parameters

- **versions** – iterable of short version numbers (not the URIs).
- **func** – function to create the schema

class ska_telmodel.sdp.version.**SdpVersion**(*version: Union[int, str]*, *prefix: Optional[str] = None*, *allowed_prefixes: Optional[Union[str, Sequence[str]]] = None*)

Wrapper around the normalise/check functions and stores the results.

Parameters

- **version** – SDP interface version
- **prefix** – schema prefix
- **allowed_prefixes** – allowed URI prefix(es)

Returns version object

ska_telmodel.sdp.version.**VERSION_TYPE**

The type of a version parameter.

alias of `Union[int, str]`

ska_telmodel.sdp.version.**check_sdp_interface_version**(*version: str*, *allowed_prefixes: Optional[Union[str, Sequence[str]]] = None*) → str

Check SDP interface version.

Checks that the interface URI has one of the allowed prefixes. If it does, the version number is returned. If not, a ValueError exception is raised.

Parameters

- **version** – SDP interface URI
- **allowed_prefixes** – allowed URI prefix(es)

Returns version number

`ska_telmodel.sdp.version.normalise_sdp_interface_version`(*version: Union[int, str], prefix: str*) → str

Normalise SDP interface version.

Converts deprecated integer version number into a schema URI, where the prefix specifies which schema to use. If the version is a string, it is assumed to be a schema URI and it is returned unchanged.

Parameters

- **version** – SDP interface version
- **prefix** – schema prefix

Returns SDP interface URI

`ska_telmodel.sdp.version.sdp_interface_versions`(*prefix: str, min_ver=None, max_ver=None*)

Returns a list of SDP interface version URIs

Parameters

- **prefix** – Interface URI prefix
- **min_ver** – Tuple of minimum version to return
- **max_ver** – Tuple of maximum version to return

1.10 Central Signal Processor schemas

Schemas used for commands for CSP LMC.

Some of these schemas are also used by Mid.CBF. See *Mid CBF schemas* for details.

1.10.1 ska-csp-assignresources

CSP assignresources 2.2

Example JSON


```
{
    "interface": "https://schema.skao.int/ska-csp-assignresources/2.2",
    "subarray_id": 1,
    "dish": {
        "receptor_ids": ["SKA001", "SKA036"]
    }
}
```

https://schema.skao.int/ska-csp-assignresources/2.2				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• **subarray_id**	The Subarray ID that the list of receptors will be assigned to. For Mid, there are a maximum of 16 subarrays.			
	type	*integer*		
• **dish**	type	*object*		
	properties			
	• **receptor_ids**	The list of receptors that will be assigned to the Subarray ID. Receptor IDs can be any string, not necessarily numbers. Valid receptor IDs include: SKA dishes: "SKAnnn", where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: "MKTnnn", where nnn is a zero padded integer in the range of 000 to 063.		
		type	*array*	
		items	type	*string*
			pattern	**^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|** (MKT(0[0-5][0-9]|06[0-3]))$
	additionalProperties	False		
additionalProperties	False			

1.10.2 ska-csp-configure

CSP config 2.4

Example (TMC input for science_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
```

(continues on next page)

(continued from previous page)

```
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for science_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
```

(continues on next page)

(continued from previous page)


```
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-00"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-01"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```


Example (CSP configuration for cal_a visibility scan)


```
{
```


(continues on next page)

(continued from previous page)

```
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9744, 1]
```

(continues on next page)

(continued from previous page)

```
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for PSS scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.1",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "PSS-BF",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }],
        "search_window": [{
            "search_window_id": 0,
            "search_window_tuning": 1000,
            "tdc_enable": true
        }]
    },
    "pss": {
        "beam_bandwidth": 300,
        "channels_per_beam": 4096,
        "acceleration_search": false,
        "single_pulse_search": true,
        "integration_time": 600,
        "acc_range": 0,
        "number_of_trials": 0,
        "time_resolution": 4,
        "ps_dm": 1000.0,
        "sps_dm": 1000.0,
        "timesample_per_block": 28125000,
        "sub_bands": 64,
```

(continues on next page)

(continued from previous page)

```
        "buffer_size": 18,
        "hsum_control": 16,
        "cxft_control": {},
        "cand_sift": {},
        "cand_output": {},
        "sp_threshold": 10.0,
        "sp_opt_pars": {},
        "dred_beam_stats": {},
        "cdos_control": {},
        "fldo_control": {
            "phase_split": true,
            "channel_scale": true,
            "max_phases": 16
        },
        "rfim_control": {},
        "beam": [{
            "beam_id": 1,
            "reference_frame": "ICRS",
            "ra": 82.75,
            "dec": 21.0,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.25",
            "dest_port": 9021
        }, {
            "beam_id": 2,
            "reference_frame": "ICRS",
            "ra": 84.25,
            "dec": 21.5,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.26",
            "dest_port": 9021
        }]
    }
}
```

Example (CSP configuration for PST beam configuration)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
```

(continues on next page)

(continued from previous page)

```
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "beam": {}
    }
}
```

Example (CSP configuration for PST pulsar timing scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1,
        "eb_id": "eb-m001-20230712-56789"
    },
    "cbf": {
        "fsp": [{
```

(continues on next page)

(continued from previous page)

```
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "1",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "PULSAR_TIMING",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
```

(continues on next page)

(continued from previous page)

```
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 10000.5,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "pt": {
                "dispersion_measure": 100.0,
                "rotation_measure": 0.0,
                "ephemeris": "",
                "pulsar_phase_predictor": "",
                "output_frequency_channels": 1,
                "output_phase_bins": 64,
                "num_sk_config": 1,
                "sk_config": [{
                    "sk_range": [0.8, 0.9],
                    "sk_integration_limit": 100,
                    "sk_excision_limit": 25.0
                }],
                "target_snr": 0.0
            }
        }
    }
}
```

Example (CSP configuration for PST dynamic spectrum scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
```

(continues on next page)

(continued from previous page)


```
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1,
        "eb_id": "eb-m001-20230712-56789"
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "1",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
```


(continues on next page)

(continued from previous page)


```
            "total_bandwidth": 361689.8148,
            "observation_mode": "DYNAMIC_SPECTRUM",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 13000.2,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "ds": {
                "dispersion_measure": 100.0,
                "output_frequency_channels": 1,
                "stokes_parameters": "Q",
                "num_bits_out": 16,
                "time_decimation_factor": 10,
                "frequency_decimation_factor": 4,
                "requantisation_scale": 1.0,
                "requantisation_length": 1.0
            }
        }
    }
}
```


Example (CSP configuration for PST flow through scan)

```
{
```

(continues on next page)

(continued from previous page)


```
    "interface": "https://schema.skao.int/ska-csp-configure/2.4",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1,
        "eb_id": "eb-m001-20230712-56789"
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "1",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
```


(continues on next page)

(continued from previous page)


```
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "FLOW_THROUGH",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 20000.0,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "ft": {
                "num_bits_out": 32,
                "num_channels": 1,
                "channels": [1],
                "requantisation_scale": 1.0,
                "requantisation_length": 1.0
            }
        }
    }
}
```


Example (CSP configuration for PST voltage recording scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.4",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "low",
        "subarray_id": 1,
        "eb_id": "eb-m001-20230712-56789"
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "1",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
```


(continues on next page)

(continued from previous page)


```
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "VOLTAGE_RECORDER",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "LIN",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 20000.0,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }]
        }
    }
}
```

https://schema.skao.int/ska-csp-configure/2.4			
type	*object*		
properties			
• **interface**	type	*string*	
• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
	type	*object*	
	properties		
	• **subarray_name**	Name and scope of current subarray the sub-array.	
		type	*string*
	additionalProperties	False	
• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
	Common CSP config 2.4		
• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
	CBF config 2.4		
• pss	default	null	
	PSS configuration 2.4		
• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
	default	null	
	PST configuration 2.4		
additionalProperties	False		

Common CSP config 2.4

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• config_id	type	*string*	
	default	null	
• **frequency_band**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array. The value of 'low' is used to only within SKA Low. As this field is a mandatory field but bands 1, 2, 3, 4, 5a and 5b only make sense for SKA Mid.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b)|low)$	
• band_5_tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarray_id**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 2.4

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*	
properties		
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)	
	type	*integer*
	default	null
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*	
	type	*integer*
	default	null
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.	
	type	*string*
	default	null
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.	
	type	*string*
	default	null
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).	
	type	*object*
	default	null
	properties	
	additionalProperties	True
• **fsp**	type	*array*
	items	*FSP config 2.4*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.	
	default	null
	VLBI config 2.4	
• search_window	type	*array*
	default	null
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 2.4*
additionalProperties	False	

FSP config 2.4

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))| (MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 1 – continued from previous page

• chan-nel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• chan-nel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• out-put_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• out-put_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• out-put_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• out-put_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 1 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.4

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.4

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.4

type	*object*	
properties		
• **beam_bandwidth**	Beam bandwidth (MHz)	
	type	*integer*
• **channels_per_beam**	Number of channels per beam	
	type	*integer*
• **acceleration_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **single_pulse_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **integration_time**	Scan duration.	

continues on next page

Table 2 – continued from previous page

	type	*integer*	
• acc_range	Range in source acceleration to be searched.		
	type	*integer*	
	default	null	
• **number_of_trials**	Number of trials to be performed.		
	type	*integer*	
• **time_resolution**	Time resolution of input data.		
	type	*integer*	
• **ps_dm**	Dispersion corretion for acceleration search.		
	type	*number*	
• **sps_dm**	Dispersion corretion for transient search.		
	type	*number*	
• **timesam-ple_per_block**	Number of time samples in each block of data.		
	type	*integer*	
• **sub_bands**	Number of frequency band groups summed up during folding.		
	type	*integer*	
• **buffer_size**	Size of the buffer receiving raw data. (2**buffer_size)		
	type	*integer*	
• **hsum_control**	Number of the "harmonic folds" on the initial Fourier power-spectrum summed up.		
	type	*integer*	
• **cxft_control**	CXFT control parameters.		
	type	*object*	
• **cand_sift**	Constraints on matches between candidates.		
	type	*object*	
• **cand_output**	Define data sinks and subscriber to be notified.		
	type	*object*	
• **sp_threshold**	Threshold for a single pulse trigger. (Tuned to system noise and RFI env.)		
	type	*number*	
• **sp_opt_pars**	Single pulse optimization parameters.		
	type	*object*	
• **dred_beam_stats**	DRED: statistics of spectra to derive the normalization factors.		
	type	*object*	
• **cdos_control**	CDOS: control parameters and related statistical data.		
	type	*object*	
• **rfim_control**	RFIM control parameters.		
	type	*object*	
• **fldo_control**	FLDO control parameters.		
	type	*object*	
	properties		
	• **phase_split**	type	*boolean*
	• **channel_scale**	type	*boolean*
	• **max_phases**	type	*integer*
	additionalProperties	True	
• **beam**	type	*array*	
	items	*PSS beam config 2.4*	

continues on next page

Table 2 – continued from previous page

additionalProperties	False

PSS beam config 2.4

type	*object*		
properties			
• **beam_id**	Search Beam ID.		
	type	*integer*	
• ra	Right Ascension of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• dec	Declination of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• reference_frame	reference frame for pointing coordinates		
	default	null	
	allOf	type	*string*
		enum	ICRS, HORIZON
• **centre_frequency**	Centre frequency of the search beam.		
	type	*number*	
• **beam_delay_centre**	Beam delay center, relative to the array delay center.		
	anyOf	type	*number*
		type	*string*
• dest_host	Per beam destination host address for PSS output.		
	type	*string*	
	default	null	
• dest_port	Per beam destination port for PSS output.		
	type	*integer*	
	default	null	
additionalProperties	False		

PST configuration 2.4

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• scan	Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.	
	default	null
	PST scan configuration 2.4	
• beam	Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement. As of version 2.3 this schema has no elements and is deprecated	
	default	null
	PST beam configuration 2.4	
additionalProperties	False	

PST scan configuration 2.4

Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• **activation_time**	Date and time when to start the PST reconfiguration. **Units:** UTC timestamp **Keyword:** ACTIVATION_TIME	
	type	*string*
• timing_beam_id	Identifier assigned by LMC/TM used to identify the beam configuraiton. PST selects which PST server to use for this scan and timing beam, and provides a mapping from the timing beam identifier by the TM to PST capability id. **Keyword:** BEAM	
	type	*string*
	default	null
• **bits_per_sample**	The number of bits per complex-values time sample in the CBF output data. Valid values are 16, 24, or 32. **Keyword:** NBIT	
	type	*integer*
• **num_of_polarizations**	The number of polarizations in the CBF output data. Valid values are 1 or 2. **Keyword:** NPOL	
	type	*integer*
• **udp_nsamp**	The number of time samples for each single polarization and the a single frequency in each UDP packet sent by CBF. Note: this must be an integer multiple of WT_NSMAP **Range:** 4 (Low), 32 (Mid) **Keyword:** UDP_NSAMP	
	type	*integer*
• **wt_nsamp**	The number of time samples described by as single relative weight. There is a unique relative weight for each frequency channel, and each relative weight describes both polarizations. **Range:** 4 (Low), 32 (Mid) **Keyword:** WT_NSAMP	
	type	*integer*
• **udp_nchan**	The number of contiguous frequency channels in each UDP packet sent by CBF. **Range:** 24 (Low), 185 (Mid) **Keyword:** UDP_NCHAN	

continues on next page

Table 3 – continued from previous page

	type	*integer*	
• **num_frequency_channels**	The total number of frequency channels into which the total critical bandwidth has been divided. This must be an integer multiple of udp_nchan **Range:** 1 to 82944 **Keyword:** OBSNCHAN		
	type	*integer*	
• **centre_frequency**	Centre frequency of to the total (critical) bandwidth spanned by the frequency channels. **Units:** Hz **Range:** 50e6 to 12800e6 **Keyword:** OBSFREQ		
	type	*number*	
• **total_bandwidth**	Total (critical) bandwidth spanned by the channels of the observation. Low: 0.00361 to 300 MHz Mid: 0.053.76 to 2500 MHz **Units:** Hz **Range:** 3610 to 2.5e9 **Keyword:** OBSBW		
	type	*number*	
• **observation_mode**	The observation mode used for the scan. The value VOLTAGE_RECORDER is added for AA0.5, while the other values will be needed for in the future for data processing. **Keyword:** OBSMODE		
	allOf	type	*string*
		enum	PULSAR_TIMING, DYNAMIC_SPECTRUM, FLOW_THROUGH, VOLTAGE_RECORDER
• **observer_id**	The observer in charge of the observations. **Keyword:** OBSERVER		
	type	*string*	
• **project_id**	The project that the observations are for. **Keyword:** PROJID		
	type	*string*	
• **pointing_id**	The ID for the sub-array pointing. **Keyword:** PNT_ID		
	type	*string*	
• **source**	The name of the source. **Keyword:** SRC_NAME		
	type	*string*	
• **itrf**	The International Terrestrial Reference Frame (ITRF) coordinates of the telescope delay centre. **Units:** metres **Keyword:** ITRF		
	type	*array*	
	items	type	*number*
• **receiver_id**	The receiver name or ID (instrument). **Keyword:** FRONTEND		
	type	*string*	
• **feed_polarization**	The native polarization of feed. **Range:** LIN or CIRC **Keyword:** FD_POLN		
	allOf	type	*string*
		enum	LIN, CIRC

continues on next page

Table 3 – continued from previous page

<table>
<tr><td rowspan="3">• feed_handedness</td><td colspan="3">Code for sense of feed.
For value of +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.
Range: -1 or +1 Keyword: FD_HAND</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td>integer</td></tr>
<tr><td>enum</td><td>-1, 1</td></tr>
<tr><td rowspan="2">• feed_angle</td><td colspan="3">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 feed angle or position angle (clockwise when looking down on a prime focuse receiver).
Units: degrees Range: -180 to 180. Keyword: FD_SANG</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
<tr><td rowspan="3">• feed_tracking_mode</td><td colspan="3">The tracking mode for the feed:
• FA - constant feed angle and that the feed stays fixed with respect to the telescope's reference frame.
• CPA - the feed rotates to maintain a constant phase angle (i.e. it tracks the variation of the parallactic angle.). When the cordinate mode is GALATIC, PA is with respect to Galactic north and similarly for coordinate mode ECLIPTIC then PA is with respect to ecliptic north.
• SPA - the feed angle is held fixed at an angle such that the requested PA is obtained at the mid-point of the observation.
• TPA - is only relevant for scan observations - the feed is rotated to maintain a constant angle with respect to the scan direction.
Range: FA, CPA, SPA, or TPA Keyword: FD_MODE</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>FA, CPA, SPA, TPA</td></tr>
<tr><td rowspan="2">• feed_position_angle</td><td colspan="3">The requested angle of feed reference.
For feed_mode = 'FA' this is respect to the telescope's reference frame (feed_angle = 0), and for feed_mode = 'CPA' this is with respect to the celestial north (parallic angle = 0) or with respect to the Galactic north for coordinate_mode = 'GALACTIC'.
Range: -180 to +180.
Keyword: FA_REQ</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
<tr><td rowspan="3">• oversampling_ratio</td><td colspan="3">The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]).
Range: 8/7 or 4/3 Keyword: OVERSAMP</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>items</td><td>type</td><td>integer</td></tr>
<tr><td rowspan="2">• coordinates</td><td colspan="3">The tied-array beam's tracking co-ordinates.
As of version 2.2 of the schema this only handles equitorial tracking which means uses RA/Dec J2000.0 coords but PST may support different tracking modes and coordinates the future.</td></tr>
<tr><td colspan="3">PST RA_Dec coordinates 2.4</td></tr>
<tr><td rowspan="2">• max_scan_length</td><td colspan="3">The maximum length of the observation.
Units: seconds Range: 30 - 43200 Keyword: SCANLEN_MAX</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
<tr><td rowspan="2">• subint_duration</td><td colspan="3">The length of each output sub-integration.
Units: seconds Range: 1 - 60 Keyword: OUTSUBINT</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
</table>

continues on next page

Table 3 – continued from previous page

• **receptors**	An array of receptor IDs for the receptors included in the sub-array. **Keyword:** ANTENNA			
	type	*array*		
	items	type	*string*	
• **receptor_weights**	Weight for each receptor. **Range:** 0 - 1.0 **Keyword:** ANT_WEIGHTS			
	type	*array*		
	items	type	*number*	
• num_rfi_frequency_masks	The number of frequency ranges to be masked. **Range:** 0 - 1024 **Keyword:** NMASK			
	type	*integer*		
	default	0		
• rfi_frequency_masks	A two-dimensional array of length of num_frequency_mask of known RFI frequency ranges to excise from the data. The array contains mask pairs of [f_min, f_max] pairs for known frequency ranges containing RFI not excised by the CBF. The overall dimension of this array is num_frequency_mask x 2. **Units:** Hz **Keyword:** FREQ_MASK			
	type	*array*		
	default	null		
	items	type	*array*	
		items	type	*number*
• destination_address	The destination address for the PST output data. Includes IPv4 Address, port number.			
	type	*array*		
	default	null		
	items	anyOf	type	*string*
			type	*integer*
• test_vector_id	Identifier for a test vectore that will be present in the tied-array beam data stream beam CBF and PST. **Keyword:** TEST_VECTOR			
	type	*string*		
	default	null		
• pt	Pulsar Timing specific parameters for the 'PULSAR_TIMING' mode configuration.			
	default	null		
	PST 'PULSAR_TIMING' mode configuration 2.4			
• ds	Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.			
	default	null		
	PST 'DYNAMIC_SPECTRUM' mode configuration 2.4			
• ft	Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.			
	default	null		
	PST 'FLOW_THROUGH' mode configuration 2.4			
• **num_channelization_stages**	The number of stages used to channelize the data: e.g. * for Low, there are 2 stages: 1 in LFAA and 1 in CBF * for Mid, there are 2 stages: 1 in FSP and 1 in PST BF. **Keyword:** NSTAGE			
	type	*integer*		
• **channelization_stages**	List of configuration for each channelization stage.			
	type	*array*		
	items	Pulsar Timing specific parameters for channelization stage configuration.		
		PST channelization stage configuration 2.4		
additionalProperties	False			

PST RA_Dec coordinates 2.4

Pulsar Timing specific parameters for RA/Dec tracking coordinates.

type	*object*	
properties		
• equinox	The coordinate epoch. This can be in Julian date or Modified Julian Date. **Units:** years **Range:** >= 2000 **Keyword:** EQUINOX	
	type	*number*
	default	2000.0
• **ra**	The Right Accession (RA) of the coordinates used for tracking. Valid formats is 'hh:mm:ss.sss' or 'ddd.ddd' **Keyword:** STT_CTD1	
	type	*string*
• **dec**	The declination (Dec) of the coordinates used for tracking. Valid formats is 'hh:mm:ss.sss' or 'ddd.ddd' **Keyword:** STT_CTD2	
	type	*string*
additionalProperties	False	

PST 'PULSAR_TIMING' mode configuration 2.4

Pulsar Timing specific parameters for the 'PULSAR_TIMING' mode configuration.

type	*object*	
properties		
• **dispersion_measure**	The dispersion measure for coherent/inchoerent de-dispersion. **Units:** pccm^-3 **Range:** 0 - 100000 **Keyword:** DM	
	type	*number*
• rotation_measure	The rotation measure for phase-coherent Faraday rotation correction. **Units:** radians per metre squared **Keyword:** RM	
	type	*number*
	default	null
• **ephemeris**	The ephemeris of the pulsar being observed. **Units:** PSRCAT compatible ASCII string **Keyword:** EPHEMERIS	
	type	*string*
• **pulsar_phase_predictor**	Pulsar phase predictor generated from ephemeris. **Units:** TEMPO2 compatible ASCII string **Keyword:** PREDICTOR	
	type	*string*
• **output_frequency_channels**	The number of output frequency channels. This must be between 1 and the number of observation channels. **Keyword:** OUTNCHAN	
	type	*integer*
• **output_phase_bins**	The number of output phase bins. **Range:** 64 - 2048 **Keyword:** OUTNBIN	
	type	*integer*
• **num_sk_config**	The number of spectral kurtosis (SK) configurations to apply. **Keyword:** N_SK	
	type	*integer*
• **sk_config**	List of spectral kurtosis configurations.	
	type	*array*
	items	Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.
		PST spectral kurtosis configuration 2.4
• **target_snr**	The signal-to-noise ratio (SNR) of the on-pulse flux for the scan. May be used to prematurely end a scan when the integrated SNR reaches the target. A value of 0 indicates there is no limit. **Keyword:** TARGET_SNR	
	type	*number*
additionalProperties	False	

PST spectral kurtosis configuration 2.4

Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.

type	*object*		
properties			
• **sk_range**	Frequency ranges for each spectral kurtosis (SK) configuration. **Units:** Hz **Keyword:** SK_RNG		
	type	*array*	
	items	type	*number*
• **sk_integration_limit**	The number of input time samples integrated into each spectral kurtosis (SK) statistic. **Range:** 64 - 1024 **Keyword:** SK_INTS		
	type	*integer*	
• **sk_excision_limit**	Spectral kurtosis excision limits (RFI threshold) in units of standard deviations. **Range:** 1 - 100 **Keyword:** SK_EXIS		
	type	*number*	
additionalProperties	False		

PST 'DYNAMIC_SPECTRUM' mode configuration 2.4

Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.

type	*object*		
properties			
• **dispersion_measure**	The dispersion measture for coherent/inchoerent de-dispersion. This is only required for pulsar timing and dynamic spectrum modes. **Range:** [0, 100000] **Keyword:** DM		
	type	*number*	
• rotation_measure	The rotation measure for phase-coherent Faraday rotation correction. **Units:** radians per metre squared **Keyword:** RM		
	type	*number*	
	default	null	
• **output_frequency_channels**	The number of output frequency channels. This must be between 1 and the number of observation channels. **Keyword:** OUTNCHAN		
	type	*integer*	
• **stokes_parameters**	The Stokes parameters to output when in Dynamic spectrum mode. **Range:** string with a combination of I, Q, U, and V. **Keyword:** STOKES_FB		
	type	*string*	
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **time_decimation_factor**	The number of input samples per output time sample when in Dynamic Spectrum mode. **Keyword:** TDEC_FB		
	type	*integer*	
• **frequency_decimation_factor**	The number of input frequency channels incoherently added to each output frequency channel in Dynamic Spectrum. This is required in addition to output_frequency_channels because some frequency channels may be merged coherently to increase temporal resolution. **Keyword:** FDEC_FB		
	type	*integer*	
• num_sk_config	The number of spectral kurtosis (SK) configurations to apply. **Keyword:** N_SK		

continues on next page

Table 4 – continued from previous page

	type	*integer*	
	default	null	
• sk_config	List of spectral kurtosis configurations.		
	type	*array*	
	default	null	
	items	Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.	
		PST spectral kurtosis configuration 2.4	
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE		
	type	*number*	
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH		
	type	*number*	
additionalProperties	False		

PST 'FLOW_THROUGH' mode configuration 2.4

Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.

type	*object*		
properties			
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **num_channels**	The number of input channels to be recorded. This value must be less than or equal to the output_frequency_channels. **Keyword:** NCHAN_FT		
	type	*integer*	
• **channels**	An array of indexes of frequency channels to be recorded. Length of this array should be num_channels. **Keyword:** CHAN_FT		
	type	*array*	
	items	type	*integer*
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE		
	type	*number*	
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH		
	type	*number*	
additionalProperties	False		

PST channelization stage configuration 2.4

Pulsar Timing specific parameters for channelization stage configuration.

type	*object*		
properties			
• **num_filter_taps**	Total number of taps in the prototype filter (i.e. over all arms) used in the stage. **Keyword:** NSTAP_k		
	type	*integer*	
• **filter_coefficients**	An array of filter coefficients that define the (time domain) response function of the prototype filter used in the stage. Length of this is num_filter_taps. **Keyword:** COEFF_k		
	type	*array*	
	items	type	*number*
• **num_frequency_channels**	The number of frequency channels output by each polyphase filter bank (PFB) for this stage. **Keyword:** NCHAN_PFB_k		
	type	*integer*	
• **oversampling_ratio**	The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]). **Keyword:** OVERSAMP_k		
	type	*array*	
	items	type	*integer*
additionalProperties	False		

PST beam configuration 2.4

Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

As of version 2.3 this schema has no elements and is deprecated

type	*object*
properties	
additionalProperties	False

CSP config 2.3

Example (TMC input for science_a visibility scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
```


(continues on next page)

(continued from previous page)


```
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```


Example (CSP configuration for science_a visibility scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
```


(continues on next page)

(continued from previous page)

```
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-00"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-01"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
```

(continues on next page)

(continued from previous page)

```
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for cal_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
```

(continues on next page)

(continued from previous page)

```
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9744, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for PSS scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.1",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "PSS-BF",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }],
        "search_window": [{
            "search_window_id": 0,
            "search_window_tuning": 1000,
            "tdc_enable": true
        }]
    },
    "pss": {
        "beam_bandwidth": 300,
        "channels_per_beam": 4096,
```

(continues on next page)

(continued from previous page)

```
        "acceleration_search": false,
        "single_pulse_search": true,
        "integration_time": 600,
        "acc_range": 0,
        "number_of_trials": 0,
        "time_resolution": 4,
        "ps_dm": 1000.0,
        "sps_dm": 1000.0,
        "timesample_per_block": 28125000,
        "sub_bands": 64,
        "buffer_size": 18,
        "hsum_control": 16,
        "cxft_control": {},
        "cand_sift": {},
        "cand_output": {},
        "sp_threshold": 10.0,
        "sp_opt_pars": {},
        "dred_beam_stats": {},
        "cdos_control": {},
        "fldo_control": {
            "phase_split": true,
            "channel_scale": true,
            "max_phases": 16
        },
        "rfim_control": {},
        "beam": [{
            "beam_id": 1,
            "reference_frame": "ICRS",
            "ra": 82.75,
            "dec": 21.0,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.25",
            "dest_port": 9021
        }, {
            "beam_id": 2,
            "reference_frame": "ICRS",
            "ra": 84.25,
            "dec": 21.5,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.26",
            "dest_port": 9021
        }]
    }
}
```

Example (CSP configuration for PST beam configuration)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
```

(continues on next page)

(continued from previous page)


```
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "beam": {}
    }
}
```


Example (CSP configuration for PST pulsar timing scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
```


(continues on next page)

(continued from previous page)


```
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
```


(continues on next page)

(continued from previous page)


```
            "total_bandwidth": 361689.8148,
            "observation_mode": "PULSAR_TIMING",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 10000.5,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "pt": {
                "dispersion_measure": 100.0,
                "rotation_measure": 0.0,
                "ephemeris": "",
                "pulsar_phase_predictor": "",
                "output_frequency_channels": 1,
                "output_phase_bins": 64,
                "num_sk_config": 1,
                "sk_config": [{
                    "sk_range": [0.8, 0.9],
                    "sk_integration_limit": 100,
                    "sk_excision_limit": 25.0
                }],
                "target_snr": 0.0
            }
        }
    }
}
```

Example (CSP configuration for PST dynamic spectrum scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
```


(continues on next page)

(continued from previous page)


```
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "DYNAMIC_SPECTRUM",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 13000.2,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "ds": {
                "dispersion_measure": 100.0,
                "output_frequency_channels": 1,
                "stokes_parameters": "Q",
                "num_bits_out": 16,
                "time_decimation_factor": 10,
                "frequency_decimation_factor": 4,
                "requantisation_scale": 1.0,
                "requantisation_length": 1.0
            }
        }
    }
}
```

Example (CSP configuration for PST flow through scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
```


(continues on next page)

(continued from previous page)


```
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "FLOW_THROUGH",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 20000.0,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }],
            "ft": {
                "num_bits_out": 32,
                "num_channels": 1,
                "channels": [1],
                "requantisation_scale": 1.0,
                "requantisation_length": 1.0
            }
        }
    }
}
```


Example (CSP configuration for PST voltage recording scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.3",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
```


(continues on next page)

(continued from previous page)


```
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "VOLTAGE_RECORDER",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "LIN",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 20000.0,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }]
        }
    }
}
```

https://schema.skao.int/ska-csp-configure/2.3			
type	*object*		
properties			
• **interface**	type	*string*	
• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
	type	*object*	
	properties		
	• **subarray_name**	Name and scope of current subarray the sub-array.	
		type	*string*
	additionalProperties	False	
• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
	Common CSP config 2.3		
• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
	CBF config 2.3		
• pss	default	null	
	PSS configuration 2.3		
• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
	default	null	
	PST configuration 2.3		
additionalProperties	False		

Common CSP config 2.3

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• config_id	type	*string*	
	default	null	
• **frequency_band**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band_5_tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarray_id**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 2.3

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*	
properties		
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)	
	type	*integer*
	default	null
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*	
	type	*integer*
	default	null
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.	
	type	*string*
	default	null
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.	
	type	*string*
	default	null
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).	
	type	*object*
	default	null
	properties	
	additionalProperties	True
• **fsp**	type	*array*
	items	*FSP config 2.3*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.	
	default	null
	VLBI config 2.3	
• search_window	type	*array*
	default	null
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 2.3*
additionalProperties	False	

FSP config 2.3

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))| (MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 5 – continued from previous page

<table>
<tr><td rowspan="5">• channel_averaging_map</td><td colspan="5">Table of up to 20 x 2 integers. Each of entries contains:
• Start channel ID, and
• averaging factor.
Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency.
TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies:
• the channel ID (integer) of the first channel, and
• the averaging factor, as follows:
– 0 means do not send channels to SDP,
– 1 means no averaging,
– 2 means average two adjacent channels,
– 3 means average three adjacent channels,
and so on.
If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td colspan="3">array</td></tr>
<tr><td>items</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• channel_offset</td><td colspan="5">Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743.
Note that this offset does not apply to channel maps in this structure (such as channelAveragingMap or outputHost).</td></tr>
<tr><td>type</td><td colspan="4">integer</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="6">• output_link_map</td><td colspan="5">Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="3">items</td><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="2">items</td><td rowspan="2">anyOf</td><td>type</td><td>integer</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="6">• output_host</td><td colspan="5">Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="3">items</td><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="2">items</td><td rowspan="2">anyOf</td><td>type</td><td>integer</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="5">• output_port</td><td colspan="5">Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td colspan="3">array</td></tr>
<tr><td>items</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• output_mac</td><td colspan="5">Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
</table>

continues on next page

Table 5 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.3

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.3

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.3

type	*object*	
properties		
• **beam_bandwidth**	Beam bandwidth (MHz)	
	type	*integer*
• **channels_per_beam**	Number of channels per beam	
	type	*integer*
• **acceleration_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **single_pulse_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **integration_time**	Scan duration.	

continues on next page

Table 6 – continued from previous page

	type	*integer*
• acc_range	Range in source acceleration to be searched.	
	type	*integer*
	default	null
• **number_of_trials**	Number of trials to be performed.	
	type	*integer*
• **time_resolution**	Time resolution of input data.	
	type	*integer*
• **ps_dm**	Dispersion corretion for acceleration search.	
	type	*number*
• **sps_dm**	Dispersion corretion for transient search.	
	type	*number*
• **timesample_per_block**	Number of time samples in each block of data.	
	type	*integer*
• **sub_bands**	Number of frequency band groups summed up during folding.	
	type	*integer*
• **buffer_size**	Size of the buffer receiving raw data. (2**buffer_size)	
	type	*integer*
• **hsum_control**	Number of the "harmonic folds" on the initial Fourier power-spectrum summed up.	
	type	*integer*
• **cxft_control**	CXFT control parameters.	
	type	*object*
• **cand_sift**	Constraints on matches between candidates.	
	type	*object*
• **cand_output**	Define data sinks and subscriber to be notified.	
	type	*object*
• **sp_threshold**	Threshold for a single pulse trigger. (Tuned to system noise and RFI env.)	
	type	*number*
• **sp_opt_pars**	Single pulse optimization parameters.	
	type	*object*
• **dred_beam_stats**	DRED: statistics of spectra to derive the normalization factors.	
	type	*object*
• **cdos_control**	CDOS: control parameters and related statistical data.	
	type	*object*
• **rfim_control**	RFIM control parameters.	
	type	*object*
• **fldo_control**	FLDO control parameters.	
	type	*object*
	properties	
	• **phase_split**	type: *boolean*
	• **channel_scale**	type: *boolean*
	• **max_phases**	type: *integer*
	additionalProperties	True
• **beam**	type	*array*
	items	*PSS beam config 2.3*

continues on next page

Table 6 – continued from previous page

additionalProperties	False

PSS beam config 2.3

type	*object*		
properties			
• **beam_id**	Search Beam ID.		
	type	*integer*	
• ra	Right Ascension of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• dec	Declination of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• reference_frame	reference frame for pointing coordinates		
	default	null	
	allOf	type	*string*
		enum	ICRS, HORIZON
• **centre_frequency**	Centre frequency of the search beam.		
	type	*number*	
• **beam_delay_centre**	Beam delay center, relative to the array delay center.		
	anyOf	type	*number*
		type	*string*
• dest_host	Per beam destination host address for PSS output.		
	type	*string*	
	default	null	
• dest_port	Per beam destination port for PSS output.		
	type	*integer*	
	default	null	
additionalProperties	False		

PST configuration 2.3

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• scan	Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.	
	default	null
	PST scan configuration 2.3	
• beam	Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement. As of version 2.3 this schema has no elements and is deprecated	
	default	null
	PST beam configuration 2.3	
additionalProperties	False	

PST scan configuration 2.3

Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• **activation_time**	Date and time when to start the PST reconfiguration. **Units:** UTC timestamp **Keyword:** ACTIVATION_TIME	
	type	*string*
• timing_beam_id	Identifier assigned by LMC/TM used to identify the beam configuraiton. PST selects which PST server to use for this scan and timing beam, and provides a mapping from the timing beam identifier by the TM to PST capability id. **Keyword:** BEAM	
	type	*string*
	default	null
• **bits_per_sample**	The number of bits per complex-values time sample in the CBF output data. Valid values are 16, 24, or 32. **Keyword:** NBIT	
	type	*integer*
• **num_of_polarizations**	The number of polarizations in the CBF output data. Valid values are 1 or 2. **Keyword:** NPOL	
	type	*integer*
• **udp_nsamp**	The number of time samples for each single polarization and the a single frequency in each UDP packet sent by CBF. Note: this must be an integer multiple of WT_NSMAP **Range:** 4 (Low), 32 (Mid) **Keyword:** UDP_NSAMP	
	type	*integer*
• **wt_nsamp**	The number of time samples described by as single relative weight. There is a unique relative weight for each frequency channel, and each relative weight describes both polarizations. **Range:** 4 (Low), 32 (Mid) **Keyword:** WT_NSAMP	
	type	*integer*
• **udp_nchan**	The number of contiguous frequency channels in each UDP packet sent by CBF. **Range:** 24 (Low), 185 (Mid) **Keyword:** UDP_NCHAN	

continues on next page

Table 7 – continued from previous page

	type	*integer*	
• **num_frequency_channels**	The total number of frequency channels into which the total critical bandwidth has been divided. This must be an integer multiple of udp_nchan **Range:** 1 to 82944 **Keyword:** OBSNCHAN		
	type	*integer*	
• **centre_frequency**	Centre frequency of to the total (critical) bandwidth spanned by the frequency channels. **Units:** Hz **Range:** 50e6 to 12800e6 **Keyword:** OBSFREQ		
	type	*number*	
• **total_bandwidth**	Total (critical) bandwidth spanned by the channels of the observation. Low: 0.00361 to 300 MHz Mid: 0.053.76 to 2500 MHz **Units:** Hz **Range:** 3610 to 2.5e9 **Keyword:** OBSBW		
	type	*number*	
• **observation_mode**	The observation mode used for the scan. The value VOLTAGE_RECORDER is added for AA0.5, while the other values will be needed for in the future for data processing. **Keyword:** OBSMODE		
	allOf	type	*string*
		enum	PULSAR_TIMING, DYNAMIC_SPECTRUM, FLOW_THROUGH, VOLTAGE_RECORDER
• **observer_id**	The observer in charge of the observations. **Keyword:** OBSERVER		
	type	*string*	
• **project_id**	The project that the observations are for. **Keyword:** PROJID		
	type	*string*	
• **pointing_id**	The ID for the sub-array pointing. **Keyword:** PNT_ID		
	type	*string*	
• **source**	The name of the source. **Keyword:** SRC_NAME		
	type	*string*	
• **itrf**	The International Terrestrial Reference Frame (ITRF) coordinates of the telescope delay centre. **Units:** metres **Keyword:** ITRF		
	type	*array*	
	items	type	*number*
• **receiver_id**	The receiver name or ID (instrument). **Keyword:** FRONTEND		
	type	*string*	
• **feed_polarization**	The native polarization of feed. **Range:** LIN or CIRC **Keyword:** FD_POLN		
	allOf	type	*string*
		enum	LIN, CIRC

continues on next page

Table 7 – continued from previous page

<table>
<tr><td>• feed_handedness</td><td>Code for sense of feed.
For value of +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.
Range: -1 or +1 Keyword: FD_HAND
<table><tr><td rowspan="2">allOf</td><td>type</td><td>integer</td></tr><tr><td>enum</td><td>-1, 1</td></tr></table></td></tr>
<tr><td>• feed_angle</td><td>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 feed angle or position angle (clockwise when looking down on a prime focuse receiver).
Units: degrees Range: -180 to 180. Keyword: FD_SANG
<table><tr><td>type</td><td>number</td></tr></table></td></tr>
<tr><td>• feed_tracking_mode</td><td>The tracking mode for the feed:
• FA - constant feed angle and that the feed stays fixed with respect to the telescope's reference frame.
• CPA - the feed rotates to maintain a constant phase angle (i.e. it tracks the variation of the parallactic angle.). When the cordinate mode is GALATIC, PA is with respect to Galactic north and similarly for coordinate mode ECLIPTIC then PA is with respect to ecliptic north.
• SPA - the feed angle is held fixed at an angle such that the requested PA is obtained at the mid-point of the observation.
• TPA - is only relevant for scan observations - the feed is rotated to maintain a constant angle with respect to the scan direction.
Range: FA, CPA, SPA, or TPA Keyword: FD_MODE
<table><tr><td rowspan="2">allOf</td><td>type</td><td>string</td></tr><tr><td>enum</td><td>FA, CPA, SPA, TPA</td></tr></table></td></tr>
<tr><td>• feed_position_angle</td><td>The requested angle of feed reference.
For feed_mode = 'FA' this is respect to the telescope's reference frame (feed_angle = 0), and for feed_mode = 'CPA' this is with respect to the celestial north (parallic angle = 0) or with respect to the Galactic north for coordinate_mode = 'GALACTIC'.
Range: -180 to +180.
Keyword: FA_REQ
<table><tr><td>type</td><td>number</td></tr></table></td></tr>
<tr><td>• oversampling_ratio</td><td>The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]).
Range: 8/7 or 4/3 Keyword: OVERSAMP
<table><tr><td>type</td><td colspan="2">array</td></tr><tr><td>items</td><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• coordinates</td><td>The tied-array beam's tracking co-ordinates.
As of version 2.2 of the schema this only handles equitorial tracking which means uses RA/Dec J2000.0 coords but PST may support different tracking modes and coordinates the future.
PST RA_Dec coordinates 2.3</td></tr>
<tr><td>• max_scan_length</td><td>The maximum length of the observation.
Units: seconds Range: 30 - 43200 Keyword: SCANLEN_MAX
<table><tr><td>type</td><td>number</td></tr></table></td></tr>
<tr><td>• subint_duration</td><td>The length of each output sub-integration.
Units: seconds Range: 1 - 60 Keyword: OUTSUBINT
<table><tr><td>type</td><td>number</td></tr></table></td></tr>
</table>

continues on next page

Table 7 – continued from previous page

• **receptors**	An array of receptor IDs for the receptors included in the sub-array. **Keyword:** ANTENNA			
	type	*array*		
	items	type	*string*	
• **receptor_weights**	Weight for each receptor. **Range:** 0 - 1.0 **Keyword:** ANT_WEIGHTS			
	type	*array*		
	items	type	*number*	
• num_rfi_frequency_masks	The number of frequency ranges to be masked. **Range:** 0 - 1024 **Keyword:** NMASK			
	type	*integer*		
	default	0		
• rfi_frequency_masks	A two-dimensional array of length of num_frequency_mask of known RFI frequency ranges to excise from the data. The array contains mask pairs of [f_min, f_max] pairs for known frequency ranges containing RFI not excised by the CBF. The overall dimension of this array is num_frequency_mask x 2. **Units:** Hz **Keyword:** FREQ_MASK			
	type	*array*		
	default	null		
	items	type	*array*	
		items	type	*number*
• destination_address	The destination address for the PST output data. Includes IPv4 Address, port number.			
	type	*array*		
	default	null		
	items	anyOf	type	*string*
			type	*integer*
• test_vector_id	Identifier for a test vectore that will be present in the tied-array beam data stream beam CBF and PST. **Keyword:** TEST_VECTOR			
	type	*string*		
	default	null		
• pt	Pulsar Timing specific parameters for the 'PULSAR_TIMING' mode configuration.			
	default	null		
	PST 'PULSAR_TIMING' mode configuration 2.3			
• ds	Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.			
	default	null		
	PST 'DYNAMIC_SPECTRUM' mode configuration 2.3			
• ft	Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.			
	default	null		
	PST 'FLOW_THROUGH' mode configuration 2.3			
• **num_channelization_stages**	The number of stages used to channelize the data: e.g. * for Low, there are 2 stages: 1 in LFAA and 1 in CBF * for Mid, there are 2 stages: 1 in FSP and 1 in PST BF. **Keyword:** NSTAGE			
	type	*integer*		
• **channelization_stages**	List of configuration for each channelization stage.			
	type	*array*		
	items	Pulsar Timing specific parameters for channelization stage configuration.		
		PST channelization stage configuration 2.3		
additionalProperties	False			

PST RA_Dec coordinates 2.3

Pulsar Timing specific parameters for RA/Dec tracking coordinates.

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="3">• equinox</td><td colspan="2">The coordinate epoch.
This can be in Julian date or Modified Julian Date.
Units: years Range: >= 2000 Keyword: EQUINOX</td></tr>
<tr><td>type</td><td>number</td></tr>
<tr><td>default</td><td>2000.0</td></tr>
<tr><td rowspan="2">• ra</td><td colspan="2">The Right Accession (RA) of the coordinates used for tracking.
Valid formats is ‘hh:mm:ss.sss’ or ‘ddd.ddd’
Keyword: STT_CTD1</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• dec</td><td colspan="2">The declination (Dec) of the coordinates used for tracking.
Valid formats is ‘hh:mm:ss.sss’ or ‘ddd.ddd’
Keyword: STT_CTD2</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

PST ‘PULSAR_TIMING’ mode configuration 2.3

Pulsar Timing specific parameters for the ‘PULSAR_TIMING’ mode configuration.

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• dispersion_measure</td><td colspan="2">The dispersion measure for coherent/inchoerent de-dispersion.
Units: pccm^-3 Range: 0 - 100000 Keyword: DM</td></tr>
<tr><td>type</td><td>number</td></tr>
<tr><td rowspan="3">• rotation_measure</td><td colspan="2">The rotation measure for phase-coherent Faraday rotation correction.
Units: radians per metre squared Keyword: RM</td></tr>
<tr><td>type</td><td>number</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• ephemeris</td><td colspan="2">The ephemeris of the pulsar being observed.
Units: PSRCAT compatible ASCII string Keyword: EPHEMERIS</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• pulsar_phase_predictor</td><td colspan="2">Pulsar phase predictor generated from ephemeris.
Units: TEMPO2 compatible ASCII string Keyword: PREDICTOR</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• output_frequency_channels</td><td colspan="2">The number of output frequency channels. This must be between 1 and the number of observation channels.
Keyword: OUTNCHAN</td></tr>
<tr><td>type</td><td>integer</td></tr>
<tr><td rowspan="2">• output_phase_bins</td><td colspan="2">The number of output phase bins.
Range: 64 - 2048 Keyword: OUTNBIN</td></tr>
<tr><td>type</td><td>integer</td></tr>
<tr><td rowspan="2">• num_sk_config</td><td colspan="2">The number of spectral kurtosis (SK) configurations to apply.
Keyword: N_SK</td></tr>
<tr><td>type</td><td>integer</td></tr>
<tr><td rowspan="4">• sk_config</td><td colspan="2">List of spectral kurtosis configurations.</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td rowspan="2">items</td><td>Pulsar Timing specific parameters for the spectral kurtosis (SK) for the ‘PULSAR_TIMING’ mode.</td></tr>
<tr><td>PST spectral kurtosis configuration 2.3</td></tr>
<tr><td rowspan="2">• target_snr</td><td colspan="2">The signal-to-noise ratio (SNR) of the on-pulse flux for the scan. May be used to prematurely end a scan when the integrated SNR reaches the target. A value of 0 indicates there is no limit.
Keyword: TARGET_SNR</td></tr>
<tr><td>type</td><td>number</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

PST spectral kurtosis configuration 2.3

Pulsar Timing specific parameters for the spectral kurtosis (SK) for the ‘PULSAR_TIMING’ mode.

type	*object*		
properties			
• **sk_range**	Frequency ranges for each spectral kurtosis (SK) configuration. **Units:** Hz **Keyword:** SK_RNG		
	type	*array*	
	items	type	*number*
• **sk_integration_limit**	The number of input time samples integrated into each spectral kurtosis (SK) statistic. **Range:** 64 - 1024 **Keyword:** SK_INTS		
	type	*integer*	
• **sk_excision_limit**	Spectral kurtosis excision limits (RFI threshold) in units of standard deviations. **Range:** 1 - 100 **Keyword:** SK_EXIS		
	type	*number*	
additionalProperties	False		

PST 'DYNAMIC_SPECTRUM' mode configuration 2.3

Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.

type	*object*		
properties			
• **dispersion_measure**	The dispersion measture for coherent/inchoerent de-dispersion. This is only required for pulsar timing and dynamic spectrum modes. **Range:** [0, 100000] **Keyword:** DM		
	type	*number*	
• rotation_measure	The rotation measure for phase-coherent Faraday rotation correction. **Units:** radians per metre squared **Keyword:** RM		
	type	*number*	
	default	null	
• **output_frequency_channels**	The number of output frequency channels. This must be between 1 and the number of observation channels. **Keyword:** OUTNCHAN		
	type	*integer*	
• **stokes_parameters**	The Stokes parameters to output when in Dynamic spectrum mode. **Range:** string with a combination of I, Q, U, and V. **Keyword:** STOKES_FB		
	type	*string*	
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **time_decimation_factor**	The number of input samples per output time sample when in Dynamic Spectrum mode. **Keyword:** TDEC_FB		
	type	*integer*	
• **frequency_decimation_factor**	The number of input frequency channels incoherently added to each output frequency channel in Dynamic Spectrum. This is required in addition to output_frequency_channels because some frequency channels may be merged coherently to increase temporal resolution. **Keyword:** FDEC_FB		
	type	*integer*	
• num_sk_config	The number of spectral kurtosis (SK) configurations to apply. **Keyword:** N_SK		

continues on next page

Table 8 – continued from previous page

	type	*integer*	
	default	null	
• sk_config	List of spectral kurtosis configurations.		
	type	*array*	
	default	null	
	items	Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.	
		PST spectral kurtosis configuration 2.3	
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE		
	type	*number*	
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH		
	type	*number*	
additionalProperties	False		

PST 'FLOW_THROUGH' mode configuration 2.3

Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.

type	*object*		
properties			
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **num_channels**	The number of input channels to be recorded. This value must be less than or equal to the output_frequency_channels. **Keyword:** NCHAN_FT		
	type	*integer*	
• **channels**	An array of indexes of frequency channels to be recorded. Length of this array should be num_channels. **Keyword:** CHAN_FT		
	type	*array*	
	items	type	*integer*
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE		
	type	*number*	
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH		
	type	*number*	
additionalProperties	False		

PST channelization stage configuration 2.3

Pulsar Timing specific parameters for channelization stage configuration.

type	*object*		
properties			
• **num_filter_taps**	Total number of taps in the prototype filter (i.e. over all arms) used in the stage. **Keyword:** NSTAP_k		
	type	*integer*	
• **filter_coefficients**	An array of filter coefficients that define the (time domain) response function of the prototype filter used in the stage. Length of this is num_filter_taps. **Keyword:** COEFF_k		
	type	*array*	
	items	type	*number*
• **num_frequency_channels**	The number of frequency channels output by each polyphase filter bank (PFB) for this stage. **Keyword:** NCHAN_PFB_k		
	type	*integer*	
• **oversampling_ratio**	The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]). **Keyword:** OVERSAMP_k		
	type	*array*	
	items	type	*integer*
additionalProperties	False		

PST beam configuration 2.3

Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

As of version 2.3 this schema has no elements and is deprecated

type	*object*
properties	
additionalProperties	False

CSP config 2.2

Example (TMC input for science_a visibility scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
```


(continues on next page)

(continued from previous page)

```
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for science_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
```

(continues on next page)

(continued from previous page)

```
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-00"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-01"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
```

(continues on next page)

(continued from previous page)


```
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```


Example (CSP configuration for cal_a visibility scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
```


(continues on next page)

(continued from previous page)

```
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9744, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for PSS scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.1",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "PSS-BF",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }],
        "search_window": [{
            "search_window_id": 0,
            "search_window_tuning": 1000,
            "tdc_enable": true
        }]
    },
    "pss": {
        "beam_bandwidth": 300,
        "channels_per_beam": 4096,
```

(continues on next page)

(continued from previous page)

```
        "acceleration_search": false,
        "single_pulse_search": true,
        "integration_time": 600,
        "acc_range": 0,
        "number_of_trials": 0,
        "time_resolution": 4,
        "ps_dm": 1000.0,
        "sps_dm": 1000.0,
        "timesample_per_block": 28125000,
        "sub_bands": 64,
        "buffer_size": 18,
        "hsum_control": 16,
        "cxft_control": {},
        "cand_sift": {},
        "cand_output": {},
        "sp_threshold": 10.0,
        "sp_opt_pars": {},
        "dred_beam_stats": {},
        "cdos_control": {},
        "fldo_control": {
            "phase_split": true,
            "channel_scale": true,
            "max_phases": 16
        },
        "rfim_control": {},
        "beam": [{
            "beam_id": 1,
            "reference_frame": "ICRS",
            "ra": 82.75,
            "dec": 21.0,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.25",
            "dest_port": 9021
        }, {
            "beam_id": 2,
            "reference_frame": "ICRS",
            "ra": 84.25,
            "dec": 21.5,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.26",
            "dest_port": 9021
        }]
    }
}
```

Example (CSP configuration for PST beam configuration)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.2",
    "subarray": {
```

(continues on next page)

(continued from previous page)

```
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
        "beam": {
            "activation_time": "2022-01-19T23:07:45Z",
            "num_channelization_stages": 1,
            "channelization_stages": [{
                "num_filter_taps": 1,
                "filter_coefficients": [1.0],
                "num_frequency_channels": 10,
                "oversampling_ratio": [8, 7]
            }]
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

Example (CSP configuration for PST pulsar timing scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.2",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {
```

(continues on next page)

(continued from previous page)


```
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "beam1",
            "capability": "capability1",
            "scan_id": 1,
            "bits_per_sample": 24,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "PULSAR_TIMING",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "subarray_id": "subarray42",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 10000.5,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "pt": {
                "dispersion_measure": 100.0,
                "rotation_measure": 0.0,
                "ephemeris": "",
                "pulsar_phase_predictor": "",
                "output_frequency_channels": 1,
                "output_phase_bins": 64,
                "num_sk_config": 1,
                "sk_config": [{
                    "sk_range": [0.8, 0.9],
                    "sk_integration_limit": 100,
                    "sk_excision_limit": 25.0
```


(continues on next page)

(continued from previous page)

```
                }],
                "target_snr": 0.0
            }
        }
    }
}
```

Example (CSP configuration for PST dynamic spectrum scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.2",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
```

(continues on next page)

(continued from previous page)


```
        "vlbi": {}
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "beam1",
            "capability": "capability1",
            "scan_id": 1,
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "DYNAMIC_SPECTRUM",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "subarray_id": "subarray42",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 13000.2,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "ds": {
                "dispersion_measure": 100.0,
                "output_frequency_channels": 1,
                "stokes_parameters": "Q",
                "num_bits_out": 16,
                "time_decimation_factor": 10,
                "frequency_decimation_factor": 4,
                "requantisation_scale": 1.0,
```


(continues on next page)

(continued from previous page)

```
                "requantisation_length": 1.0
            }
        }
    }
}
```

Example (CSP configuration for PST flow through scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.2",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
```

(continues on next page)

(continued from previous page)

```
    },
    "pst": {
        "scan": {
            "activation_time": "2022-01-19T23:07:45Z",
            "timing_beam_id": "beam1",
            "capability": "capability1",
            "scan_id": 1,
            "bits_per_sample": 32,
            "num_of_polarizations": 2,
            "udp_nsamp": 32,
            "wt_nsamp": 32,
            "udp_nchan": 24,
            "num_frequency_channels": 432,
            "centre_frequency": 100000000.0,
            "total_bandwidth": 361689.8148,
            "observation_mode": "FLOW_THROUGH",
            "observer_id": "jdoe",
            "project_id": "project1",
            "pointing_id": "pointing1",
            "subarray_id": "subarray42",
            "source": "J1921+2153",
            "itrf": [5109360.133, 2006852.586, -3238948.127],
            "receiver_id": "receiver3",
            "feed_polarization": "CIRC",
            "feed_handedness": 1,
            "feed_angle": 1.234,
            "feed_tracking_mode": "FA",
            "feed_position_angle": 10.0,
            "oversampling_ratio": [8, 7],
            "coordinates": {
                "equinox": 2000.0,
                "ra": "19:21:44.815",
                "dec": "21.884"
            },
            "max_scan_length": 20000.0,
            "subint_duration": 30.0,
            "receptors": ["SKA001", "SKA036"],
            "receptor_weights": [0.4, 0.6],
            "num_rfi_frequency_masks": 1,
            "rfi_frequency_masks": [
                [1.0, 1.1]
            ],
            "destination_address": ["192.168.178.26", 9021],
            "ft": {
                "num_bits_out": 32,
                "num_channels": 1,
                "channels": [1],
                "requantisation_scale": 1.0,
                "requantisation_length": 1.0
            }
        }
    }
```

(continues on next page)

(continued from previous page)

```
}
```

https://schema.skao.int/ska-csp-configure/2.2			
type	*object*		
properties			
• **interface**	type	*string*	
• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
	type	*object*	
	properties		
	• **subarray_name**	Name and scope of current subarray the sub-array.	
		type	*string*
	additionalProperties	False	
• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
	Common CSP config 2.2		
• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
	CBF config 2.2		
• pss	default	null	
	PSS configuration 2.2		
• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
	default	null	
	PST configuration 2.2		
additionalProperties	False		

Common CSP config 2.2

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• config_id	type	*string*	
	default	null	
• **frequency_band**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band_5_tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarray_id**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 2.2

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*	
properties		
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)	
	type	*integer*
	default	null
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*	
	type	*integer*
	default	null
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.	
	type	*string*
	default	null
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.	
	type	*string*
	default	null
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).	
	type	*object*
	default	null
	properties	
	additionalProperties	True
• **fsp**	type	*array*
	items	*FSP config 2.2*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.	
	default	null
	VLBI config 2.2	
• search_window	type	*array*
	default	null
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 2.2*
additionalProperties	False	

FSP config 2.2

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))|
(MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 9 – continued from previous page

• channel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• channel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• output_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• output_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 9 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.2

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.2

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.2

type	*object*	
properties		
• **beam_bandwidth**	Beam bandwidth (MHz)	
	type	*integer*
• **channels_per_beam**	Number of channels per beam	
	type	*integer*
• **acceleration_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **single_pulse_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **integration_time**	Scan duration.	

continues on next page

Table 10 – continued from previous page

	type	*integer*	
• acc_range	Range in source acceleration to be searched.		
	type	*integer*	
	default	null	
• **number_of_trials**	Number of trials to be performed.		
	type	*integer*	
• **time_resolution**	Time resolution of input data.		
	type	*integer*	
• **ps_dm**	Dispersion corretion for acceleration search.		
	type	*number*	
• **sps_dm**	Dispersion corretion for transient search.		
	type	*number*	
• **timesample_per_block**	Number of time samples in each block of data.		
	type	*integer*	
• **sub_bands**	Number of frequency band groups summed up during folding.		
	type	*integer*	
• **buffer_size**	Size of the buffer receiving raw data. (2**buffer_size)		
	type	*integer*	
• **hsum_control**	Number of the "harmonic folds" on the initial Fourier power-spectrum summed up.		
	type	*integer*	
• **cxft_control**	CXFT control parameters.		
	type	*object*	
• **cand_sift**	Constraints on matches between candidates.		
	type	*object*	
• **cand_output**	Define data sinks and subscriber to be notified.		
	type	*object*	
• **sp_threshold**	Threshold for a single pulse trigger. (Tuned to system noise and RFI env.)		
	type	*number*	
• **sp_opt_pars**	Single pulse optimization parameters.		
	type	*object*	
• **dred_beam_stats**	DRED: statistics of spectra to derive the normalization factors.		
	type	*object*	
• **cdos_control**	CDOS: control parameters and related statistical data.		
	type	*object*	
• **rfim_control**	RFIM control parameters.		
	type	*object*	
• **fldo_control**	FLDO control parameters.		
	type	*object*	
	properties		
	• **phase_split**	type	*boolean*
	• **channel_scale**	type	*boolean*
	• **max_phases**	type	*integer*
	additionalProperties	True	
• **beam**	type	*array*	
	items	*PSS beam config 2.2*	

continues on next page

Table 10 – continued from previous page

additionalProperties	False

PSS beam config 2.2

type	*object*		
properties			
• **beam_id**	Search Beam ID.		
	type	*integer*	
• ra	Right Ascension of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• dec	Declination of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• reference_frame	reference frame for pointing coordinates		
	default	null	
	allOf	type	*string*
		enum	ICRS, HORIZON
• **centre_frequency**	Centre frequency of the search beam.		
	type	*number*	
• **beam_delay_centre**	Beam delay center, relative to the array delay center.		
	anyOf	type	*number*
		type	*string*
• dest_host	Per beam destination host address for PSS output.		
	type	*string*	
	default	null	
• dest_port	Per beam destination port for PSS output.		
	type	*integer*	
	default	null	
additionalProperties	False		

PST configuration 2.2

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• scan	Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.	
	default	null
	PST scan configuration 2.2	
• beam	Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement. As of version 2.3 this schema has no elements and is deprecated	
	default	null
	PST beam configuration 2.2	
additionalProperties	False	

PST scan configuration 2.2

Pulsar Timing specific scan configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• **activation_time**	Date and time when to start the PST reconfiguration. **Units:** UTC timestamp **Keyword:** ACTIVATION_TIME	
	type	*string*
• **timing_beam_id**	Identifier assigned by LMC/TM used to identify the beam configuraiton. PST selects which PST server to use for this scan and timing beam, and provides a mapping from the timing beam identifier by the TM to PST capability id. **Keyword:** BEAM	
	type	*string*
	default	null
• **capability**	Identifier of the capability PST Beam to be used for this configuration. **Keyword:** CAPABILITY	
	type	*string*
• **scan_id**	The identifier for the scan to be configured. This is a 64bits long. **Keyword:** SCAN_ID	
	type	*integer*
• **subarray_id**	The ID for the sub-array. **Keyword:** SUBARRAY_ID	
	type	*string*
• **bits_per_sample**	The number of bits per complex-values time sample in the CBF output data. Valid values are 16, 24, or 32. **Keyword:** NBIT	
	type	*integer*
• **num_of_polarizations**	The number of polarizations in the CBF output data. Valid values are 1 or 2. **Keyword:** NPOL	
	type	*integer*

continues on next page

Table 11 – continued from previous page

• **udp_nsamp**	The number of time samples for each single polarization and the a single frequency in each UDP packet sent by CBF. Note: this must be an integer multiple of WT_NSMAP **Range:** 4 (Low), 32 (Mid) **Keyword:** UDP_NSAMP		
	type	*integer*	
• **wt_nsamp**	The number of time samples described by as single relative weight. There is a unique relative weight for each frequency channel, and each relative weight describes both polarizations. **Range:** 4 (Low), 32 (Mid) **Keyword:** WT_NSAMP		
	type	*integer*	
• **udp_nchan**	The number of contiguous frequency channels in each UDP packet sent by CBF. **Range:** 24 (Low), 185 (Mid) **Keyword:** UDP_NCHAN		
	type	*integer*	
• **num_frequency_channels**	The total number of frequency channels into which the total critical bandwidth has been divided. This must be an integer multiple of udp_nchan **Range:** 1 to 82944 **Keyword:** OBSNCHAN		
	type	*integer*	
• **centre_frequency**	Centre frequency of to the total (critical) bandwidth spanned by the frequency channels. **Units:** Hz **Range:** 50e6 to 12800e6 **Keyword:** OBSFREQ		
	type	*number*	
• **total_bandwidth**	Total (critical) bandwidth spanned by the channels of the observation. Low: 0.00361 to 300 MHz Mid: 0.053.76 to 2500 MHz **Units:** Hz **Range:** 3610 to 2.5e9 **Keyword:** OBSBW		
	type	*number*	
• **observation_mode**	The observation mode used for the scan. **Range:** PULSAR_TIMING, DYNAMIC_SPECTRUM, or FLOW_THROUGH **Keyword:** OBSMODE		
	allOf	type	*string*
		enum	PULSAR_TIMING, DYNAMIC_SPECTRUM, FLOW_THROUGH
• **observer_id**	The observer in charge of the observations. **Keyword:** OBSERVER		
	type	*string*	
• **project_id**	The project that the observations are for. **Keyword:** PROJID		
	type	*string*	
• **pointing_id**	The ID for the sub-array pointing. **Keyword:** PNT_ID		
	type	*string*	
• **source**	The name of the source. **Keyword:** SRC_NAME		
	type	*string*	
• **itrf**	The International Terrestrial Reference Frame (ITRF) coordinates of the telescope delay centre. **Units:** metres **Keyword:** ITRF		
	type	*array*	
	items	type	*number*
• **receiver_id**	The receiver name or ID (instrument). **Keyword:** FRONTEND		
	type	*string*	

continues on next page

Table 11 – continued from previous page

<table>
<tr><td rowspan="3">• feed_polarization</td><td colspan="3">The native polarization of feed.
Range: LIN or CIRC Keyword: FD_POLN</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>LIN, CIRC</td></tr>
<tr><td rowspan="3">• feed_handedness</td><td colspan="3">Code for sense of feed.
For value of +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.
Range: -1 or +1 Keyword: FD_HAND</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td>integer</td></tr>
<tr><td>enum</td><td>-1, 1</td></tr>
<tr><td rowspan="2">• feed_angle</td><td colspan="3">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 feed angle or position angle (clockwise when looking down on a prime focuse receiver).
Units: degrees Range: -180 to 180. Keyword: FD_SANG</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
<tr><td rowspan="3">• feed_tracking_mode</td><td colspan="3">The tracking mode for the feed:
• FA - constant feed angle and that the feed stays fixed with respect to the telescope's reference frame.
• CPA - the feed rotates to maintain a constant phase angle (i.e. it tracks the variation of the parallactic angle.). When the cordinate mode is GALATIC, PA is with respect to Galactic north and similarly for coordinate mode ECLIPTIC then PA is with respect to ecliptic north.
• SPA - the feed angle is held fixed at an angle such that the requested PA is obtained at the mid-point of the observation.
• TPA - is only relevant for scan observations - the feed is rotated to maintain a constant angle with respect to the scan direction.
Range: FA, CPA, SPA, or TPA Keyword: FD_MODE</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>FA, CPA, SPA, TPA</td></tr>
<tr><td rowspan="2">• feed_position_angle</td><td colspan="3">The requested angle of feed reference.
For feed_mode = 'FA' this is respect to the telescope's reference frame (feed_angle = 0), and for feed_mode = 'CPA' this is with respect to the celestial north (parallic angle = 0) or with respect to the Galactic north for coordinate_mode = 'GALACTIC'.
Range: -180 to +180.
Keyword: FA_REQ</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
<tr><td rowspan="3">• oversampling_ratio</td><td colspan="3">The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]).
Range: 8/7 or 4/3 Keyword: OVERSAMP</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>items</td><td>type</td><td>integer</td></tr>
<tr><td rowspan="2">• coordinates</td><td colspan="3">The tied-array beam's tracking co-ordinates.
As of version 2.2 of the schema this only handles equitorial tracking which means uses RA/Dec J2000.0 coords but PST may support different tracking modes and coordinates the future.</td></tr>
<tr><td colspan="3">PST RA_Dec coordinates 2.2</td></tr>
<tr><td rowspan="2">• max_scan_length</td><td colspan="3">The maximum length of the observation.
Units: seconds Range: 30 - 43200 Keyword: SCANLEN_MAX</td></tr>
<tr><td>type</td><td colspan="2">number</td></tr>
</table>

continues on next page

Table 11 – continued from previous page

• **subint_duration**	The length of each output sub-integration. **Units:** seconds **Range:** 1 - 60 **Keyword:** OUTSUBINT			
	type	*number*		
• **receptors**	An array of receptor IDs for the receptors included in the sub-array. **Keyword:** ANTENNA			
	type	*array*		
	items	type	*string*	
• **receptor_weights**	Weight for each receptor. **Range:** 0 - 1.0 **Keyword:** ANT_WEIGHTS			
	type	*array*		
	items	type	*number*	
• num_rfi_frequency_masks	The number of frequency ranges to be masked. **Range:** 0 - 1024 **Keyword:** NMASK			
	type	*integer*		
	default	0		
• rfi_frequency_masks	A two-dimensional array of length of num_frequency_mask of known RFI frequency ranges to excise from the data. The array contains mask pairs of [f_min, f_max] pairs for known frequency ranges containing RFI not excised by the CBF. The overall dimension of this array is num_frequency_mask x 2. **Units:** Hz **Keyword:** FREQ_MASK			
	type	*array*		
	default	null		
	items	type	*array*	
		items	type	*number*
• cal_mode	Operation mode for the injected calibration: • OFF: there is no injected calibration. • SYNC: the calibration is pulsed synchronously with the folding frequency. • EXT1/EXT2: the calibration is driven by one of two possible user defined external signals. **Range:** [OFF, SYNC, EXT1, EXT2] **Keyword:** CAL_MODE			
	default	null		
	allOf	type	*string*	
		enum	OFF, SYNC, EXT1, EXT2	
• calibration_modulation_frequency	The modulation frequency for the injected calibration signal. **Range:** 0.001 - 1000 **Units:** Hertz **Keyword:** CAL_FREQ			
	type	*number*		
	default	null		
• calibration_duty_cycle	Duty cycle for the injected calibration signal. **Range:** 0.0 - 1.0 **Keyword:** CAL_DCYC			
	type	*number*		
	default	null		
• calibration_phase	The calibration phase with respect to time. Phase of the leasing edge of the injected calibration signal in calibration SYNC mode. **Range:** 0.0 - 1.0 **Keyword:** CAL_PHS			
	type	*number*		
	default	null		
• calibration_num_phase_states	The number of pulses in one period of the calibration phase. **Keyword:** CAL_NPHS			
	type	*number*		
	default	null		

continues on next page

Table 11 – continued from previous page

• destination_address	The destination address for the PST output data. Includes IPv4 Address, port number.			
	type	*array*		
	default	null		
	items	anyOf	type	*string*
			type	*integer*
• test_vector_id	Identifier for a test vectore that will be present in the tied-array beam data stream beam CBF and PST. **Keyword:** TEST_VECTOR			
	type	*string*		
	default	null		
• pt	Pulsar Timing specific parameters for the 'PULSAR_TIMING' mode configuration.			
	default	null		
	PST 'PULSAR_TIMING' mode configuration 2.2			
• ds	Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.			
	default	null		
	PST 'DYNAMIC_SPECTRUM' mode configuration 2.2			
• ft	Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.			
	default	null		
	PST 'FLOW_THROUGH' mode configuration 2.2			
additionalProperties	False			

PST RA_Dec coordinates 2.2

Pulsar Timing specific parameters for RA/Dec tracking coordinates.

type	*object*	
properties		
• equinox	The coordinate epoch. This can be in Julian date or Modified Julian Date. **Units:** years **Range:** >= 2000 **Keyword:** EQUINOX	
	type	*number*
	default	2000.0
• **ra**	The Right Accession (RA) of the coordinates used for tracking. Valid formats is 'hh:mm:ss.sss' or 'ddd.ddd' **Keyword:** STT_CTD1	
	type	*string*
• **dec**	The declination (Dec) of the coordinates used for tracking. Valid formats is 'hh:mm:ss.sss' or 'ddd.ddd' **Keyword:** STT_CTD2	
	type	*string*
additionalProperties	False	

PST ‘PULSAR_TIMING’ mode configuration 2.2

Pulsar Timing specific parameters for the ‘PULSAR_TIMING’ mode configuration.

type	*object*	
properties		
• **dispersion_measure**	The dispersion measure for coherent/inchoerent de-dispersion. **Units:** pccm^-3 **Range:** 0 - 100000 **Keyword:** DM	
	type	*number*
• rotation_measure	The rotation measure for phase-coherent Faraday rotation correction. **Units:** radians per metre squared **Keyword:** RM	
	type	*number*
	default	null
• **ephemeris**	The ephemeris of the pulsar being observed. **Units:** PSRCAT compatible ASCII string **Keyword:** EPHEMERIS	
	type	*string*
• **pulsar_phase_predictor**	Pulsar phase predictor generated from ephemeris. **Units:** TEMPO2 compatible ASCII string **Keyword:** PREDICTOR	
	type	*string*
• **output_frequency_channels**	The number of output frequency channels. This must be between 1 and the number of observation channels. **Keyword:** OUTNCHAN	
	type	*integer*
• **output_phase_bins**	The number of output phase bins. **Range:** 64 - 2048 **Keyword:** OUTNBIN	
	type	*integer*
• **num_sk_config**	The number of spectral kurtosis (SK) configurations to apply. **Keyword:** N_SK	
	type	*integer*
• **sk_config**	List of spectral kurtosis configurations.	
	type	*array*
	items	Pulsar Timing specific parameters for the spectral kurtosis (SK) for the ‘PULSAR_TIMING’ mode. *PST spectral kurtosis configuration 2.2*
• **target_snr**	The signal-to-noise ratio (SNR) of the on-pulse flux for the scan. May be used to prematurely end a scan when the integrated SNR reaches the target. A value of 0 indicates there is no limit. **Keyword:** TARGET_SNR	
	type	*number*
additionalProperties	False	

PST spectral kurtosis configuration 2.2

Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.

type	*object*		
properties			
• **sk_range**	Frequency ranges for each spectral kurtosis (SK) configuration. **Units:** Hz **Keyword:** SK_RNG		
	type	*array*	
	items	type	*number*
• **sk_integration_limit**	The number of input time samples integrated into each spectral kurtosis (SK) statistic. **Range:** 64 - 1024 **Keyword:** SK_INTS		
	type	*integer*	
• **sk_excision_limit**	Spectral kurtosis excision limits (RFI threshold) in units of standard deviations. **Range:** 1 - 100 **Keyword:** SK_EXIS		
	type	*number*	
additionalProperties	False		

PST 'DYNAMIC_SPECTRUM' mode configuration 2.2

Pulsar Timing specific parameters for the 'DYNAMIC_SPECTRUM' mode configuration.

type	*object*		
properties			
• **dispersion_measure**	The dispersion measure for coherent/inchoerent de-dispersion. This is only required for pulsar timing and dynamic spectrum modes. **Range:** [0, 100000] **Keyword:** DM		
	type	*number*	
• rotation_measure	The rotation measure for phase-coherent Faraday rotation correction. **Units:** radians per metre squared **Keyword:** RM		
	type	*number*	
	default	null	
• **output_frequency_channels**	The number of output frequency channels. This must be between 1 and the number of observation channels. **Keyword:** OUTNCHAN		
	type	*integer*	
• **stokes_parameters**	The Stokes parameters to output when in Dynamic spectrum mode. **Range:** string with a combination of I, Q, U, and V. **Keyword:** STOKES_FB		
	type	*string*	
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **time_decimation_factor**	The number of input samples per output time sample when in Dynamic Spectrum mode. **Keyword:** TDEC_FB		
	type	*integer*	

continues on next page

Table 12 – continued from previous page

• **frequency_decimation_factor**	The number of input frequency channels incoherently added to each output frequency channel in Dynamic Spectrum. This is required in addition to output_frequency_channels because some frequency channels may be merged coherently to increase temporal resolution. **Keyword:** FDEC_FB	
	type	*integer*
• num_sk_config	The number of spectral kurtosis (SK) configurations to apply. **Keyword:** N_SK	
	type	*integer*
	default	null
• sk_config	List of spectral kurtosis configurations.	
	type	*array*
	default	null
	items	Pulsar Timing specific parameters for the spectral kurtosis (SK) for the 'PULSAR_TIMING' mode.
		PST spectral kurtosis configuration 2.2
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE	
	type	*number*
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH	
	type	*number*
additionalProperties	False	

PST 'FLOW_THROUGH' mode configuration 2.2

Pulsar Timing specific parameters for the 'FLOW_THROUGH' mode configuration.

type	*object*		
properties			
• **num_bits_out**	The number of bits per output sample. **Range:** 1, 2, 4, 8, 16 or 32 **Keyword:** NBIT_OUT		
	allOf	type	*integer*
		enum	1, 2, 4, 8, 16, 32
• **num_channels**	The number of input channels to be recorded. This value must be less than or equal to the output_frequency_channels. **Keyword:** NCHAN_FT		
	type	*integer*	
• **channels**	An array of indexes of frequency channels to be recorded. Length of this array should be num_channels. **Keyword:** CHAN_FT		
	type	*array*	
	items	type	*integer*
• **requantisation_scale**	Scale factor to govern the dynamic range for fixed precision output to be applied during re-quantisation. **Keyword:** DIGITIZER_SCALE		
	type	*number*	
• **requantisation_length**	Length of data to be used when determining the scaling factors used for fixed precision output during re-quantisation. **Units:** seconds **Keyword:** DIGITIZER_LENGTH		
	type	*number*	
additionalProperties	False		

PST beam configuration 2.2

Pulsar Timing specific beam configuration parameters. This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

As of version 2.3 this schema has no elements and is deprecated

type	*object*	
properties		
• **activation_time**	Date and time when to start the PST reconfiguration in UTC. **Keyword:** ACTIVATION_TIME	
	type	*string*
• **num_channelization_stages**	The number of stages used to channelize the data: e.g. * for Low, there are 2 stages: 1 in LFAA and 1 in CBF * for Mid, there are 2 stages: 1 in FSP and 1 in PST BF. **Keyword:** NSTAGE	
	type	*integer*
• **channelization_stages**	List of configuration for each channelization stage.	
	type	*array*
	items	Pulsar Timing specific parameters for channelization stage configuration. *PST channelization stage configuration 2.2*
additionalProperties	False	

PST channelization stage configuration 2.2

Pulsar Timing specific parameters for channelization stage configuration.

type	*object*		
properties			
• **num_filter_taps**	Total number of taps in the prototype filter (i.e. over all arms) used in the stage. **Keyword:** NSTAP_k		
	type	*integer*	
• **filter_coefficients**	An array of filter coefficients that define the (time domain) response function of the prototype filter used in the stage. Length of this is num_filter_taps. **Keyword:** COEFF_k		
	type	*array*	
	items	type	*number*
• **num_frequency_channels**	The number of frequency channels output by each polyphase filter bank (PFB) for this stage. **Keyword:** NCHAN_PFB_k		
	type	*integer*	
• **oversampling_ratio**	The oversampling ratio expressed as a fraction as an array of int, with the first value the numerator and the second is the denominator. (e.g. 8/7 is assigned as [8,7]). **Keyword:** OVERSAMP_k		
	type	*array*	
	items	type	*integer*
additionalProperties	False		

CSP config 2.1

Example (TMC input for science_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
```

(continues on next page)

(continued from previous page)

```
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for science_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
```

(continues on next page)

(continued from previous page)

```
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-00"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-01"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

Example (CSP configuration for cal_a visibility scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
```

(continues on next page)

(continued from previous page)

```
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9744, 1]
            ]
        }],
        "vlbi": {}
```

(continues on next page)

(continued from previous page)

```
    },
    "pst": {}
}
```

Example (CSP configuration for PSS scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.1",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "PSS-BF",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0
        }],
        "search_window": [{
            "search_window_id": 0,
            "search_window_tuning": 1000,
            "tdc_enable": true
        }]
    },
    "pss": {
        "beam_bandwidth": 300,
        "channels_per_beam": 4096,
        "acceleration_search": false,
        "single_pulse_search": true,
        "integration_time": 600,
        "acc_range": 0,
        "number_of_trials": 0,
        "time_resolution": 4,
        "ps_dm": 1000.0,
        "sps_dm": 1000.0,
        "timesample_per_block": 28125000,
        "sub_bands": 64,
        "buffer_size": 18,
        "hsum_control": 16,
        "cxft_control": {},
```

(continues on next page)

(continued from previous page)


```
        "cand_sift": {},
        "cand_output": {},
        "sp_threshold": 10.0,
        "sp_opt_pars": {},
        "dred_beam_stats": {},
        "cdos_control": {},
        "fldo_control": {
            "phase_split": true,
            "channel_scale": true,
            "max_phases": 16
        },
        "rfim_control": {},
        "beam": [{
            "beam_id": 1,
            "reference_frame": "ICRS",
            "ra": 82.75,
            "dec": 21.0,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.25",
            "dest_port": 9021
        }, {
            "beam_id": 2,
            "reference_frame": "ICRS",
            "ra": 84.25,
            "dec": 21.5,
            "centre_frequency": 1400.0,
            "beam_delay_centre": 0.0,
            "dest_host": "192.168.178.26",
            "dest_port": 9021
        }]
    }
}
```

https://schema.skao.int/ska-csp-configure/2.1			
type	*object*		
properties			
• **interface**	type	*string*	
• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
	type	*object*	
	properties		
	• **subarray_name**	Name and scope of current subarray the sub-array.	
		type	*string*
	additionalProperties	False	
• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
	Common CSP config 2.1		
• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
	CBF config 2.1		
• pss	default	null	
	PSS configuration 2.1		
• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
	default	null	
	PST configuration 2.1		
additionalProperties	False		

Common CSP config 2.1

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• config_id</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="3">• frequency_band</td><td colspan="3">Frequency band applies for all the receptors (VCCs) that belong to the sub-array.</td></tr>
<tr><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^(1|2|3|4|5(a|b))$</td></tr>
<tr><td rowspan="4">• band_5_tuning</td><td colspan="3">Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4).
Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream.
The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td>items</td><td>type</td><td>number</td></tr>
<tr><td rowspan="4">• eb_id</td><td colspan="3">Execution block ID to associate scan configs to an observation.
This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation.
This ID does not have to be unique for a scan configuration but should be unique for different observations.
For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.</td></tr>
<tr><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• subarray_id</td><td colspan="3">Subarray number</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

CBF config 2.1

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*	
properties		
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)	
	type	*integer*
	default	null
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*	
	type	*integer*
	default	null
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.	
	type	*string*
	default	null
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.	
	type	*string*
	default	null
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).	
	type	*object*
	default	null
	properties	
	additionalProperties	True
• **fsp**	type	*array*
	items	*FSP config 2.1*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.	
	default	null
	VLBI config 2.1	
• search_window	type	*array*
	default	null
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 2.1*
additionalProperties	False	

FSP config 2.1

type	*object*		
properties			
• **fsp_id**	type	*integer*	
• **function_mode**	allOf	type	*string*
		enum	CORR, PSS-BF, PST-BF, VLBI
• receptors	Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP). Valid receptor IDs include: SKA dishes: "SKAnnn", where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: "MKTnnn", where nnn is a zero padded integer in the range of 000 to 063.		
	type	*array*	
	default	null	
	items	type	*string*
		pattern	**^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))|** (MKT(0[0-5][0-9]|06[0-3]))$
• **frequency_slice_id**	Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).		
	type	*integer*	
• **zoom_factor**	Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6]. When n=0 the full Frequency Slice bandwidth is correlated. BW > 0 implies 'Zoom Window' configuration; the spectral Zoom Window tuning must be specified.		
	type	*integer*	
• zoom_window_tuning	The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated. Step size <= 0.01MHz. The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.		
	type	*integer*	
	default	null	
• **integration_factor**	Integration time for the correlation products, defines multiple of 140 milliseconds.		
	type	*integer*	

continues on next page

Table 13 – continued from previous page

• chan-nel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• chan-nel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• out-put_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• out-put_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• out-put_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• out-put_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 13 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.1

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.1

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.1

type	*object*	
properties		
• **beam_bandwidth**	Beam bandwidth (MHz)	
	type	*integer*
• **channels_per_beam**	Number of channels per beam	
	type	*integer*
• **acceleration_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **single_pulse_search**	Processing Mode: Acceleration Search (a.k.a. Pulsar Search) and Single Pulse Search (a.k.a. Transient Search) can be performed concurrently.	
	type	*boolean*
• **integration_time**	Scan duration.	

continues on next page

Table 14 – continued from previous page

	type	*integer*	
• acc_range	Range in source acceleration to be searched.		
	type	*integer*	
	default	null	
• **number_of_trials**	Number of trials to be performed.		
	type	*integer*	
• **time_resolution**	Time resolution of input data.		
	type	*integer*	
• **ps_dm**	Dispersion corretion for acceleration search.		
	type	*number*	
• **sps_dm**	Dispersion corretion for transient search.		
	type	*number*	
• **timesam-ple_per_block**	Number of time samples in each block of data.		
	type	*integer*	
• **sub_bands**	Number of frequency band groups summed up during folding.		
	type	*integer*	
• **buffer_size**	Size of the buffer receiving raw data. (2**buffer_size)		
	type	*integer*	
• **hsum_control**	Number of the "harmonic folds" on the initial Fourier power-spectrum summed up.		
	type	*integer*	
• **cxft_control**	CXFT control parameters.		
	type	*object*	
• **cand_sift**	Constraints on matches between candidates.		
	type	*object*	
• **cand_output**	Define data sinks and subscriber to be notified.		
	type	*object*	
• **sp_threshold**	Threshold for a single pulse trigger. (Tuned to system noise and RFI env.)		
	type	*number*	
• **sp_opt_pars**	Single pulse optimization parameters.		
	type	*object*	
• **dred_beam_stats**	DRED: statistics of spectra to derive the normalization factors.		
	type	*object*	
• **cdos_control**	CDOS: control parameters and related statistical data.		
	type	*object*	
• **rfim_control**	RFIM control parameters.		
	type	*object*	
• **fldo_control**	FLDO control parameters.		
	type	*object*	
	properties		
	• **phase_split**	type	*boolean*
	• **channel_scale**	type	*boolean*
	• **max_phases**	type	*integer*
	additionalProperties	True	
• **beam**	type	*array*	
	items	*PSS beam config 2.1*	

continues on next page

Table 14 – continued from previous page

additionalProperties	False

PSS beam config 2.1

type	*object*		
properties			
• **beam_id**	Search Beam ID.		
	type	*integer*	
• ra	Right Ascension of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• dec	Declination of sub-array beam target, in degrees.		
	type	*number*	
	default	null	
• reference_frame	reference frame for pointing coordinates		
	default	null	
	allOf	type	*string*
		enum	ICRS, HORIZON
• **centre_frequency**	Centre frequency of the search beam.		
	type	*number*	
• **beam_delay_centre**	Beam delay center, relative to the array delay center.		
	anyOf	type	*number*
		type	*string*
• dest_host	Per beam destination host address for PSS output.		
	type	*string*	
	default	null	
• dest_port	Per beam destination port for PSS output.		
	type	*integer*	
	default	null	
additionalProperties	False		

PST configuration 2.1

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

CSP config 2.0

Example (TMC input)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```


Example (CSP configuration for science_a scan)

```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-00"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
```


(continues on next page)

(continued from previous page)


```
            "output_host": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "output_mac": [
                [0, "06-00-00-00-00-01"]
            ],
            "output_port": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```


Example (CSP configuration for cal_a scan)


```
{
    "interface": "https://schema.skao.int/ska-csp-configure/2.0",
    "subarray": {
        "subarray_name": "science period 23"
    },
    "common": {
        "config_id": "sbi-mvp01-20200325-00001-science_A",
        "frequency_band": "1",
        "subarray_id": 1
    },
    "cbf": {
        "fsp": [{
            "fsp_id": 1,
            "function_mode": "CORR",
            "frequency_slice_id": 1,
            "integration_factor": 1,
            "zoom_factor": 0,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 0,
            "output_link_map": [
                [0, 0],
                [200, 1]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9000, 1]
            ]
        }, {
```


(continues on next page)

(continued from previous page)


```
            "fsp_id": 2,
            "function_mode": "CORR",
            "frequency_slice_id": 2,
            "integration_factor": 1,
            "zoom_factor": 1,
            "zoom_window_tuning": 650000,
            "channel_averaging_map": [
                [0, 2],
                [744, 0]
            ],
            "channel_offset": 744,
            "output_link_map": [
                [0, 4],
                [200, 5]
            ],
            "output_host": [
                [0, "192.168.1.1"]
            ],
            "output_port": [
                [0, 9744, 1]
            ]
        }],
        "vlbi": {}
    },
    "pst": {}
}
```

<table>
<tr><td colspan="4">https://schema.skao.int/ska-csp-configure/2.0</td></tr>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• interface</td><td>type</td><td colspan="2">string</td></tr>
<tr><td rowspan="6">• subarray</td><td colspan="3">subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.</td></tr>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• subarray_name</td><td colspan="2">Name and scope of current subarray the sub-array.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
<tr><td rowspan="2">• common</td><td colspan="3">Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.</td></tr>
<tr><td colspan="3">Common CSP config 2.0</td></tr>
<tr><td rowspan="2">• cbf</td><td colspan="3">Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD</td></tr>
<tr><td colspan="3">CBF config 2.0</td></tr>
<tr><td rowspan="2">• pss</td><td>default</td><td colspan="2">null</td></tr>
<tr><td colspan="3">PSS configuration 2.0</td></tr>
<tr><td rowspan="3">• pst</td><td colspan="3">Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td colspan="3">PST configuration 2.0</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

Common CSP config 2.0

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• config_id	type	*string*	
	default	null	
• **frequency_band**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band_5_tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarray_id**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 2.0

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="3">• frequency_band_offset_stream1</td><td colspan="2">Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a ‘natural’ Frequency Slice boundary). If specified, applies for all the receptors in the sub-array.
Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only.
Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH).
Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)</td></tr>
<tr><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="3">• frequency_band_offset_stream2</td><td colspan="2">See frequencyBandOffsetStream1</td></tr>
<tr><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="3">• delay_model_subscription_point</td><td colspan="2">FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="3">• doppler_phase_corr_subscription_point</td><td colspan="2">The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="5">• rfi_flagging_mask</td><td colspan="2">Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>additionalProperties</td><td>True</td></tr>
<tr><td rowspan="2">• fsp</td><td>type</td><td>array</td></tr>
<tr><td>items</td><td>FSP config 2.0</td></tr>
<tr><td rowspan="3">• vlbi</td><td colspan="2">Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td colspan="2">VLBI config 2.0</td></tr>
<tr><td rowspan="4">• search_window</td><td>type</td><td>array</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">items</td><td>Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming.</td></tr>
<tr><td>Search window config 2.0</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

FSP config 2.0

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))| (MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 15 – continued from previous page

• channel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• channel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• output_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• output_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 15 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.0

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.0

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.0

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

PST configuration 2.0

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

CSP config 1.0

Example (TMC input)


```
{
    "interface": "https://schema.skatelescope.org/ska-csp-configure/1.0",
    "subarray": {
        "subarrayName": "science period 23"
    },
    "common": {
        "id": "sbi-mvp01-20200325-00001-science_A",
        "frequencyBand": "1",
        "subarrayID": 1
    },
    "cbf": {
        "fsp": [{
            "fspID": 1,
            "functionMode": "CORR",
            "frequencySliceID": 1,
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 0,
            "outputLinkMap": [
                [0, 0],
                [200, 1]
            ]
        }, {
            "fspID": 2,
            "functionMode": "CORR",
            "frequencySliceID": 2,
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 744,
```


(continues on next page)

(continued from previous page)

```
            "outputLinkMap": [
                [0, 4],
                [200, 5]
            ]
        }],
        "vlbi": {}
    }
}
```

Example (CSP configuration for science_a scan)

```
{
    "interface": "https://schema.skatelescope.org/ska-csp-configure/1.0",
    "subarray": {
        "subarrayName": "science period 23"
    },
    "common": {
        "id": "sbi-mvp01-20200325-00001-science_A",
        "frequencyBand": "1",
        "subarrayID": 1
    },
    "cbf": {
        "fsp": [{
            "fspID": 1,
            "functionMode": "CORR",
            "frequencySliceID": 1,
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 0,
            "outputLinkMap": [
                [0, 0],
                [200, 1]
            ],
            "outputHost": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"]
            ],
            "outputMac": [
                [0, "06-00-00-00-00-00"]
            ],
            "outputPort": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }, {
            "fspID": 2,
            "functionMode": "CORR",
            "frequencySliceID": 2,
```

(continues on next page)

(continued from previous page)

```
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 744,
            "outputLinkMap": [
                [0, 4],
                [200, 5]
            ],
            "outputHost": [
                [0, "192.168.0.3"],
                [400, "192.168.0.4"]
            ],
            "outputMac": [
                [0, "06-00-00-00-00-01"]
            ],
            "outputPort": [
                [0, 9000, 1],
                [400, 9000, 1]
            ]
        }],
        "vlbi": {}
    }
}
```

Example (CSP configuration for cal_a scan)

```
{
    "interface": "https://schema.skatelescope.org/ska-csp-configure/1.0",
    "subarray": {
        "subarrayName": "science period 23"
    },
    "common": {
        "id": "sbi-mvp01-20200325-00001-science_A",
        "frequencyBand": "1",
        "subarrayID": 1
    },
    "cbf": {
        "fsp": [{
            "fspID": 1,
            "functionMode": "CORR",
            "frequencySliceID": 1,
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 0,
            "outputLinkMap": [
```

(continues on next page)

(continued from previous page)


```
                [0, 0],
                [200, 1]
            ],
            "outputHost": [
                [0, "192.168.1.1"]
            ],
            "outputPort": [
                [0, 9000, 1]
            ]
        }, {
            "fspID": 2,
            "functionMode": "CORR",
            "frequencySliceID": 2,
            "integrationTime": 1400,
            "corrBandwidth": 0,
            "channelAveragingMap": [
                [0, 2],
                [744, 0]
            ],
            "fspChannelOffset": 744,
            "outputLinkMap": [
                [0, 4],
                [200, 5]
            ],
            "outputHost": [
                [0, "192.168.1.1"]
            ],
            "outputPort": [
                [0, 9744, 1]
            ]
        }],
        "vlbi": {}
    }
}
```

<table>
<tr><td colspan="4">https://schema.skatelescope.org/ska-csp-configure/1.0</td></tr>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• interface</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="6">• subarray</td><td colspan="3">subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.</td></tr>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• subarrayName</td><td colspan="2">Name and scope of current subarray the sub-array.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
<tr><td rowspan="2">• common</td><td colspan="3">Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.</td></tr>
<tr><td colspan="3">Common CSP config 1.0</td></tr>
<tr><td rowspan="2">• cbf</td><td colspan="3">Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD</td></tr>
<tr><td colspan="3">CBF config 1.0</td></tr>
<tr><td rowspan="2">• pss</td><td>default</td><td colspan="2">null</td></tr>
<tr><td colspan="3">PSS configuration 1.0</td></tr>
<tr><td rowspan="3">• pst</td><td colspan="3">Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td colspan="3">PST configuration 1.0</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

Common CSP config 1.0

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• id	type	*string*	
	default	null	
• **frequencyBand**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band5Tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarrayID**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 1.0

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*	
properties		
• frequencyBandOffsetStream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)	
	type	*integer*
	default	null
• frequencyBandOffsetStream2	See *frequencyBandOffsetStream1*	
	type	*integer*
	default	null
• delayModelSubscriptionPoint	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.	
	type	*string*
	default	null
• dopplerPhaseCorrSubscriptionPoint	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.	
	type	*string*
	default	null
• rfiFlaggingMask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).	
	type	*object*
	default	null
	properties	
	additionalProperties	True
• **fsp**	type	*array*
	items	*FSP config 1.0*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.	
	default	null
	VLBI config 1.0	
• search_window	type	*array*
	default	null
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 1.0*
additionalProperties	False	

FSP config 1.0

type	*object*		
properties			
• **fspID**	type	*integer*	
• **functionMode**	allOf	type	*string*
		enum	CORR, PSS-BF, PST-BF, VLBI
• receptors	Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP). Valid receptor IDs include: SKA dishes: "SKAnnn", where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: "MKTnnn", where nnn is a zero padded integer in the range of 000 to 063.		
	type	*array*	
	default	null	
	items	type	*string*
		pattern	**^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))|**(MKT(0[0-5][0-9]|06[0-3]))$
• **frequencySliceID**	Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).		
	type	*integer*	
• **corrBandwidth**	Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6]. When n=0 the full Frequency Slice bandwidth is correlated. BW > 0 implies 'Zoom Window' configuration; the spectral Zoom Window tuning must be specified.		
	type	*integer*	
• zoomWindowTuning	The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated. Step size <= 0.01MHz. The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.		
	type	*integer*	
	default	null	
• **integrationTime**	Integration time for the correlation products, defines multiple of 140 milliseconds.		
	const	1400	

continues on next page

Table 16 – continued from previous page

• channelAveragingMap	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• fspChannelOffset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• outputLinkMap	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• outputHost	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• outputPort	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• outputMac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 16 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 1.0

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 1.0

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

<table>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td rowspan="2">• searchWindowID</td><td colspan="4">Identifier of the 300MHz Search Window. Unique within a sub-array.</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td rowspan="2">• searchWindowTuning</td><td colspan="4">The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td rowspan="2">• tdcEnable</td><td colspan="4">Enable / disable Transient Data Capturefor the Search Window.</td></tr>
<tr><td>type</td><td colspan="3">boolean</td></tr>
<tr><td rowspan="3">• tdcNumBits</td><td colspan="4">Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="3">• tdcPeriodBeforeEpoch</td><td colspan="4">Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction.
The epoch is specified in the command that triggers TDC off-loading (transmission of data).</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="3">• tdcPeriodAfterEpoch</td><td colspan="4">see tdcPeriodBeforeEpoch</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="5">• tdcDestinationAddress</td><td colspan="4">Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses.
Required if TDC is enabled, otherwise not specified.</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="2">items</td><td rowspan="2">anyOf</td><td>type</td><td>integer</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>additionalProperties</td><td colspan="4">False</td></tr>
</table>

PSS configuration 1.0

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• dummy_param</td><td>type</td><td>string</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

PST configuration 1.0

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

CSP config 0.1

Example (TMC input)

```
{
    "id": "sbi-mvp01-20200325-00001-science_A",
    "frequencyBand": "1",
    "fsp": [{
        "fspID": 1,
        "functionMode": "CORR",
        "frequencySliceID": 1,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 0,
        "outputLinkMap": [
            [0, 0],
            [200, 1]
        ]
    }, {
        "fspID": 2,
        "functionMode": "CORR",
        "frequencySliceID": 2,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 744,
        "outputLinkMap": [
            [0, 4],
            [200, 5]
        ]
    }]
}
```

Example (CSP configuration for science_a scan)

```
{
    "id": "sbi-mvp01-20200325-00001-science_A",
    "frequencyBand": "1",
    "fsp": [{
        "fspID": 1,
        "functionMode": "CORR",
        "frequencySliceID": 1,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 0,
        "outputLinkMap": [
            [0, 0],
            [200, 1]
        ],
        "outputHost": [
            [0, "192.168.0.1"],
            [400, "192.168.0.2"]
        ],
        "outputMac": [
            [0, "06-00-00-00-00-00"]
        ],
        "outputPort": [
            [0, 9000, 1],
            [400, 9000, 1]
        ]
    }, {
        "fspID": 2,
        "functionMode": "CORR",
        "frequencySliceID": 2,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 744,
        "outputLinkMap": [
            [0, 4],
            [200, 5]
        ],
        "outputHost": [
            [0, "192.168.0.3"],
            [400, "192.168.0.4"]
        ],
        "outputMac": [
            [0, "06-00-00-00-00-01"]
        ],
        "outputPort": [
            [0, 9000, 1],
```


(continues on next page)

(continued from previous page)

```
            [400, 9000, 1]
        ]
    }]
}
```

Example (CSP configuration for cal_a scan)

```
{
    "id": "sbi-mvp01-20200325-00001-science_A",
    "frequencyBand": "1",
    "fsp": [{
        "fspID": 1,
        "functionMode": "CORR",
        "frequencySliceID": 1,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 0,
        "outputLinkMap": [
            [0, 0],
            [200, 1]
        ],
        "outputHost": [
            [0, "192.168.1.1"]
        ],
        "outputPort": [
            [0, 9000, 1]
        ]
    }, {
        "fspID": 2,
        "functionMode": "CORR",
        "frequencySliceID": 2,
        "integrationTime": 1400,
        "corrBandwidth": 0,
        "channelAveragingMap": [
            [0, 2],
            [744, 0]
        ],
        "fspChannelOffset": 744,
        "outputLinkMap": [
            [0, 4],
            [200, 5]
        ],
        "outputHost": [
            [0, "192.168.1.1"]
        ],
        "outputPort": [
            [0, 9744, 1]
        ]
```

(continues on next page)

(continued from previous page)

```
    }]
}
```

https://schema.skatelescope.org/ska-csp-configure/0.1			
type	*object*		
properties			
• id	type	*string*	
	default	null	
• **frequencyBand**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band5Tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **fsp**	type	*array*	
	items	*FSP config 0.1*	
additionalProperties	False		

FSP config 0.1

type	*object*		
properties			
• **fspID**	type	*integer*	
• **function-Mode**	allOf	type	*string*
		enum	CORR, PSS-BF, PST-BF, VLBI

continues on next page

Table 17 – continued from previous page

<table>
<tr><td rowspan="6">• receptors</td><td colspan="4">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))|
(MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr></tr>
<tr><td rowspan="2">• frequencySliceID</td><td colspan="4">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td rowspan="2">• corrBandwidth</td><td colspan="4">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td rowspan="3">• zoomWindowTuning</td><td colspan="4">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="3">integer</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="2">• integrationTime</td><td colspan="4">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>const</td><td colspan="3">1400</td></tr>
<tr><td rowspan="5">• channelAveragingMap</td><td colspan="4">Table of up to 20 x 2 integers. Each of entries contains:
• Start channel ID, and
• averaging factor.
Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency.
TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies:
• the channel ID (integer) of the first channel, and
• the averaging factor, as follows:
– 0 means do not send channels to SDP,
– 1 means no averaging,
– 2 means average two adjacent channels,
– 3 means average three adjacent channels,
and so on.
If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td colspan="2">array</td></tr>
<tr><td>items</td><td>type</td><td>integer</td></tr>
</table>

continues on next page

Table 17 – continued from previous page

• fspChannelOffset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• outputLinkMap	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• outputHost	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• outputPort	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• outputMac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

1.10.3 ska-csp-scan

CSP scan 2.2

Example JSON

```
{
    "interface": "https://schema.skao.int/ska-csp-scan/2.2",
    "scan_id": 7
}
```

https://schema.skao.int/ska-csp-scan/2.2		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **scan_id**	Scan ID to associate with the data.	
	type	*integer*
additionalProperties	False	

1.10.4 ska-csp-endscan

CSP endscan 2.2

Example JSON


```
{
    "interface": "https://schema.skao.int/ska-csp-endscan/2.2",
    "scan_id": 15
}
```


https://schema.skao.int/ska-csp-endscan/2.2		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **scan_id**	Scan ID to end.	
	type	*integer*
additionalProperties	False	

1.10.5 ska-csp-releaseresources

CSP releaseresources 2.2

Example JSON


```
{
    "interface": "https://schema.skao.int/ska-csp-releaseresources/2.2",
    "subarray_id": 1,
    "release_all": true,
    "receptor_ids": ["SKA001", "SKA036"]
}
```

<table>
<tr><td colspan="4">https://schema.skao.int/ska-csp-releaseresources/2.2</td></tr>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• interface</td><td colspan="3">URI of JSON schema applicable to this JSON payload.</td></tr>
<tr><td>type</td><td colspan="2">string</td></tr>
<tr><td rowspan="2">• subarray_id</td><td colspan="3">Subarray ID which will have its resource(s) released.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• release_all</td><td colspan="3">Set to true if you wish to release all resources assigned to the Subarray.</td></tr>
<tr><td>type</td><td colspan="2">boolean</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="5">• receptor_ids</td><td colspan="3">The list of receptors that will be released from the Subarray ID. Receptor IDs can be any string, not necessarily numbers.
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3
(MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

1.10.6 ska-csp-delaymodel

CSP delaymodel 2.2

Example JSON


```json
{
    "interface": "https://schema.skao.int/ska-csp-delaymodel/2.2",
    "epoch": 12345678,
    "validity_period": 10.0,
    "delay_details": [{
        "receptor": "SKA001",
        "poly_info": [{
            "polarization": "X",
            "coeffs": [1.01, 1.02, 1.03, 1.04, 1.05, 1.06]
        }, {
            "polarization": "Y",
            "coeffs": [1.1, 1.2, 1.3, 1.4, 1.5, 1.6]
        }]
    }, {
        "receptor": "SKA100",
        "poly_info": [{
            "polarization": "X",
            "coeffs": [1.101, 1.102, 1.103, 1.104, 1.105, 1.106]
        }, {
            "polarization": "Y",
```


(continues on next page)

(continued from previous page)

```
            "coeffs": [1.11, 1.12, 1.13, 1.14, 1.15, 1.16]
        }]
    }]
}
```

https://schema.skao.int/ska-csp-delaymodel/2.2		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **epoch**	Time when delay model becomes valid (when Mid.CBF shall apply the new model) specified as 32bit UTC time code containing a count of the number of integer seconds since the January 1, 1970, 00h:00m:00s epoch. Range: 32-bit number	
	type	*integer*
• **validity_period**	validity period of the delay model (starting at epoch) [s] Range: positive number	
	type	*number*
• **delay_details**	type	*array*
	items	*delay details 2.2*
additionalProperties	False	

delay details 2.2

type	*object*	
properties		
• **receptor**	ICD DISH to CSP defines DISH ID as 16 bit field. The receptorID specified in the delay model should be the same as the one inserted in the data stream received from the receptor. Valid receptor IDs include: SKA dishes: "SKAnnn", where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: "MKTnnn", where nnn is a zero padded integer in the range of 000 to 063. Range: any string	
	type	*string*
	pattern	**^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))|** (MKT(0[0-5][0-9]|06[0-3]))$
• **poly_info**	type	*array*
	items	*poly info 2.2*
additionalProperties	False	

poly info 2.2

type	*object*		
properties			
• **polarization**	Polarization of the delay model entry Range: X or Y		
	type	*string*	
• **coeffs**	Delay Model is specified as coefficients for a 5th order polynomial. Coefficients of the polynomial are specified as an array. The delay at time t, where t is measured with respect the beginning of the validity interval is calculated as: d(t) = c0 + c1*t + c2*t^2 + c3*t^3 + c4*t^4 + c5*t^5 Units for coefficients c0,c1,..c5: ns/s^k where k=0,1,..5 Range for coefficients: 64 bit number		
	type	*array*	
	items	type	*number*
additionalProperties	False		

1.11 Low CBF schemas

Schemas used for commands to Low.CBF subarrays

1.11.1 ska-low-cbf-assignresources

LOWCBF assign resources 0.1

Example JSON


```json
{
    "interface": "https://schema.skao.int/ska-low-cbf-assignresources/0.0",
    "lowcbf": {
        "resources": [{
            "device": "fsp_01",
            "shared": true,
            "fw_image": "pst",
            "fw_mode": "unused"
        }, {
            "device": "p4_01",
            "shared": true,
            "fw_image": "p4.bin",
            "fw_mode": "p4"
        }]
    }
}
```

https://schema.skao.int/ska-low-cbf-assignresources/0.1					
type	*object*				
properties					
• **interface**	URI of JSON schema for this command's JSON payload.				
	type	*string*			
• **lowcbf**	LOWCBF resources				
	type	*object*			
	properties				
	• **resources**	array of LOWCBF resources			
		type	*array*		
		items	type	*object*	
			properties		
			• **device**	Name of FSP or P4 device	
				type	*string*
			• **shared**	Whether device is shared with other subarrays	
				type	*boolean*
			• fw_image	Name of firmware image to load on device	
				type	*string*
				default	null
			• fw_mode	Mode in which firmware runs	
				type	*string*
				default	null
			additionalProperties	False	
	additionalProperties	False			
additionalProperties	False				

1.11.2 ska-low-cbf-configurescan

LOWCBF configurescan 0.1

Example JSON

```
{
    "interface": "https://schema.skao.int/ska-low-cbf-configurescan/0.0",
    "lowcbf": {
        "stations": {
            "stns": [
                [1, 0],
                [2, 0],
                [3, 0],
                [4, 0]
            ],
            "stn_beams": [{
                "beam_id": 1,
                "freq_ids": [64, 65, 66, 67, 68, 69, 70, 71],
                "boresight_dly_poly": "tango://delays.skao.int/low/stn-beam/1"
```

(continues on next page)

(continued from previous page)

```
            }]
        },
        "timing_beams": {
            "beams": [{
                "pst_beam_id": 13,
                "stn_beam_id": 1,
                "offset_dly_poly": "url",
                "stn_weights": [0.9, 1.0, 1.0, 0.9],
                "jones": "url",
                "dest_ip": ["10.22.0.1:2345", "10.22.0.3:3456"],
                "dest_chans": [128, 256],
                "rfi_enable": [true, true, true],
                "rfi_static_chans": [1, 206, 997],
                "rfi_dynamic_chans": [242, 1342],
                "rfi_weighted": 0.87
            }]
        },
        "search_beams": "tbd",
        "zooms": "tbd"
    }
}
```

https://schema.skao.int/ska-low-cbf-configurescan/0.1						
type	*object*					
properties						
• **interface**	URI of JSON schema for this command's JSON payload..					
	type	*string*				
• **lowcbf**	LOWCBF configuration for scan					
	type	*object*				
	properties					
	• **stations**	Subarray Stations and station beam input descriptions				
		type	*object*			
		properties				
		• **stns**	type	*array*		
			items	type	*array*	
				items	type	*integer*
		• **stn_beams**	type	*array*		
			items	type	*object*	
				properties		
				• **beam_id**	station beam id	
					type	*integer*
				• **freq_ids**	list of station beam frequency ids	
					type	*array*
					items	type *integer*
				• **boresight_dly_poly**	URL	
					type	*string*
				additionalProperties	False	

continues on next page

Table 18 – continued from previous page

		additional-Properties	False				
	• timing_beams	PST beam outputs descriptions					
		type	*object*				
		default	null				
		properties					
		• **beams**	inner				
			type	*array*			
			items	type	*object*		
				properties			
				• **stn_beam_id**	Station beam ID for pst beamforming		
					type	*integer*	
				• **pst_beam_id**	PST beam ID		
					type	*integer*	
				• firmware	Firmware name		
					type	*string*	
					default	null	
				• **offset_dly_poly**	Delay polynomial source URI		
					type	*string*	
				• **dest_ip**	Beam destination [ip_addr:port]		
					type	*array*	
					items	type	*string*
				• **dest_chans**	Number of fine chans to a destination		
					type	*array*	
					items	type	*integer*
				• **jones**	Jones matrix source URI		
					type	*string*	
				• **stn_weights**	weights for each station		
					type	*array*	
					items	type	*number*
				• rfi_enable	Master enable for RFI flagging		
					type	*array*	
					default	null	
					items	type	*boolean*
				• rfi_static_chans	Freqency IDs to be always flagged		
					type	*array*	
					default	null	
					items	type	*integer*
				• rfi_dynamic_chans	Frequency IDs to be dynamically flagged		
					type	*array*	
					default	null	
					items	type	*integer*
				• rfi_weighted	Parameter for dynamic flagging		
					type	*number*	
					default	null	
				additional-Properties	False		
		additional-Properties	False				
	• search_beams	PSS beam outputs descriptions					
		type	*string*				
		default	null				

continues on next page

Table 18 – continued from previous page

	• visibilities	Visibility output descriptions	
		type	*string*
		default	null
	• zooms	Zoom visibility output descriptions	
		type	*string*
		default	null
	additionalProperties	False	
additionalProperties	False		

1.11.3 ska-low-cbf-scan

LOWCBF scan description 0.1

Example JSON

```
{
    "interface": "https://schema.skao.int/ska-low-cbf-scan/0.0",
    "lowcbf": {
        "scan_id": 1357924680
    }
}
```

https://schema.skao.int/ska-low-cbf-scan/0.1			
type	*object*		
properties			
• **interface**	URI of JSON schema for this command's JSON payload..		
	type	*string*	
• **lowcbf**	LOWCBF scan arguments		
	type	*object*	
	properties		
	• **scan_id**	Scan ID	
		type	*integer*
	additionalProperties	False	
additionalProperties	False		

1.11.4 ska-low-cbf-releaseresources

LOWCBF release resources 0.1

Example JSON

```
{
    "interface": "https://schema.skao.int/ska-low-cbf-releaseresources/0.0",
    "lowcbf": {
        "resources": [{
            "device": "fsp_01"
        }]
```

(continues on next page)

(continued from previous page)

```
    }
}
```

<table>
<tr><td colspan="2">https://schema.skao.int/ska-low-cbf-releaseresources/0.1</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• interface</td><td><table><tr><td colspan="2">URI of JSON schema for this command’s JSON payload..</td></tr><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>• lowcbf</td><td><table>
<tr><td colspan="2">LOWCBF Release resources schema</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• resources</td><td><table>
<tr><td colspan="2">array of LOWCBF resources</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td><table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• device</td><td><table><tr><td colspan="2">Name of FSP or P4 device</td></tr><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table>

1.12 Low MCCS schemas

1.12.1 ska-low-mccs-assignedresources

Low MCCS assigned resources 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-mccs-assignedresources/1.0",
    "subarray_beam_ids": [1],
    "station_ids": [
        [1, 2]
    ],
    "channel_blocks": [3]
}
```

https://schema.skatelescope.org/ska-low-mccs-assignedresources/1.0				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• **subar-ray_beam_ids**	IDs of the MCCS sub-array beams allocated to this MCCS subarray. Each ID must be between 1 and 48, the maximum number of MCCS sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.			
	type	*array*		
	items	type	*integer*	
• **station_ids**	IDs of MCCS stations allocated to each sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.			
	type	*array*		
	items	type	*array*	
		items	type	*integer*
• **chan-nel_blocks**	Number of channel blocks allocated to each sub-array beam. Maximum number of channel blocks = 48.			
	type	*array*		
	items	type	*integer*	
additionalProperties	False			

1.12.2 ska-low-mccs-assignresources

Low MCCS assign resources 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-mccs-assignresources/1.0",
    "subarray_id": 1,
    "subarray_beam_ids": [1],
    "station_ids": [
        [1, 2]
    ],
    "channel_blocks": [3]
}
```

https://schema.skatelescope.org/ska-low-mccs-assignresources/1.0				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• **subarray_id**	ID of sub-array targeted by this resource allocation request			
	type	*integer*		
• **subarray_beam_ids**	IDs of the MCCS sub-array beams to allocate to this MCCS subarray. Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.			
	type	*array*		
	items	type	*integer*	
• **station_ids**	IDs of MCCS stations to allocate to this sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.			
	type	*array*		
	items	type	*array*	
		items	type	*integer*
• **channel_blocks**	Number of channel blocks to allocate to this sub-array beam. Maximum number of channel blocks = 48.			
	type	*array*		
	items	type	*integer*	
additionalProperties	False			

1.12.3 ska-low-mccs-releaseresources

Low MCCS resource release 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-mccs-releaseresources/1.0",
    "subarray_id": 1,
    "release_all": true
}
```

https://schema.skatelescope.org/ska-low-mccs-releaseresources/1.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **subarray_id**	ID of the MCCS sub-array which should release resources.	
	type	*integer*
• **release_all**	true to release all resources, false to release only the resources defined in this payload. Note: partial resource release for MCCS is not implemented and the identification of the resources to release is not yet part of the schema.	
	type	*boolean*
additionalProperties	False	

1.12.4 ska-low-mccs-configure

Low MCCS configure 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-mccs-configure/1.0",
    "stations": [{
        "station_id": 1
    }, {
        "station_id": 2
    }],
    "subarray_beams": [{
        "subarray_beam_id": 1,
        "station_ids": [1, 2],
        "update_rate": 0.0,
        "channels": [
            [0, 8, 1, 1],
            [8, 8, 2, 1],
            [24, 16, 2, 1]
        ],
        "sky_coordinates": [0.0, 180.0, 0.0, 45.0, 0.0],
        "antenna_weights": [1.0, 1.0, 1.0],
        "phase_centre": [0.0, 0.0]
    }]
}
```

<table>
<tr><td colspan="5">https://schema.skatelescope.org/ska-low-mccs-configure/1.0</td></tr>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td rowspan="2">• interface</td><td colspan="4">URI of JSON schema applicable to this JSON payload.</td></tr>
<tr><td>type</td><td colspan="3">string</td></tr>
<tr><td rowspan="6">• stations</td><td colspan="4">IDs of the MCCS stations to configure.
Maximum array size = 512, the maximum number of MCCS stations.</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="4">items</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td>• station_id</td><td>type</td><td>integer</td></tr>
<tr><td>additionalProperties</td><td colspan="2">True</td></tr>
<tr><td rowspan="6">• subarray_beams</td><td colspan="4">MCCS sub-array beam configuration.</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="4">items</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• subarray_beam_id</td><td colspan="2">ID of MCCS sub-array beam to configure.
ID must be an integer between 1 and 48.</td></tr>
<tr><td>type</td><td>integer</td></tr>
</table>

continues on next page

Table 19 – continued from previous page

		• **station_ids**	IDs of MCCS stations within this sub-array beamto configure. Array size must be less than 512, the maximum number of MCCS stations. Each item in the list must be an integer between 1 and 512.			
			type	*array*		
			items	type	*integer*	
		• **update_rate**	Update rate for pointing information. Value must be 0.0 or greater. TODO: clarify whether this is specified as a frequency or as a cadence, plus units.			
			type	*number*		
		• **channels**	Channel block configurations. Each item in the list is a channel block configuration, each specified as a list of 4 numbers as follows: [start channel, number of channels, beam index, sub-station index] Constraints are: 0 < start channel < 376 start channel must be a multiple of 8 8 < number of channels < 48 1 < beam index < 48 1 < sub-station index < 8			
			type	*array*		
			items	type	*array*	
				items	type	*integer*
		• **antenna_weights**	Antenna weights. Maximum array size = 512 (=256 antennas x2 pols per sub-array beam). Antennas signals can be weighted to modify the station beam, varying from 0.0 for full exclusion to potentially 256.0 for an antenna contribution compensated for the number of antennas in the beam. This value is an amplitude multiplier added to that antenna signal before adding into the sum. Weights apply to all channels assigned to a beam.			
			type	*array*		
			items	type	*number*	
		• **phase_centre**	Phase centre offset for the station beam, in metres. The reference position for station phase must be modified to reflect antenna weighting and their contribution to the station beam. This offset can be can considered the desired centre of mass for the station. Constraints: array size = 2 -20 < phase centre value < 20			
			type	*array*		
			items	type	*number*	
		• **sky_coordinates**	Azimuth/elevation of sub-array beam target, in degrees.			
			type	*array*		
			items	type	*number*	
		additionalProperties	False			
additionalProperties	False					

1.12.5 ska-low-mccs-scan

Low MCCS scan 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-mccs-scan/1.0",
    "scan_id": 1,
    "start_time": 0.0
}
```

https://schema.skatelescope.org/ska-low-mccs-scan/1.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **scan_id**	Scan ID to associate with the data. The scan ID and SBI ID are used together to uniquely associate the data taken with the telescope configuration in effect at the moment of observation.	
	type	*integer*
• **start_time**	Start time for the scan. Currently unused and can be set to 0.0.	
	type	*number*
additionalProperties	False	

1.12.6 ska-low-mccs-antenna-config

Antennas 1.0

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-telmodel-antenna/1.0",
    "type": "FeatureCollection",
    "name": "antenna_export_w2",
    "features": [{
        "interface": "https://schema.skao.int/ska-telmodel-antenna-features/1.0",
        "type": "Feature",
        "properties": {
            "interface": "https://schema.skao.int/ska-telmodel-antenna-features-
↪properties/1.0",
            "antenna_station_id": 0,
            "station_id": "object(534nfhwh2)",
            "x_pos": 6.1,
            "y_pos": 6.1,
            "z_pos": 6.1,
            "base_id": 1,
            "tpm_id": 1,
            "tpm_rx": 1,
            "status_x": "some status",
```

(continues on next page)

(continued from previous page)


```
            "status_y": "some status",
            "tpm_name": "Tpm 1",
            "delay_x": 5,
            "delay_y": 5,
            "station_num": 1
        },
        "geometry": {
            "interface": "https://schema.skao.int/ska-telmodel-antenna-features-geometry/
↪1.0",
            "type": "Point",
            "coordinates": [1.5, 6.2]
        }
    }, {
        "interface": "https://schema.skao.int/ska-telmodel-antenna-features/1.0",
        "type": "Feature",
        "properties": {
            "interface": "https://schema.skao.int/ska-telmodel-antenna-features-
↪properties/1.0",
            "antenna_station_id": 0,
            "station_id": "object(534nfhwh2)",
            "x_pos": 6.1,
            "y_pos": 6.1,
            "z_pos": 6.1,
            "base_id": 1,
            "tpm_id": 1,
            "tpm_rx": 1,
            "status_x": "some status",
            "status_y": "some status",
            "tpm_name": "Tpm 1",
            "delay_x": 5,
            "delay_y": 5,
            "station_num": 1
        },
        "geometry": {
            "interface": "https://schema.skao.int/ska-telmodel-antenna-features-geometry/
↪1.0",
            "type": "Point",
            "coordinates": [1.5, 6.2]
        }
    }]
}
```


Configuration data for antennas stored in geojson format

https://schema.skao.int/ska-telmodel-antenna/1.0		
type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **type**	Type	
	type	*string*
• **name**	Name	
	type	*string*
• **features**	Features	
	type	*array*
	items	Features of the antenna.
		Features 1.0
additionalProperties	False	

Features 1.0

Features of the antenna.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **type**	Type	
	type	*string*
• **properties**	Antenna properties	
	Properties 1.0	
• **geometry**	Antenna geometry	
	Geometry - type, coordinates 1.0	
additionalProperties	False	

Properties 1.0

The properties of the antenna

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **antenna_station_id**	Id of the antenna station	
	type	*integer*
• **station_id**	Id of the station	
	type	*string*
• **x_pos**	x position of the antenna	
	type	*number*
• **y_pos**	y position of the antenna	
	type	*number*
• **z_pos**	z position of the antenna	

continues on next page

Table 20 – continued from previous page

	type	*number*
• **base_id**	base id	
	type	*integer*
• **tpm_id**	Id of the TPM	
	type	*integer*
• **tpm_rx**	TPM receiver	
	type	*integer*
• **status_x**	Status x	
	type	*string*
• **status_y**	status y	
	type	*string*
• **tpm_name**	TPM name	
	type	*string*
• **delay_x**	delay in the x direction	
	type	*integer*
• **delay_y**	delay in the y direction	
	type	*integer*
• **station_num**	station number	
	type	*integer*
additionalProperties	False	

Geometry - type, coordinates 1.0

Postion of the antenna.

type	*object*		
properties			
• **interface**	Interface version		
	type	*string*	
• **type**	Coordinate type		
	type	*string*	
• **coordinates**	Array of coordinates		
	type	*array*	
	items	type	*number*
additionalProperties	False		

1.12.7 ska-low-mccs-station-config

stations 1.0

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-telmodel-station/1.0",
    "type": "FeatureCollection",
    "name": "station_export_w2",
    "features": [{
        "interface": "https://schema.skao.int/ska-telmodel-station-features/1.0",
        "type": "Feature",
```

(continues on next page)

(continued from previous page)


```
        "properties": {
            "interface": "https://schema.skao.int/ska-telmodel-station-features-
↪properties/1.0",
            "name": "Station 1",
            "nof_antennas": 256,
            "antenna_type": "EDA2",
            "tpms": {
                "0": 1,
                "1": 2,
                "2": 3,
                "3": 4
            },
            "station_num": 1
        },
        "geometry": {
            "interface": "https://schema.skao.int/ska-telmodel-station-features-geometry/
↪1.0",
            "type": "Point",
            "coordinates": [1.5, 6.2]
        }
    }, {
        "interface": "https://schema.skao.int/ska-telmodel-station-features/1.0",
        "type": "Feature",
        "properties": {
            "interface": "https://schema.skao.int/ska-telmodel-station-features-
↪properties/1.0",
            "name": "Station 1",
            "nof_antennas": 256,
            "antenna_type": "EDA2",
            "tpms": {
                "0": 1,
                "1": 2,
                "2": 3,
                "3": 4
            },
            "station_num": 1
        },
        "geometry": {
            "interface": "https://schema.skao.int/ska-telmodel-station-features-geometry/
↪1.0",
            "type": "Point",
            "coordinates": [1.5, 6.2]
        }
    }]
}
```


Configuration data for stations stored in geojson format

https://schema.skao.int/ska-telmodel-station/1.0		
type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **type**	Type	
	type	*string*
• **name**	Name	
	type	*string*
• **features**	Features	
	type	*array*
	items	Features of the station.
		Features 1.0
additionalProperties	False	

Features 1.0

Features of the station.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **type**	Type	
	type	*string*
• **properties**	station properties	
	Properties 1.0	
• **geometry**	station geometry	
	Geometry - type, coordinates 1.0	
additionalProperties	False	

Properties 1.0

The properties of the station

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **name**	name of station	
	type	*string*
• **nof_antennas**	number of antennas on station	
	type	*integer*
• **antenna_type**	type of antenna	
	type	*string*
• **tpms**	tiles	
• **station_num**	station number	
	type	*integer*
additionalProperties	False	

Geometry - type, coordinates 1.0

Postion of the station.

type	*object*		
properties			
• **interface**	Interface version		
	type	*string*	
• **type**	Coordinate type		
	type	*string*	
• **coordinates**	Array of coordinates		
	type	*array*	
	items	type	*number*
additionalProperties	False		

1.13 Mid CBF schemas

Schemas used for commands to the CSP Mid.CBF.

See Mid.CBF Controller and Subarray command documentation for documentation of all commands.

1.13.1 ska-mid-cbf-initsysparam

MID.CBF Parameters 1.0

Example (MID.CBF Parameters)

```
{
    "interface": "https://schema.skao.int/ska-midcbf-initsysparam/1.0",
    "dish_parameters": {
        "SKA001": {
            "vcc": 1,
```

(continues on next page)

(continued from previous page)

```
            "k": 11
        },
        "SKA100": {
            "vcc": 2,
            "k": 101
        },
        "SKA036": {
            "vcc": 3,
            "k": 1127
        },
        "SKA063": {
            "vcc": 4,
            "k": 620
        }
    }
}
```

Example (MID.CBF Parameters Source URI)

```
{
    "interface": "https://schema.skao.int/ska-mid-cbf-initsysparam/1.0",
    "tm_data_sources": ["car://gitlab.com/ska-telescope/ska-telmodel-data?1.0.0#tmdata"],
    "tm_data_filepath": "instrument/ska1_mid_psi/ska-mid-cbf-system-parameters.json"
}
```

https://schema.skao.int/ska-mid-cbf-initsysparam/1.0	
anyOf	*mid-cbf parameters 1.0*
	mid-cbf parameters source URI 1.0

mid-cbf parameters 1.0

type	*object*	
properties		
• **interface**	URI of JSON schema for this command's JSON payload.	
	type	*string*
• **dish_parameters**	Dish parameters section containing the information needed to map each dish ID to its initialization parameters, including the vcc ID and offset-index k value.	
	dish mapping 1.0	
additionalProperties	False	

dish mapping 1.0

type	*object*
properties	
• **dish ID**	At least one dish ID must be specified, and each dish ID must be a valid ID. Valid dish IDs include: SKA dishes: "SKAnnn", where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: "MKTnnn", where nnn is a zero padded integer in the range of 000 to 063.
	dish mapping details 1.0
additionalProperties	False

dish mapping details 1.0

type	*object*	
properties		
• **vcc**	The VCC ID for the given dish ID. Range: [1-197]	
	type	*integer*
• **k**	The offset-index k value for the dish ID. Range: [1-2222]	
	type	*integer*
additionalProperties	False	

mid-cbf parameters source URI 1.0

type	*object*		
properties			
• **interface**	URI of JSON schema for this command's JSON payload.		
	type	*string*	
• **tm_data_sources**	The telmodel data source. This parameter must be provided as a list containing a single entry.		
	type	*array*	
	items	type	*string*
• **tm_data_filepath**	Path to the JSON file containing the dish parameters required to execute the Mid CBF InitSysParam command.		
	type	*string*	
	pattern	^\S+\.json$	
additionalProperties	False		

1.14 Science Data Processor schemas

Schemas used for commands to / attributes from the SDP LMC. See SDP LMC subarray documentation for an overview of the interactions.

1.14.1 ska-sdp-assignres

SDP assign resources 0.4

Example


```
{
    "execution_block": {
        "eb_id": "eb-mvp01-20210623-00000",
        "max_length": 100.0,
        "context": {},
        "beams": [{
            "beam_id": "vis0",
            "function": "visibilities"
        }, {
            "beam_id": "pss1",
            "search_beam_id": 1,
            "function": "pulsar search"
        }, {
            "beam_id": "pss2",
            "search_beam_id": 2,
            "function": "pulsar search"
        }, {
            "beam_id": "pst1",
            "timing_beam_id": 1,
            "function": "pulsar timing"
        }, {
            "beam_id": "pst2",
            "timing_beam_id": 2,
            "function": "pulsar timing"
        }, {
            "beam_id": "vlbi1",
            "vlbi_beam_id": 1,
            "function": "vlbi"
        }],
        "scan_types": [{
            "scan_type_id": ".default",
            "beams": {
                "vis0": {
                    "channels_id": "vis_channels",
                    "polarisations_id": "all"
                },
                "pss1": {
                    "field_id": "pss_field_0",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
```


(continues on next page)

(continued from previous page)


```
                "pss2": {
                    "field_id": "pss_field_1",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "pst1": {
                    "field_id": "pst_field_0",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "pst2": {
                    "field_id": "pst_field_1",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "vlbi": {
                    "field_id": "vlbi_field",
                    "channels_id": "vlbi_channels",
                    "polarisations_id": "all"
                }
            }
        }, {
            "scan_type_id": "target:a",
            "derive_from": ".default",
            "beams": {
                "vis0": {
                    "field_id": "field_a"
                }
            }
        }],
        "channels": [{
            "channels_id": "vis_channels",
            "spectral_windows": [{
                "spectral_window_id": "fsp_1_channels",
                "count": 744,
                "start": 0,
                "stride": 2,
                "freq_min": 350000000.0,
                "freq_max": 368000000.0,
                "link_map": [
                    [0, 0],
                    [200, 1],
                    [744, 2],
                    [944, 3]
                ]
            }, {
                "spectral_window_id": "fsp_2_channels",
                "count": 744,
                "start": 2000,
                "stride": 1,
                "freq_min": 360000000.0,
                "freq_max": 368000000.0,
```


(continues on next page)

(continued from previous page)

```
                "link_map": [
                    [2000, 4],
                    [2200, 5]
                ]
            }, {
                "spectral_window_id": "zoom_window_1",
                "count": 744,
                "start": 4000,
                "stride": 1,
                "freq_min": 360000000.0,
                "freq_max": 361000000.0,
                "link_map": [
                    [4000, 6],
                    [4200, 7]
                ]
            }]
        }, {
            "channels_id": "pulsar_channels",
            "spectral_windows": [{
                "spectral_window_id": "pulsar_fsp_channels",
                "count": 744,
                "start": 0,
                "freq_min": 350000000.0,
                "freq_max": 368000000.0
            }]
        }],
        "polarisations": [{
            "polarisations_id": "all",
            "corr_type": ["XX", "XY", "YY", "YX"]
        }],
        "fields": [{
            "field_id": "field_a",
            "phase_dir": {
                "ra": [123, 0.1],
                "dec": [80, 0.1],
                "reference_time": "...",
                "reference_frame": "ICRF3"
            },
            "pointing_fqdn": "low-tmc/telstate/0/pointing"
        }]
    },
    "processing_blocks": [{
        "pb_id": "pb-mvp01-20210623-00000",
        "sbi_ids": ["sbi-mvp01-20200325-00001"],
        "script": {
            "kind": "realtime",
            "name": "vis_receive",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "pb_id": "pb-mvp01-20210623-00001",
```

(continues on next page)

(continued from previous page)


```
        "sbi_ids": ["sbi-mvp01-20200325-00001"],
        "script": {
            "kind": "realtime",
            "name": "test_realtime",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "pb_id": "pb-mvp01-20210623-00002",
        "sbi_ids": ["sbi-mvp01-20200325-00002"],
        "script": {
            "kind": "batch",
            "name": "ical",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20210623-00000",
            "kind": ["visibilities"]
        }]
    }, {
        "pb_id": "pb-mvp01-20210623-00003",
        "sbi_ids": ["sbi-mvp01-20200325-00001", "sbi-mvp01-20200325-00002"],
        "script": {
            "kind": "batch",
            "name": "dpreb",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20210623-00002",
            "kind": ["calibration"]
        }]
    }],
    "resources": {
        "csp_links": [1, 2, 3, 4],
        "receptors": ["FS4", "FS8", "FS16", "FS17", "FS22", "FS23", "FS30", "FS31", "FS32
↪", "FS33", "FS36", "FS52", "FS56", "FS57", "FS59", "FS62", "FS66", "FS69", "FS70",
↪"FS72", "FS73", "FS78", "FS80", "FS88", "FS89", "FS90", "FS91", "FS98", "FS108", "FS111
↪", "FS132", "FS144", "FS146", "FS158", "FS165", "FS167", "FS176", "FS183", "FS193",
↪"FS200", "FS345", "FS346", "FS347", "FS348", "FS349", "FS350", "FS351", "FS352", "FS353
↪", "FS354", "FS355", "FS356", "FS429", "FS430", "FS431", "FS432", "FS433", "FS434",
↪"FS465", "FS466", "FS467", "FS468", "FS469", "FS470"],
        "receive_nodes": 10
    }
}
```


Used for assigning resources to an SDP subarray.

As concrete resource usage for the SDP depend strongly on the underlying processing script, this fully parameterises all processing blocks to be executed. This especially means that in contrast to most other sub-systems, SDP processing deployments might persist across scans (and scan configuration) boundaries.

<table>
<tr><td colspan="6">https://schema.skao.int/ska-sdp-assignres/0.4</td></tr>
<tr><td>type</td><td colspan="5">object</td></tr>
<tr><td colspan="6">properties</td></tr>
<tr><td>• interface</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td>• transaction_id</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td>pattern</td><td colspan="4">^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td>• execution_block</td><td colspan="5">Execution block</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td colspan="5">Execution block 0.4</td></tr>
<tr><td>• resources</td><td colspan="5">External resources</td></tr>
<tr><td></td><td>type</td><td colspan="4">object</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td colspan="5">properties</td></tr>
<tr><td></td><td>• receptors</td><td>type</td><td colspan="3">array</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="3">null</td></tr>
<tr><td></td><td></td><td>items</td><td>anyOf</td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^[ENS]([1-9]|1[0-6])-[1-6]$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^MKT0([0-5][0-9]|6[0-3])$</td></tr>
<tr><td></td><td>additionalProperties</td><td colspan="4">True</td></tr>
<tr><td>• processing_blocks</td><td colspan="5">Processing blocks</td></tr>
<tr><td></td><td>type</td><td colspan="4">array</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
</table>

continues on next page

Table 21 – continued from previous page

	items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.
		Processing block 0.4
additionalProperties	False	

Execution block 0.4

type	*object*			
properties				
• **eb_id**	type	*string*		
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
• **max_length**	type	*number*		
• **context**	Free-form information from OET, see ADR-54			
• **beams**	Beam parameters			
	type	*array*		
	items	Beam parameters for the purpose of the Science Data Processor.		
		Beam 0.4		
• **scan_types**	Scan types. Associates scans with per-beam fields & channel configurations			
	type	*array*		
	items	type	*object*	
		properties		
		• **scan_type_id**	const	(any scan type)
		• derive_from	type	*string*
		• beams	type	*object*
		additionalProperties	False	
• **channels**	Channels			
	type	*array*		
	items	Spectral windows per channel configuration.		

continues on next page

Table 22 – continued from previous page

		Scan channels 0.4			
• **polarisations**	Polarisation definitions				
	type	*array*			
	items	Polarisation definition.			
		type	*object*		
		properties			
		• **polarisations_id**	type	*string*	
		• **corr_type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• **fields**	Fields / targets				
	type	*array*			
	items	Fields / Targets			
		type	*object*		
		properties			
		• **field_id**	type	*string*	
		• **phase_dir**	Phase direction		
			type	*object*	
			properties		
			• **ra**	type	*array*
				items	
			• **dec**	type	*array*
				items	
			• **reference_time**	type	*string*
			• **reference_frame**	const	ICRF3
			additionalProperties	False	
		• pointing_fqdn	type	*string*	
		additionalProperties	False		
additionalProperties	False				

Beam 0.4

Beam parameters for the purpose of the Science Data Processor.

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• beam_id</td><td colspan="2">Name to identify the beam within the SDP configuration.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• function</td><td colspan="2">Identifies the type and origin of the generated beam data. This corresponds to a certain kind of calibration or receive functionality SDP is meant to provide for it.
Possible options:
• visibilities: Correlated voltages from CBF used for calibration and imaging
• pulsar search: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar search data products
• pulsar timing: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar timing data products
• vlbi: SDP provides calibrations for tied-array beam
• transient buffer: SDP receives and delivers transient buffer data dumps</td></tr>
<tr><td>enum</td><td>visibilities, pulsar search, pulsar timing, vlbi, transient buffer</td></tr>
<tr><td rowspan="2">• search_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• timing_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• vlbi_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

Scan channels 0.4

Spectral windows per channel configuration.

<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td colspan="2">• channels_id</td></tr>
<tr><td>• spectral_windows</td><td>
<table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td>
<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td colspan="2">• spectral_window_id</td></tr>
<tr><td>• count</td><td>Number of channels<table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• start</td><td>First channel ID<table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• stride</td><td>Distance between subsequent channel IDs<table><tr><td>type</td><td>integer</td></tr><tr><td>default</td><td>null</td></tr></table></td></tr>
<tr><td>• freq_min</td><td>Lower bound of first channel<table><tr><td>type</td><td>number</td></tr></table></td></tr>
<tr><td>• freq_max</td><td>Upper bound of last channel<table><tr><td>type</td><td>number</td></tr></table></td></tr>
<tr><td>• link_map</td><td>Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.<table><tr><td>type</td><td>array</td></tr><tr><td>default</td><td>null</td></tr><tr><td colspan="2">items</td></tr></table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table>

Processing block 0.4

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• pb_id</td><td>Unique identifier for this processing block.<table><tr><td>type</td><td>string</td></tr><tr><td>pattern</td><td>^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr></table></td></tr>
<tr><td>• script</td><td>Specification of the workflow to be executed along with configuration parameters for the workflow.
<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• kind</td><td>The kind of processing script (realtime or batch)<table><tr><td>allOf</td><td><table><tr><td>type</td><td>string</td></tr><tr><td>enum</td><td>realtime, batch</td></tr></table></td></tr></table></td></tr>
<tr><td>• name</td><td>The name of the processing script</td></tr>
</table>
</td></tr>
</table>

continues on next page

Table 23 – continued from previous page

		type	*string*		
	• **version**	Version of the processing script. Uses semantic versioning.			
		type	*string*		
	additionalProperties	False			
• parameters	Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.				
	type	*object*			
	default	null			
• dependencies	A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that 1. The dependent processing block might only be able to start once the dependency has been fulfilled 2. Data associated with the dependency must be kept alive until the dependent processing block is finished. As processing blocks might have many different outputs, the dependency "kind" can be used to specify how this dependency is meant to be interpreted (e.g. "visibilities", "calibration"…)				
	type	*array*			
	default	null			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **kind**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• sbi_ids	Scheduling block instances that the processing block belongs to.				
	type	*array*			
	default	null			
	items	type	*string*		
		pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
additionalProperties	False				

SDP assign resources 0.3

Example

```
{
    "eb_id": "eb-mvp01-20210623-00000",
    "max_length": 100.0,
    "scan_types": [{
        "scan_type_id": "science",
        "reference_frame": "ICRS",
        "ra": "02:42:40.771",
        "dec": "-00:00:47.84",
        "channels": [{
            "count": 744,
```

(continues on next page)

(continued from previous page)

```
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [0, 0],
                [200, 1],
                [744, 2],
                [944, 3]
            ]
        }, {
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 360000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [2000, 4],
                [2200, 5]
            ]
        }]
    }, {
        "scan_type_id": "calibration",
        "reference_frame": "ICRS",
        "ra": "12:29:06.699",
        "dec": "02:03:08.598",
        "channels": [{
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [0, 0],
                [200, 1],
                [744, 2],
                [944, 3]
            ]
        }, {
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 360000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [2000, 4],
                [2200, 5]
            ]
        }]
    }],
    "processing_blocks": [{
        "pb_id": "pb-mvp01-20210623-00000",
```

(continues on next page)

(continued from previous page)

```
        "workflow": {
            "kind": "realtime",
            "name": "vis_receive",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "pb_id": "pb-mvp01-20210623-00001",
        "workflow": {
            "kind": "realtime",
            "name": "test_realtime",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "pb_id": "pb-mvp01-20210623-00002",
        "workflow": {
            "kind": "batch",
            "name": "ical",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20210623-00000",
            "kind": ["visibilities"]
        }]
    }, {
        "pb_id": "pb-mvp01-20210623-00003",
        "workflow": {
            "kind": "batch",
            "name": "dpreb",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20210623-00002",
            "kind": ["calibration"]
        }]
    }]
}
```

Used for assigning resources to an SDP subarray.

As concrete resource usage for the SDP depend strongly on the underlying processing script, this fully parameterises all processing blocks to be executed. This especially means that in contrast to most other sub-systems, SDP processing deployments might persist across scans (and scan configuration) boundaries.

https://schema.skao.int/ska-sdp-assignres/0.3		
type	*object*	
properties		
• interface	type	*string*
	default	null
• transaction_id	type	*string*
	pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
	default	null
• **eb_id**	Execution block ID to associate with processing	
	type	*string*
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• max_length	Hint about the maximum observation length to support by the SDP. Used for ensuring that enough buffer capacity is available to capture measurements. Resources assignment might fail if we do not have enough space to guarantee that all data could be captured.	
	type	*number*
	default	null
• **scan_types**	Scan types to be supported on subarray	
	type	*array*
	items	A scan configuration for SDP. Once AssignResources has been performed successfully, subsequent Configure commands can select from these scan types in order to coordinate SDP with other sub-systems participating in the observation - for instance to switch between targets, or perform special calibration scans.
		Scan type 0.3
• **processing_blocks**	type	*array*
	items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.
		Processing block 0.3
additionalProperties	False	

Scan type 0.3

A scan configuration for SDP. Once AssignResources has been performed successfully, subsequent Configure commands can select from these scan types in order to coordinate SDP with other sub-systems participating in the observation - for instance to switch between targets, or perform special calibration scans.

type	*object*		
properties			
• **scan_type_id**	const	(any scan type)	
• reference_frame	Specification of the reference frame or system for a set of pointing coordinates (see ADR-49)		
	default	null	
	allOf	type	*string*
		const	ICRS
• ra	Right Ascension in degrees (see ADR-49)		
	type	*string*	
	default	null	
• dec	Declination in degrees (see ADR-49)		
	type	*string*	
	default	null	
• channels	type	*array*	
	default	null	
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.	
		Scan channels 0.3	
additionalProperties	False		

Scan channels 0.3

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

Processing block 0.3

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

type	*object*				
properties					
• **pb_id**	Unique identifier for this processing block.				
	type	*string*			
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
• **workflow**	Specification of the workflow to be executed along with configuration parameters for the workflow.				
	type	*object*			
	properties				
	• **kind**	The kind of processing script (realtime or batch)			
		allOf	type	*string*	
			enum	realtime, batch	
	• **name**	The name of the processing script			
		type	*string*		
	• **version**	Version of the processing script. Uses semantic versioning.			
		type	*string*		
	additionalProperties	False			
• parameters	Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.				
	type	*object*			
	default	null			
• dependencies	A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that 1. The dependent processing block might only be able to start once the dependency has been fulfilled 2. Data associated with the dependency must be kept alive until the dependent processing block is finished. As processing blocks might have many different outputs, the dependency "kind" can be used to specify how this dependency is meant to be interpreted (e.g. "visibilities", "calibration"…)				
	type	*array*			
	default	null			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **kind**	type	*array*	
			items	type	*string*
		additionalProperties	False		
additionalProperties	False				

SDP assign resources 0.2

Example


```
{
    "id": "sbi-mvp01-20200325-00001",
    "max_length": 100.0,
    "scan_types": [{
        "id": "science",
        "coordinate_system": "ICRS",
        "ra": "02:42:40.771",
        "dec": "-00:00:47.84",
        "channels": [{
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [0, 0],
                [200, 1],
                [744, 2],
                [944, 3]
            ]
        }, {
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 360000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [2000, 4],
                [2200, 5]
            ]
        }]
    }, {
        "id": "calibration",
        "coordinate_system": "ICRS",
        "ra": "12:29:06.699",
        "dec": "02:03:08.598",
        "channels": [{
            "count": 744,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [0, 0],
                [200, 1],
                [744, 2],
                [944, 3]
            ]
        }, {
```


(continues on next page)

(continued from previous page)

```
            "count": 744,
            "start": 2000,
            "stride": 1,
            "freq_min": 360000000.0,
            "freq_max": 368000000.0,
            "link_map": [
                [2000, 4],
                [2200, 5]
            ]
        }]
    }],
    "processing_blocks": [{
        "id": "pb-mvp01-20200325-00001",
        "workflow": {
            "type": "realtime",
            "id": "vis_receive",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "id": "pb-mvp01-20200325-00002",
        "workflow": {
            "type": "realtime",
            "id": "test_realtime",
            "version": "0.1.0"
        },
        "parameters": {}
    }, {
        "id": "pb-mvp01-20200325-00003",
        "workflow": {
            "type": "batch",
            "id": "ical",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20200325-00001",
            "type": ["visibilities"]
        }]
    }, {
        "id": "pb-mvp01-20200325-00004",
        "workflow": {
            "type": "batch",
            "id": "dpreb",
            "version": "0.1.0"
        },
        "parameters": {},
        "dependencies": [{
            "pb_id": "pb-mvp01-20200325-00003",
            "type": ["calibration"]
        }]
    }]
```

(continues on next page)

(continued from previous page)

```
}
```

https://schema.skao.int/ska-sdp-assignres/0.2		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*string*
	pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• max_length	type	*number*
• **scan_types**	Scan types to be supported on subarray	
	type	*array*
	items	*Scan type 0.2*
• **processing_blocks**	type	*array*
	items	*Processing block 0.2*
additionalProperties	False	

Scan type 0.2

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.2
additionalProperties	False	

Scan channels 0.2

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

Processing block 0.2

type	*object*				
properties					
• **id**	type	*string*			
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
• **workflow**	type	*object*			
	properties				
	• **type**	type	*string*		
	• **id**	type	*string*		
	• **version**	type	*string*		
	additionalProperties	True			
• parameters	type	*object*			
• dependencies	type	*array*			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
additionalProperties	False				

SDP assign resources 0.1

https://schema.skao.int/ska-sdp-assignres/0.1		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*string*
	pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• max_length	type	*number*
• **scan_types**	Scan types to be supported on subarray	
	type	*array*
	items	*Scan type 0.1*
• **processing_blocks**	type	*array*
	items	*Processing block 0.1*
additionalProperties	False	

Scan type 0.1

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.1
additionalProperties	False	

Scan channels 0.1

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

Processing block 0.1

type	*object*				
properties					
• **id**	type	*string*			
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
• **workflow**	type	*object*			
	properties				
	• **type**	type	*string*		
	• **id**	type	*string*		
	• **version**	type	*string*		
	additionalProperties	True			
• parameters	type	*object*			
• dependencies	type	*array*			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
additionalProperties	False				

SDP assign resources 0.0

https://schema.skao.int/ska-sdp-assignres/0.0		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*string*
	pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• max_length	type	*number*
• **scan_types**	Scan types to be supported on subarray	
	type	*array*
	items	*Scan type 0.0*
• **processing_blocks**	type	*array*
	items	*Processing block 0.0*
additionalProperties	False	

Scan type 0.0

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.0
additionalProperties	False	

Scan channels 0.0

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

Processing block 0.0

type	*object*				
properties					
• **id**	type	*string*			
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
• **workflow**	type	*object*			
	properties				
	• **type**	type	*string*		
	• **id**	type	*string*		
	• **version**	type	*string*		
	additionalProperties	True			
• parameters	type	*object*			
• dependencies	type	*array*			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
additionalProperties	False				

1.14.2 ska-sdp-configure

SDP configure 0.4

Example

```
{
    "scan_type": "science"
}
```

Configures an SDP subarray for a number of scans of a certain previously-assigned type. See resource assignment.

<table>
<tr><td colspan="5">https://schema.skao.int/ska-sdp-configure/0.4</td></tr>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td rowspan="2">• interface</td><td>type</td><td colspan="3">string</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="3">• transaction_id</td><td>type</td><td colspan="3">string</td></tr>
<tr><td>pattern</td><td colspan="3">^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td>• scan_type</td><td>type</td><td colspan="3">string</td></tr>
<tr><td rowspan="8">• new_scan_types</td><td>type</td><td colspan="3">array</td></tr>
<tr><td>default</td><td colspan="3">null</td></tr>
<tr><td rowspan="6">items</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td>• scan_type_id</td><td>const</td><td>(any scan type)</td></tr>
<tr><td>• derive_from</td><td>type</td><td>string</td></tr>
<tr><td>• beams</td><td>type</td><td>object</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
<tr><td>additionalProperties</td><td colspan="4">False</td></tr>
</table>

SDP configure 0.3

Example

```
{
    "scan_type": "science"
}
```

Example with new scan types

```
{
    "new_scan_types": [{
        "scan_type_id": "new_calibration",
        "channels": [{
            "count": 372,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 358000000.0,
            "link_map": [
                [0, 0],
                [200, 1]
            ]
        }]
    }],
    "scan_type": "new_calibration"
}
```

Configures an SDP subarray for a number of scans of a certain previously-assigned type. See resource assignment.

https://schema.skao.int/ska-sdp-configure/0.3		
type	*object*	
properties		
• interface	type	*string*
	default	null
• transaction_id	type	*string*
	pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
	default	null
• **scan_type**	type	*string*
• new_scan_types	type	*array*
	default	null
	items	A scan configuration for SDP. Once AssignResources has been performed successfully, subsequent Configure commands can select from these scan types in order to coordinate SDP with other sub-systems participating in the observation - for instance to switch between targets, or perform special calibration scans.
		Scan type 0.3
additionalProperties	False	

Scan type 0.3

A scan configuration for SDP. Once AssignResources has been performed successfully, subsequent Configure commands can select from these scan types in order to coordinate SDP with other sub-systems participating in the observation - for instance to switch between targets, or perform special calibration scans.

type	*object*		
properties			
• **scan_type_id**	const	(any scan type)	
• reference_frame	Specification of the reference frame or system for a set of pointing coordinates (see ADR-49)		
	default	null	
	allOf	type	*string*
		const	ICRS
• ra	Right Ascension in degrees (see ADR-49)		
	type	*string*	
	default	null	
• dec	Declination in degrees (see ADR-49)		
	type	*string*	
	default	null	
• channels	type	*array*	
	default	null	
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.	
		Scan channels 0.3	
additionalProperties	False		

Scan channels 0.3

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

SDP configure 0.2

Example

```
{
    "scan_type": "science"
}
```

Example with new scan types

```
{
    "new_scan_types": [{
        "id": "new_calibration",
        "channels": [{
            "count": 372,
            "start": 0,
            "stride": 2,
            "freq_min": 350000000.0,
            "freq_max": 358000000.0,
            "link_map": [
                [0, 0],
                [200, 1]
            ]
        }]
    }],
    "scan_type": "new_calibration"
}
```

https://schema.skao.int/ska-sdp-configure/0.2		
type	*object*	
properties		
• interface	type	*string*
• **scan_type**	type	*string*
• new_scan_types	type	*array*
	items	*Scan type 0.2*
additionalProperties	False	

Scan type 0.2

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.2
additionalProperties	False	

Scan channels 0.2

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

SDP configure 0.1

https://schema.skao.int/ska-sdp-configure/0.1		
type	*object*	
properties		
• interface	type	*string*
• **scan_type**	type	*string*
• new_scan_types	type	*array*
	items	*Scan type 0.1*
additionalProperties	False	

Scan type 0.1

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.1
additionalProperties	False	

Scan channels 0.1

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

SDP configure 0.0

https://schema.skao.int/ska-sdp-configure/0.0		
type	*object*	
properties		
• interface	type	*string*
• **scan_type**	type	*string*
• new_scan_types	type	*array*
	items	*Scan type 0.0*
additionalProperties	False	

Scan type 0.0

type	*object*	
properties		
• **id**	const	(any scan type)
• coordinate_system	const	ICRS
• ra	type	*string*
• dec	type	*string*
• channels	type	*array*
	items	Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.
		Scan channels 0.0
additionalProperties	False	

Scan channels 0.0

Informs SDP ingest about the expected channel configuration, especially which frequencies are expected to be mapped to which channel ID. Note that channel IDs are not guaranteed to be continuous, so this might involve gaps and/or strides.

type	*object*	
properties		
• **count**	Number of channels	
	type	*integer*
• **start**	First channel ID	
	type	*integer*
• stride	Distance between subsequent channel IDs	
	type	*integer*
	default	null
• **freq_min**	Lower bound of first channel	
	type	*number*
• **freq_max**	Upper bound of last channel	
	type	*number*
• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
	type	*array*
	default	null
	items	
additionalProperties	False	

1.14.3 ska-sdp-scan

SDP scan 0.4

Example

```json
{
    "scan_id": 1
}
```

Indicates to SDP that a new scan is about to start

https://schema.skao.int/ska-sdp-scan/0.4		
type	*object*	
properties		
• interface	type	*string*
• transaction_id	type	*string*
	pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• **scan_id**	ID associated with new scan	
	type	*integer*
additionalProperties	False	

SDP scan 0.3

Example

```json
{
    "scan_id": 1
}
```

Indicates to SDP that a new scan is about to start

https://schema.skao.int/ska-sdp-scan/0.3		
type	*object*	
properties		
• interface	type	*string*
• transaction_id	type	*string*
	pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• **scan_id**	ID associated with new scan	
	type	*integer*
additionalProperties	False	

SDP scan 0.2

Example

```
{
    "id": 1
}
```

https://schema.skao.int/ska-sdp-scan/0.2		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*integer*
additionalProperties	False	

SDP scan 0.1

https://schema.skao.int/ska-sdp-scan/0.1		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*integer*
additionalProperties	False	

SDP scan 0.0

https://schema.skao.int/ska-sdp-scan/0.0		
type	*object*	
properties		
• interface	type	*string*
• **id**	type	*integer*
additionalProperties	False	

1.14.4 ska-sdp-recvaddrs

SDP receive addresses map 0.5

Example


```
{
    "science": {
        "vis0": {
            "function": "visibilities",
            "host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"],
                [744, "192.168.0.3"],
                [1144, "192.168.0.4"]
            ],
            "port": [
                [0, 9000, 1],
                [400, 9000, 1],
                [744, 9000, 1],
                [1144, 9000, 1]
            ],
            "mac": [
                [0, "06-00-00-00-00-00"],
                [744, "06-00-00-00-00-01"]
            ],
            "delay_cal": "low-sdp/telstate/rcal0/delay",
            "pointing_cal": "tango://low-sdp/queueconnector/01/pointing_offsets"
        },
        "pss1": {
            "function": "pulsar search",
            "search_beam_id": 1,
            "host": [
                [0, "192.168.60.0"]
            ],
            "port": [
                [0, 8000]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pss2": {
            "function": "pulsar search",
            "search_beam_id": 2,
            "host": [
                [0, "192.168.60.1"]
            ],
            "port": [
                [0, 8000]
```


(continues on next page)

(continued from previous page)


```
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pst1": {
            "function": "pulsar timing",
            "timing_beam_id": 1,
            "host": [
                [0, "192.168.60.2"]
            ],
            "port": [
                [0, 8001]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pst2": {
            "function": "pulsar timing",
            "timing_beam_id": 2,
            "host": [
                [0, "192.168.60.3"]
            ],
            "port": [
                [0, 8002]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        }
    },
    "calibration": {
        "vis0": {
            "function": "visibilities",
            "host": [
                [0, "192.168.1.1"]
            ],
            "port": [
                [0, 9000, 1]
            ],
            "delay_cal": "low-sdp/telstate/rcal0/delay",
            "pointing_cal": "tango://low-sdp/queueconnector/01/pointing_offsets"
```


(continues on next page)

(continued from previous page)

```
        },
        "pss1": {
            "function": "pulsar search",
            "search_beam_id": 1,
            "host": [
                [0, "192.168.60.0"]
            ],
            "port": [
                [0, 8003]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pss2": {
            "function": "pulsar search",
            "search_beam_id": 2,
            "host": [
                [0, "192.168.60.1"]
            ],
            "port": [
                [0, 8002]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pst1": {
            "function": "pulsar timing",
            "timing_beam_id": 0,
            "host": [
                [0, "192.168.60.2"]
            ],
            "port": [
                [0, 8001]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pst2": {
            "function": "pulsar timing",
            "timing_beam_id": 1,
            "host": [
                [0, "192.168.60.3"]
            ],
            "port": [
                [0, 8000]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
```

(continues on next page)

(continued from previous page)

```
        }
    }
}
```

Provides information about receive node addresses to use for ingesting measurement data to SDP (such as visibility SPEAD streams).

Receive addresses consists of a map of scan type to a receive address map. This address map must be set once the SDP subarray finishes the transition following `AssignResources` (i.e. IDLE following the current state of ADR-8). TMC will then check SDP's subarray `receiveAddresses` attribute when preparing to configure elements for a certain scan type.

Note that this has been changed to use the more compact channel map format defined in ADR-4. The general idea still applies: A map is given as a list, each entry of the format `[start_channel, value]`. The first entry specifies the first channel ID the map applies to. So in the example, the host for channels 0-399 is "192.168.0.1", while the host for channels 400-799 is "192.168.0.2" and so forth.

A minor extension applies to the port map, where every map entry is given as `[start_channel, start_value, increment]`. The true value for a channel is given from the applicable map entry by:

```
value = start_value + (channel - start_channel) * increment
```

So in the example, channels 0-399 should be sent to host "192.168.0.1" at ports 9000-9399, and channels 400-799 to host "192.168.0.2" at ports 9000-9399. If we had said `"port": [[0, 9000, 0]` all packets would be sent to the same port. Equally `"port": [[0, 9000, 2]` would indicate spacing the ports out by steps of 2.

Unused channel IDs should be ignored. This especially applies to unused gaps and channel ID strides possibly resulting from averaging at CBF. This means that with an averaging degree of 2 (see channelAveragingMap in ADR-4), only every second channel ID would be used in the example above.

https://schema.skao.int/ska-sdp-recvaddrs/0.5		
type	*object*	
properties		
• interface	type	*string*
• **(any scan type)**	Set of beams	
	type	*object*
	properties	
	• **(any beam type)**	Beam
		Beam receive addresses 0.5
	additionalProperties	False
additionalProperties	False	

Beam receive addresses 0.5

Receive addresses associated with a certain beam

type	*object*
properties	
• **host**	Destination host names (as channel map) Note that these are not currently guaranteed to be IP addresses, so a DNS resolution might be required.

continues on next page

Table 24 – continued from previous page

	type	*array*
	items	
• **port**	Destination ports (as channel map)	
	type	*array*
	items	
• mac	Destination MAC addresses (as channel map) Likely not going to be used, downstream systems should use ARP to determine the MAC address using `host` instead. See ADR-36	
	type	*array*
	default	null
	items	
• **function**	Type of beam configured. Beam identity is then given by the appropriate *beam_id* field.	
	enum	visibilities, pulsar search, pulsar timing, vlbi, transient buffer
• visibility_beam_id	Identifies visibility beam Might get omitted for SKA Mid, as it is assumed to have only one visibility beam.	
	type	*integer*
	default	null
• search_beam_id	Identifies pulsar search beam	
	type	*integer*
	default	null
• timing_beam_id	Identifies pulsar timing beam	
	type	*integer*
	default	null
• vlbi_beam_id	Identifies very long baseline interferometry beam	
	type	*integer*
	default	null
• search_window_id	Identifies search window for transient data capture	
	type	*integer*
	default	null
• jones_cal	Tango FQDNs serving real-time calibration Jones matrices for CBF	
	type	*array*
	default	null
	items	
• pointing_cal	Tango FQDNs serving pointing calibration offsets for TMC	
	type	*string*
	default	null
• delay_cal	Tango FQDNs serving gain/ delay calibration solutions for TMC	
	type	*string*
	default	null
additionalProperties	False	

SDP receive addresses map 0.4

Example


```
{
    "science": {
        "vis0": {
            "function": "visibilities",
            "host": [
                [0, "192.168.0.1"],
                [400, "192.168.0.2"],
                [744, "192.168.0.3"],
                [1144, "192.168.0.4"]
            ],
            "port": [
                [0, 9000, 1],
                [400, 9000, 1],
                [744, 9000, 1],
                [1144, 9000, 1]
            ],
            "mac": [
                [0, "06-00-00-00-00-00"],
                [744, "06-00-00-00-00-01"]
            ],
            "delay_cal": [
                [0, "low-sdp/telstate/rcal0/delay0"],
                [400, "low-sdp/telstate/rcal0/delay1"],
                [744, "low-sdp/telstate/rcal0/delay2"],
                [1144, "low-sdp/telstate/rcal0/delay2"]
            ]
        },
        "pss1": {
            "function": "pulsar search",
            "search_beam_id": 1,
            "host": [
                [0, "192.168.60.0"]
            ],
            "port": [
                [0, 8000]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pss2": {
            "function": "pulsar search",
            "search_beam_id": 2,
            "host": [
                [0, "192.168.60.1"]
            ],
```


(continues on next page)

(continued from previous page)


```
            "port": [
                [0, 8000]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pst1": {
            "function": "pulsar timing",
            "timing_beam_id": 1,
            "host": [
                [0, "192.168.60.2"]
            ],
            "port": [
                [0, 8001]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        },
        "pst2": {
            "function": "pulsar timing",
            "timing_beam_id": 2,
            "host": [
                [0, "192.168.60.3"]
            ],
            "port": [
                [0, 8002]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"],
                [400, "low-sdp/telstate/rcal0/jones1"],
                [744, "low-sdp/telstate/rcal0/jones2"],
                [1144, "low-sdp/telstate/rcal0/jones2"]
            ]
        }
    },
    "calibration": {
        "vis0": {
            "function": "visibilities",
            "host": [
                [0, "192.168.1.1"]
            ],
            "port": [
                [0, 9000, 1]
            ],
```


(continues on next page)

(continued from previous page)


```
            "delay_cal": [
                [0, "low-sdp/telstate/rcal0/delay0"]
            ]
        },
        "pss1": {
            "function": "pulsar search",
            "search_beam_id": 1,
            "host": [
                [0, "192.168.60.0"]
            ],
            "port": [
                [0, 8003]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pss2": {
            "function": "pulsar search",
            "search_beam_id": 2,
            "host": [
                [0, "192.168.60.1"]
            ],
            "port": [
                [0, 8002]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pst1": {
            "function": "pulsar timing",
            "timing_beam_id": 0,
            "host": [
                [0, "192.168.60.2"]
            ],
            "port": [
                [0, 8001]
            ],
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        },
        "pst2": {
            "function": "pulsar timing",
            "timing_beam_id": 1,
            "host": [
                [0, "192.168.60.3"]
            ],
            "port": [
                [0, 8000]
            ],
```


(continues on next page)

(continued from previous page)

```
            "jones_cal": [
                [0, "low-sdp/telstate/rcal0/jones0"]
            ]
        }
    }
}
```

Provides information about receive node addresses to use for ingesting measurement data to SDP (such as visibility SPEAD streams).

Receive addresses consists of a map of scan type to a receive address map. This address map must be set once the SDP subarray finishes the transition following `AssignResources` (i.e. IDLE following the current state of ADR-8). TMC will then check SDP's subarray `receiveAddresses` attribute when preparing to configure elements for a certain scan type.

Note that this has been changed to use the more compact channel map format defined in ADR-4. The general idea still applies: A map is given as a list, each entry of the format `[start_channel, value]`. The first entry specifies the first channel ID the map applies to. So in the example, the host for channels 0-399 is "192.168.0.1", while the host for channels 400-799 is "192.168.0.2" and so forth.

A minor extension applies to the port map, where every map entry is given as `[start_channel, start_value, increment]`. The true value for a channel is given from the applicable map entry by:

```
value = start_value + (channel - start_channel) * increment
```

So in the example, channels 0-399 should be sent to host "192.168.0.1" at ports 9000-9399, and channels 400-799 to host "192.168.0.2" at ports 9000-9399. If we had said `"port": [[0, 9000, 0]` all packets would be sent to the same port. Equally `"port": [[0, 9000, 2]` would indicate spacing the ports out by steps of 2.

Unused channel IDs should be ignored. This especially applies to unused gaps and channel ID strides possibly resulting from averaging at CBF. This means that with an averaging degree of 2 (see channelAveragingMap in ADR-4), only every second channel ID would be used in the example above.

<table>
<tr><td colspan="3">https://schema.skao.int/ska-sdp-recvaddrs/0.4</td></tr>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td>• interface</td><td>type</td><td>string</td></tr>
<tr><td rowspan="6">• (any scan type)</td><td colspan="2">Set of beams</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td rowspan="2">• (any beam type)</td><td>Beam</td></tr>
<tr><td>Beam receive addresses 0.4</td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

Beam receive addresses 0.4

Receive addresses associated with a certain beam

type	*object*	
properties		
• **host**	Destination host names (as channel map) Note that these are not currently guaranteed to be IP addresses, so a DNS resolution might be required.	
	type	*array*
	items	
• **port**	Destination ports (as channel map)	
	type	*array*
	items	
• mac	Destination MAC addresses (as channel map) Likely not going to be used, downstream systems should use ARP to determine the MAC address using `host` instead. See ADR-36	
	type	*array*
	default	null
	items	
• **function**	Type of beam configured. Beam identity is then given by the appropriate *beam_id* field.	
	enum	visibilities, pulsar search, pulsar timing, vlbi, transient buffer
• visibility_beam_id	Identifies visibility beam Might get omitted for SKA Mid, as it is assumed to have only one visibility beam.	
	type	*integer*
	default	null
• search_beam_id	Identifies pulsar search beam	
	type	*integer*
	default	null
• timing_beam_id	Identifies pulsar timing beam	
	type	*integer*
	default	null
• vlbi_beam_id	Identifies very long baseline interferometry beam	
	type	*integer*
	default	null
• search_window_id	Identifies search window for transient data capture	
	type	*integer*
	default	null
• jones_cal	Tango FQDNs serving real-time calibration Jones matrices for CBF	
	type	*array*
	default	null
	items	
• delay_cal	Tango FQDNs serving gain/ delay calibration solutions for TMC	
	type	*array*
	default	null
	items	
additionalProperties	False	

SDP receive addresses map 0.3

Example

```
{
    "science": {
        "host": [
            [0, "192.168.0.1"],
            [400, "192.168.0.2"],
            [744, "192.168.0.3"],
            [1144, "192.168.0.4"]
        ],
        "mac": [
            [0, "06-00-00-00-00-00"],
            [744, "06-00-00-00-00-01"]
        ],
        "port": [
            [0, 9000, 1],
            [400, 9000, 1],
            [744, 9000, 1],
            [1144, 9000, 1]
        ]
    },
    "calibration": {
        "host": [
            [0, "192.168.1.1"]
        ],
        "port": [
            [0, 9000, 1]
        ]
    }
}
```

Provides information about receive node addresses to use for ingesting measurement data to SDP (such as visibility SPEAD streams).

Receive addresses consists of a map of scan type to a receive address map. This address map must be set once the SDP subarray finishes the transition following `AssignResources` (i.e. IDLE following the current state of ADR-8). TMC will then check SDP's subarray `receiveAddresses` attribute when preparing to configure elements for a certain scan type.

Note that this has been changed to use the more compact channel map format defined in ADR-4. The general idea still applies: A map is given as a list, each entry of the format `[start_channel, value]`. The first entry specifies the first channel ID the map applies to. So in the example, the host for channels 0-399 is "192.168.0.1", while the host for channels 400-799 is "192.168.0.2" and so forth.

A minor extension applies to the port map, where every map entry is given as `[start_channel, start_value, increment]`. The true value for a channel is given from the applicable map entry by:

```
value = start_value + (channel - start_channel) * increment
```

So in the example, channels 0-399 should be sent to host "192.168.0.1" at ports 9000-9399, and channels 400-799 to host "192.168.0.2" at ports 9000-9399. If we had said `"port": [[0, 9000, 0]` all packets would be sent to the same port. Equally `"port": [[0, 9000, 2]` would indicate spacing the ports out by steps of 2.

Unused channel IDs should be ignored. This especially applies to unused gaps and channel ID strides possibly resulting from averaging at CBF. This means that with an averaging degree of 2 (see channelAveragingMap in ADR-4), only every second channel ID would be used in the example above.

<table>
<tr><td colspan="4">https://schema.skao.int/ska-sdp-recvaddrs/0.3</td></tr>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• interface</td><td>type</td><td colspan="2">string</td></tr>
<tr><td rowspan="12">• (any scan type)</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="3">• host</td><td colspan="2">Destination host names (as channel map)
Note that these are not currently guaranteed to be IP addresses, so a DNS resolution might be required.</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td colspan="2">items</td></tr>
<tr><td rowspan="3">• mac</td><td colspan="2">Destination MAC addresses (as channel map)
Likely not going to be used, downstream systems should use ARP to determine the MAC address using <code>host</code> instead. See ADR-36</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td colspan="2">items</td></tr>
<tr><td rowspan="3">• port</td><td colspan="2">Destination ports (as channel map)</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td colspan="2">items</td></tr>
<tr><td>additionalProperties</td><td colspan="2">True</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

SDP receive addresses map 0.2

Example

```
{
    "science": {
        "host": [
            [0, "192.168.0.1"],
            [400, "192.168.0.2"],
            [744, "192.168.0.3"],
            [1144, "192.168.0.4"]
        ],
        "mac": [
            [0, "06-00-00-00-00-00"],
            [744, "06-00-00-00-00-01"]
        ],
        "port": [
            [0, 9000, 1],
            [400, 9000, 1],
            [744, 9000, 1],
            [1144, 9000, 1]
        ]
    },
    "calibration": {
```

(continues on next page)

(continued from previous page)

```
        "host": [
            [0, "192.168.1.1"]
        ],
        "port": [
            [0, 9000, 1]
        ]
    }
}
```

Provides information about receive node addresses to use for ingesting measurement data to SDP (such as visibility SPEAD streams).

Receive addresses consists of a map of scan type to a receive address map. This address map must be set once the SDP subarray finishes the transition following `AssignResources` (i.e. IDLE following the current state of ADR-8). TMC will then check SDP's subarray `receiveAddresses` attribute when preparing to configure elements for a certain scan type.

Note that this has been changed to use the more compact channel map format defined in ADR-4. The general idea still applies: A map is given as a list, each entry of the format `[start_channel, value]`. The first entry specifies the first channel ID the map applies to. So in the example, the host for channels 0-399 is "192.168.0.1", while the host for channels 400-799 is "192.168.0.2" and so forth.

A minor extension applies to the port map, where every map entry is given as `[start_channel, start_value, increment]`. The true value for a channel is given from the applicable map entry by:

```
value = start_value + (channel - start_channel) * increment
```

So in the example, channels 0-399 should be sent to host "192.168.0.1" at ports 9000-9399, and channels 400-799 to host "192.168.0.2" at ports 9000-9399. If we had said `"port": [[0, 9000, 0]` all packets would be sent to the same port. Equally `"port": [[0, 9000, 2]` would indicate spacing the ports out by steps of 2.

Unused channel IDs should be ignored. This especially applies to unused gaps and channel ID strides possibly resulting from averaging at CBF. This means that with an averaging degree of 2 (see channelAveragingMap in ADR-4), only every second channel ID would be used in the example above.

<table>
<tr><td colspan="5">https://schema.skao.int/ska-sdp-recvaddrs/0.2</td></tr>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td>• interface</td><td>type</td><td colspan="3">string</td></tr>
<tr><td rowspan="12">• (any scan type)</td><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="3">• host</td><td colspan="3">Destination host names (as channel map)
Note that these are not currently guaranteed to be IP addresses, so a DNS resolution might be required.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td colspan="3">items</td></tr>
<tr><td rowspan="3">• mac</td><td colspan="3">Destination MAC addresses (as channel map)
Likely not going to be used, downstream systems should use ARP to determine the MAC address using <code>host</code> instead. See ADR-36</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td colspan="3">items</td></tr>
<tr><td rowspan="3">• port</td><td colspan="3">Destination ports (as channel map)</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td colspan="3">items</td></tr>
<tr><td>additionalProperties</td><td colspan="3">True</td></tr>
<tr><td>additionalProperties</td><td colspan="4">False</td></tr>
</table>

SDP receive addresses 0.1

Example

```
{
    "scanId": 1,
    "totalChannels": 7,
    "receiveAddresses": [{
        "phaseBinId": 0,
        "fspId": 1,
        "hosts": [{
            "host": "192.168.0.0",
            "channels": [{
                "portOffset": 9153,
                "startChannel": 153,
                "numChannels": 1
            }, {
                "portOffset": 9273,
                "startChannel": 273,
                "numChannels": 1
            }, {
                "portOffset": 9313,
                "startChannel": 313,
                "numChannels": 1
            }, {
                "portOffset": 9529,
                "startChannel": 529,
```

(continues on next page)

(continued from previous page)

```
                "numChannels": 1
            }, {
                "portOffset": 9665,
                "startChannel": 665,
                "numChannels": 1
            }, {
                "portOffset": 9681,
                "startChannel": 681,
                "numChannels": 2
            }]
        }]
    }]
}
```

https://schema.skao.int/ska-sdp-recvaddrs/0.1								
type	*object*							
properties								
• interface	type	*string*						
• **scanId**	type	*integer*						
• **totalChannels**	type	*integer*						
• **receiveAddresses**	type	*array*						
	items	type	*object*					
		properties						
		• **phaseBinId**	type	*integer*				
		• **fspId**	type	*integer*				
		• **hosts**	type	*array*				
			items	type	*object*			
				properties				
				• **host**	type	*string*		
				• **channels**	type	*array*		
					items	type	*object*	
						properties		
						• **portOffset**	type	*integer*
						• **startChannel**	type	*integer*
						• **numChannels**	type	*integer*
						additionalProperties	True	
				additionalProperties	True			
		additionalProperties	True					
addition...	False							

SDP receive addresses 0.0

<table>
<tr><td colspan="2">https://schema.skao.int/ska-sdp-recvaddrs/0.0</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• interface</td><td><table><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>• scanId</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• totalChannels</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• receiveAddresses</td><td><table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td><table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• phaseBinId</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• fspId</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• hosts</td><td><table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td><table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• host</td><td><table><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>• channels</td><td><table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td><table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• portOffset</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• startChannel</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• numChannels</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>additionalProperties</td><td>True</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>additionalProperties</td><td>True</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>addition-alProper[...]</td><td>True</td></tr>
</table></td></tr>
</table></td></tr>
</table>

1.14.5 ska-sdp-releaseres

SDP release resources 0.4

Example


```
{
    "resources": {
        "csp_links": [1, 2, 3, 4],
        "receptors": ["FS4", "FS8", "FS16", "FS17", "FS22", "FS23", "FS30", "FS31", "FS32
↪", "FS33", "FS36", "FS52", "FS56", "FS57", "FS59", "FS62", "FS66", "FS69", "FS70",
↪"FS72", "FS73", "FS78", "FS80", "FS88", "FS89", "FS90", "FS91", "FS98", "FS108", "FS111
↪", "FS132", "FS144", "FS146", "FS158", "FS165", "FS167", "FS176", "FS183", "FS193",
↪"FS200", "FS345", "FS346", "FS347", "FS348", "FS349", "FS350", "FS351", "FS352", "FS353
↪", "FS354", "FS355", "FS356", "FS429", "FS430", "FS431", "FS432", "FS433", "FS434",
↪"FS465", "FS466", "FS467", "FS468", "FS469", "FS470"],
        "receive_nodes": 10
    }
}
```


Used for releasing resources for an SDP subarray.

<table>
<tr><td colspan="6">https://schema.skao.int/ska-sdp-releaseres/0.4</td></tr>
<tr><td>type</td><td colspan="5">object</td></tr>
<tr><td colspan="6">properties</td></tr>
<tr><td>• interface</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td>• transaction_id</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td>pattern</td><td colspan="4">^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td>• resources</td><td>type</td><td colspan="4">object</td></tr>
<tr><td></td><td colspan="5">properties</td></tr>
<tr><td></td><td>• receptors</td><td>type</td><td colspan="3">array</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="3">null</td></tr>
<tr><td></td><td></td><td>items</td><td>anyOf</td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^[ENS]([1-9]|1[0-6])-[1-6]$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$</td></tr>
<tr><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td>pattern</td><td>^MKT0([0-5][0-9]|6[0-3])$</td></tr>
<tr><td></td><td>additionalProperties</td><td colspan="4">True</td></tr>
<tr><td>additionalProperties</td><td colspan="5">False</td></tr>
</table>

1.15 Telescope Manager Control schemas

1.15.1 ska-low-tmc-assignresources

Low TMC assign resources 3.2

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-assignresources/3.2",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-assignres/0.4",
        "resources": {
            "receptors": ["SKA001", "SKA002", "SKA003", "SKA004"]
        },
        "execution_block": {
            "eb_id": "eb-test-20220916-00000",
            "context": {},
            "max_length": 3600.0,
            "beams": [{
                "beam_id": "vis0",
                "function": "visibilities"
            }],
            "scan_types": [{
                "scan_type_id": ".default",
                "beams": {
                    "vis0": {
                        "channels_id": "vis_channels",
                        "polarisations_id": "all"
                    }
                }
            }, {
                "scan_type_id": "target:a",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_a"
                    }
                }
            }, {
                "scan_type_id": "calibration:b",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_b"
                    }
                }
            }],
            "channels": [{
                "channels_id": "vis_channels",
                "spectral_windows": [{
                    "spectral_window_id": "fsp_1_channels",
```


(continues on next page)

(continued from previous page)


```
                    "count": 4,
                    "start": 0,
                    "stride": 2,
                    "freq_min": 350000000.0,
                    "freq_max": 368000000.0,
                    "link_map": [
                        [0, 0],
                        [200, 1],
                        [744, 2],
                        [944, 3]
                    ]
                }]
            }],
            "polarisations": [{
                "polarisations_id": "all",
                "corr_type": ["XX", "XY", "YX", "YY"]
            }],
            "fields": [{
                "field_id": "field_a",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }, {
                "field_id": "field_b",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }]
        },
        "processing_blocks": [{
            "pb_id": "pb-test-20220916-00000",
            "script": {
                "kind": "realtime",
                "name": "test-receive-addresses",
                "version": "0.5.0"
            },
            "sbi_ids": ["sbi-test-20220916-00000"],
            "parameters": {}
        }]
    }
}
```


https://schema.skao.int/ska-low-tmc-assignresources/3.2

continues on next page

Table 26 – continued from previous page

type	*object*					
properties						
• **interface**	URI of JSON schema applicable to this JSON payload.					
	type	*string*				
• transaction_id	A transaction id specific to the command					
	type	*string*				
	default	null				
• **subarray_id**	ID of sub-array targeted by this resource allocation request					
	type	*integer*				
• **mccs**	MCCS specification for resource allocation.					
	type	*object*				
	properties					
	• **subarray_beam_ids**	IDs of the MCCS sub-array beams to allocate to this subarray. Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.				
		type	*array*			
		items	type	*integer*		
	• **station_ids**	IDs of MCCS stations to allocate to this sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.				
		type	*array*			
		items	type	*array*		
			items	type	*integer*	
	• **channel_blocks**	Number of channel blocks to allocate to this sub-array beam. Maximum number of channel blocks = 48.				
		type	*array*			
		items	type	*integer*		
	additionalProperties	False				
• **sdp**	SDP configuration specification					
	type	*object*				
	properties					
	• interface	type	*string*			
		default	null			
	• transaction_id	type	*string*			
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
		default	null			
	• execution_block	Execution block				
		default	null			
		Execution block 0.4				
	• resources	External resources				
		type	*object*			
		default	null			
		properties				
		• receptors	type	*array*		
			default	null		
			items	anyOf	type	*string*
					pattern	^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$

continues on next page

Table 26 – continued from previous page

					type	*string*
					pattern	^[ENS]([1-9]|1[0-6])-[1-6]$
					type	*string*
					pattern	^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$
					type	*string*
					pattern	^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$
					type	*string*
					pattern	^MKT0([0-5][0-9]|6[0-3])$
		additional-Properties	True			
	• processing_blocks	Processing blocks				
		type	*array*			
		default	null			
		items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.			
			Processing block 0.4			
	additional-Properties	False				
• csp	CSP configuration specification					
	type	*object*				
	default	null				
	properties					
	• **interface**	CSP Interface configuration				
		type	*string*			
	• **common**	type	*object*			
		properties				
		• **subarray_id**	type	*integer*		

continues on next page

Table 26 – continued from previous page

		additional-Properties	True
	• **lowcbf**	type	*object*
		properties	
		additional-Properties	True
	additional-Properties	False	
additional-Properties	False		

Execution block 0.4

type	*object*			
properties				
• **eb_id**	type	*string*		
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
• **max_length**	type	*number*		
• **context**	Free-form information from OET, see ADR-54			
• **beams**	Beam parameters			
	type	*array*		
	items	Beam parameters for the purpose of the Science Data Processor.		
		Beam 0.4		
• **scan_types**	Scan types. Associates scans with per-beam fields & channel configurations			
	type	*array*		
	items	type	*object*	
		properties		
		• **scan_type_id**	type	*string*
		• de-rive_from	type	*string*
		• beams	type	*object*
		additionalProp-erties	False	
• **channels**	Channels			
	type	*array*		
	items	Spectral windows per channel configuration.		
		Scan channels 0.4		
• **polarisa-tions**	Polarisation definitions			

continues on next page

Table 27 – continued from previous page

	type	*array*			
	items	Polarisation definition.			
		type	*object*		
		properties			
		• **polarisations_id**	type	*string*	
		• **corr_type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• **fields**	Fields / targets				
	type	*array*			
	items	Fields / Targets			
		type	*object*		
		properties			
		• **field_id**	type	*string*	
		• **phase_dir**	Phase direction		
			type	*object*	
			properties		
			• **ra**	type	*array*
				items	
			• **dec**	type	*array*
				items	
			• **reference_time**	type	*string*
			• **reference_frame**	const	ICRF3
			additionalProperties	False	
		• pointing_fqdn	type	*string*	
		additionalProperties	False		
additionalProperties	False				

Beam 0.4

Beam parameters for the purpose of the Science Data Processor.

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• beam_id</td><td colspan="2">Name to identify the beam within the SDP configuration.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• function</td><td colspan="2">Identifies the type and origin of the generated beam data. This corresponds to a certain kind of calibration or receive functionality SDP is meant to provide for it.
Possible options:
• visibilities: Correlated voltages from CBF used for calibration and imaging
• pulsar search: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar search data products
• pulsar timing: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar timing data products
• vlbi: SDP provides calibrations for tied-array beam
• transient buffer: SDP receives and delivers transient buffer data dumps</td></tr>
<tr><td>enum</td><td>visibilities, pulsar search, pulsar timing, vlbi, transient buffer</td></tr>
<tr><td rowspan="2">• search_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• timing_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• vlbi_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

Scan channels 0.4

Spectral windows per channel configuration.

<table>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td colspan="5">• channels_id</td></tr>
<tr><td rowspan="13">• spectral_windows</td><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="12">items</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td colspan="3">• spectral_window_id</td></tr>
<tr><td>• count</td><td colspan="2">Number of channels
type: integer</td></tr>
<tr><td>• start</td><td colspan="2">First channel ID
type: integer</td></tr>
<tr><td>• stride</td><td colspan="2">Distance between subsequent channel IDs
type: integer
default: null</td></tr>
<tr><td>• freq_min</td><td colspan="2">Lower bound of first channel
type: number</td></tr>
<tr><td>• freq_max</td><td colspan="2">Upper bound of last channel
type: number</td></tr>
<tr><td rowspan="4">• link_map</td><td colspan="2">Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.</td></tr>
<tr><td>type</td><td>array</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td colspan="2">items</td></tr>
<tr><td></td><td></td><td>additionalProperties</td><td colspan="2">False</td></tr>
<tr><td>additionalProperties</td><td colspan="4">False</td></tr>
</table>

Processing block 0.4

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

<table>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td rowspan="3">• pb_id</td><td colspan="4">Unique identifier for this processing block.</td></tr>
<tr><td>type</td><td colspan="3">string</td></tr>
<tr><td>pattern</td><td colspan="3">^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td rowspan="6">• script</td><td colspan="4">Specification of the workflow to be executed along with configuration parameters for the workflow.</td></tr>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• kind</td><td colspan="3">The kind of processing script (realtime or batch)</td></tr>
<tr><td>allOf</td><td colspan="2">type: string
enum: realtime, batch</td></tr>
<tr><td>• name</td><td colspan="3">The name of the processing script</td></tr>
</table>

continues on next page

Table 28 – continued from previous page

		type	*string*		
	• **version**	Version of the processing script. Uses semantic versioning.			
		type	*string*		
	additionalProperties	False			
• parameters	Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.				
	type	*object*			
	default	null			
• dependencies	A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that 1. The dependent processing block might only be able to start once the dependency has been fulfilled 2. Data associated with the dependency must be kept alive until the dependent processing block is finished. As processing blocks might have many different outputs, the dependency "kind" can be used to specify how this dependency is meant to be interpreted (e.g. "visibilities", "calibration"…)				
	type	*array*			
	default	null			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **kind**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• sbi_ids	Scheduling block instances that the processing block belongs to.				
	type	*array*			
	default	null			
	items	type	*string*		
		pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
additionalProperties	False				

Low TMC assign resources 3.1

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-assignresources/3.1",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
```

(continues on next page)

(continued from previous page)

```
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-assignres/0.4",
        "resources": {
            "receptors": ["SKA001", "SKA002"]
        },
        "execution_block": {
            "eb_id": "eb-test-20220916-00000",
            "context": {},
            "max_length": 3600.0,
            "beams": [{
                "beam_id": "vis0",
                "function": "visibilities"
            }],
            "scan_types": [{
                "scan_type_id": ".default",
                "beams": {
                    "vis0": {
                        "channels_id": "vis_channels",
                        "polarisations_id": "all"
                    }
                }
            }, {
                "scan_type_id": "target:a",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_a"
                    }
                }
            }, {
                "scan_type_id": "calibration:b",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_b"
                    }
                }
            }],
            "channels": [{
                "channels_id": "vis_channels",
                "spectral_windows": [{
                    "spectral_window_id": "fsp_1_channels",
                    "count": 4,
                    "start": 0,
                    "stride": 2,
                    "freq_min": 350000000.0,
                    "freq_max": 368000000.0,
                    "link_map": [
                        [0, 0],
                        [200, 1],
                        [744, 2],
```

(continues on next page)

(continued from previous page)


```
                        [944, 3]
                    ]
                }]
            }],
            "polarisations": [{
                "polarisations_id": "all",
                "corr_type": ["XX", "XY", "YX", "YY"]
            }],
            "fields": [{
                "field_id": "field_a",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }, {
                "field_id": "field_b",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }]
        },
        "processing_blocks": [{
            "pb_id": "pb-test-20220916-00000",
            "script": {
                "kind": "realtime",
                "name": "test-receive-addresses",
                "version": "0.5.0"
            },
            "sbi_ids": ["sbi-test-20220916-00000"],
            "parameters": {}
        }]
    }
}
```


<table>
<tr><td colspan="3">https://schema.skao.int/ska-low-tmc-assignresources/3.1</td></tr>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• interface</td><td colspan="2">URI of JSON schema applicable to this JSON payload.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="3">• transaction_id</td><td colspan="2">A transaction id specific to the command</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• subarray_id</td><td colspan="2">ID of sub-array targeted by this resource allocation request</td></tr>
<tr><td>type</td><td>integer</td></tr>
</table>

continues on next page

Table 29 – continued from previous page

<table>
<tr><td>• mccs</td><td colspan="6">MCCS specification for resource allocation.</td></tr>
<tr><td></td><td>type</td><td colspan="5">object</td></tr>
<tr><td></td><td colspan="6">properties</td></tr>
<tr><td></td><td>• subarray_beam_ids</td><td colspan="5">IDs of the MCCS sub-array beams to allocate to this subarray.
Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.</td></tr>
<tr><td></td><td></td><td>type</td><td colspan="4">array</td></tr>
<tr><td></td><td></td><td>items</td><td>type</td><td colspan="3">integer</td></tr>
<tr><td></td><td>• station_ids</td><td colspan="5">IDs of MCCS stations to allocate to this sub-array beam.
Each ID must be between 1 and 512, the maximum number of stations.</td></tr>
<tr><td></td><td></td><td>type</td><td colspan="4">array</td></tr>
<tr><td></td><td></td><td>items</td><td>type</td><td colspan="3">array</td></tr>
<tr><td></td><td></td><td></td><td>items</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td></td><td>• channel_blocks</td><td colspan="5">Number of channel blocks to allocate to this sub-array beam.
Maximum number of channel blocks = 48.</td></tr>
<tr><td></td><td></td><td>type</td><td colspan="4">array</td></tr>
<tr><td></td><td></td><td>items</td><td>type</td><td colspan="3">integer</td></tr>
<tr><td></td><td>additionalProperties</td><td colspan="5">False</td></tr>
<tr><td>• sdp</td><td colspan="6">SDP configuration specification</td></tr>
<tr><td></td><td>type</td><td colspan="5">object</td></tr>
<tr><td></td><td colspan="6">properties</td></tr>
<tr><td></td><td>• interface</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td>• transaction_id</td><td>type</td><td colspan="4">string</td></tr>
<tr><td></td><td></td><td>pattern</td><td colspan="4">^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td>• execution_block</td><td colspan="5">Execution block</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td></td><td colspan="5">Execution block 0.4</td></tr>
<tr><td></td><td>• resources</td><td colspan="5">External resources</td></tr>
<tr><td></td><td></td><td>type</td><td colspan="4">object</td></tr>
<tr><td></td><td></td><td>default</td><td colspan="4">null</td></tr>
<tr><td></td><td></td><td colspan="5">properties</td></tr>
<tr><td></td><td></td><td>• receptors</td><td>type</td><td colspan="3">array</td></tr>
<tr><td></td><td></td><td></td><td>default</td><td colspan="3">null</td></tr>
<tr><td></td><td></td><td></td><td>items</td><td>anyOf</td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>pattern</td><td>^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>pattern</td><td>^[ENS]([1-9]|1[0-6])-[1-6]$</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>type</td><td>string</td></tr>
</table>

continues on next page

Table 29 – continued from previous page

					pattern	^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$
					type	*string*
					pattern	^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$
					type	*string*
					pattern	^MKT0([0-5][0-9]|6[0-3])$
		additional-Properties	True			
	• processing_blocks	Processing blocks				
		type	*array*			
		default	null			
		items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.			
			Processing block 0.4			
	additional-Properties	False				
• csp	CSP configuration specification					
	type	*object*				
	default	null				
	properties					
	• **interface**	CSP Interface configuration				
		type	*string*			
	• **common**	type	*object*			
		properties				
		• **subarray_id**	type	*integer*		
		additional-Properties	True			
	• **lowcbf**	type	*object*			
		properties				
		additional-Properties	True			

continues on next page

Table 29 – continued from previous page

	additional-Properties	False
additional-Properties	False	

Execution block 0.4

<table>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td colspan="5">properties</td></tr>
<tr><td rowspan="2">• eb_id</td><td>type</td><td colspan="3">string</td></tr>
<tr><td>pattern</td><td colspan="3">^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td>• max_length</td><td>type</td><td colspan="3">number</td></tr>
<tr><td>• context</td><td colspan="4">Free-form information from OET, see ADR-54</td></tr>
<tr><td rowspan="4">• beams</td><td colspan="4">Beam parameters</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="2">items</td><td colspan="3">Beam parameters for the purpose of the Science Data Processor.</td></tr>
<tr><td colspan="3">Beam 0.4</td></tr>
<tr><td rowspan="8">• scan_types</td><td colspan="4">Scan types. Associates scans with per-beam fields & channel configurations</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="6">items</td><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td>• scan_type_id</td><td>type</td><td>string</td></tr>
<tr><td>• derive_from</td><td>type</td><td>string</td></tr>
<tr><td>• beams</td><td>type</td><td>object</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
<tr><td rowspan="4">• channels</td><td colspan="4">Channels</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="2">items</td><td colspan="3">Spectral windows per channel configuration.</td></tr>
<tr><td colspan="3">Scan channels 0.4</td></tr>
<tr><td rowspan="5">• polarisations</td><td colspan="4">Polarisation definitions</td></tr>
<tr><td>type</td><td colspan="3">array</td></tr>
<tr><td rowspan="3">items</td><td colspan="3">Polarisation definition.</td></tr>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
</table>

continues on next page

Table 30 – continued from previous page

		• **polarisations_id**	type	*string*	
		• **corr_type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• **fields**	Fields / targets				
	type	*array*			
	items	Fields / Targets			
		type	*object*		
		properties			
		• **field_id**	type	*string*	
		• **phase_dir**	Phase direction		
			type	*object*	
			properties		
			• **ra**	type	*array*
				items	
			• **dec**	type	*array*
				items	
			• **reference_time**	type	*string*
			• **reference_frame**	const	ICRF3
			additionalProperties	False	
		• pointing_fqdn	type	*string*	
		additionalProperties	False		
additionalProperties	False				

Beam 0.4

Beam parameters for the purpose of the Science Data Processor.

<table>
<tr><td>type</td><td colspan="2">object</td></tr>
<tr><td colspan="3">properties</td></tr>
<tr><td rowspan="2">• beam_id</td><td colspan="2">Name to identify the beam within the SDP configuration.</td></tr>
<tr><td>type</td><td>string</td></tr>
<tr><td rowspan="2">• function</td><td colspan="2">Identifies the type and origin of the generated beam data. This corresponds to a certain kind of calibration or receive functionality SDP is meant to provide for it.
Possible options:
• visibilities: Correlated voltages from CBF used for calibration and imaging
• pulsar search: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar search data products
• pulsar timing: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar timing data products
• vlbi: SDP provides calibrations for tied-array beam
• transient buffer: SDP receives and delivers transient buffer data dumps</td></tr>
<tr><td>enum</td><td>visibilities, pulsar search, pulsar timing, vlbi, transient buffer</td></tr>
<tr><td rowspan="2">• search_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• timing_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td rowspan="2">• vlbi_beam_id</td><td>type</td><td>integer</td></tr>
<tr><td>default</td><td>null</td></tr>
<tr><td>additionalProperties</td><td colspan="2">False</td></tr>
</table>

Scan channels 0.4

Spectral windows per channel configuration.

type	*object*			
properties				
• **channels_id**				
• **spectral_windows**	type	*array*		
	items	type	*object*	
		properties		
		• **spectral_window_id**		
		• **count**	Number of channels	
			type	*integer*
		• **start**	First channel ID	
			type	*integer*
		• stride	Distance between subsequent channel IDs	
			type	*integer*
			default	null
		• **freq_min**	Lower bound of first channel	
			type	*number*
		• **freq_max**	Upper bound of last channel	
			type	*number*
		• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
			type	*array*
			default	null
			items	
		additionalProperties	False	
additionalProperties	False			

Processing block 0.4

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

type	*object*			
properties				
• **pb_id**	Unique identifier for this processing block.			
	type	*string*		
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
• **script**	Specification of the workflow to be executed along with configuration parameters for the workflow.			
	type	*object*		
	properties			
	• **kind**	The kind of processing script (realtime or batch)		
		allOf	type	*string*
			enum	realtime, batch
	• **name**	The name of the processing script		

continues on next page

Table 31 – continued from previous page

		type	*string*		
	• **version**	Version of the processing script. Uses semantic versioning.			
		type	*string*		
	additionalProperties	False			
• parameters	Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.				
	type	*object*			
	default	null			
• dependencies	A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that 1. The dependent processing block might only be able to start once the dependency has been fulfilled 2. Data associated with the dependency must be kept alive until the dependent processing block is finished. As processing blocks might have many different outputs, the dependency "kind" can be used to specify how this dependency is meant to be interpreted (e.g. "visibilities", "calibration"…)				
	type	*array*			
	default	null			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **kind**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• sbi_ids	Scheduling block instances that the processing block belongs to.				
	type	*array*			
	default	null			
	items	type	*string*		
		pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
additionalProperties	False				

Low TMC assign resources 3.0

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-assignresources/3.0",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
```

(continues on next page)

(continued from previous page)


```
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-assignres/0.4",
        "resources": {
            "receptors": ["SKA001", "SKA002", "SKA003", "SKA004"]
        },
        "execution_block": {
            "eb_id": "eb-test-20220916-00000",
            "context": {},
            "max_length": 3600.0,
            "beams": [{
                "beam_id": "vis0",
                "function": "visibilities"
            }],
            "scan_types": [{
                "scan_type_id": ".default",
                "beams": {
                    "vis0": {
                        "channels_id": "vis_channels",
                        "polarisations_id": "all"
                    }
                }
            }, {
                "scan_type_id": "target:a",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_a"
                    }
                }
            }, {
                "scan_type_id": "calibration:b",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_b"
                    }
                }
            }],
            "channels": [{
                "channels_id": "vis_channels",
                "spectral_windows": [{
                    "spectral_window_id": "fsp_1_channels",
                    "count": 4,
                    "start": 0,
                    "stride": 2,
                    "freq_min": 350000000.0,
                    "freq_max": 368000000.0,
                    "link_map": [
                        [0, 0],
                        [200, 1],
                        [744, 2],
```


(continues on next page)

(continued from previous page)


```
                        [944, 3]
                    ]
                }]
            }],
            "polarisations": [{
                "polarisations_id": "all",
                "corr_type": ["XX", "XY", "YX", "YY"]
            }],
            "fields": [{
                "field_id": "field_a",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }, {
                "field_id": "field_b",
                "phase_dir": {
                    "ra": [123.0],
                    "dec": [-60.0],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "..."
            }]
        },
        "processing_blocks": [{
            "pb_id": "pb-test-20220916-00000",
            "script": {
                "kind": "realtime",
                "name": "test-receive-addresses",
                "version": "0.6.1"
            },
            "sbi_ids": ["sbi-test-20220916-00000"],
            "parameters": {
                "time-to-ready": 5
            }
        }]
    },
    "csp": {
        "interface": "https://schema.skao.int/ska-low-csp-assignresources/2.0",
        "common": {
            "subarray_id": 1
        },
        "lowcbf": {
            "resources": [{
                "device": "fsp_01",
                "shared": true,
                "fw_image": "pst",
                "fw_mode": "unused"
```


(continues on next page)

(continued from previous page)

```
            }, {
                "device": "p4_01",
                "shared": true,
                "fw_image": "p4.bin",
                "fw_mode": "p4"
            }]
        }
    }
}
```

https://schema.skao.int/ska-low-tmc-assignresources/3.0					
type	*object*				
properties					
• **interface**	URI of JSON schema applicable to this JSON payload.				
	type	*string*			
• transaction_id	A transaction id specific to the command				
	type	*string*			
	default	null			
• **subarray_id**	ID of sub-array targeted by this resource allocation request				
	type	*integer*			
• **mccs**	MCCS specification for resource allocation.				
	type	*object*			
	properties				
	• **subarray_beam_ids**	IDs of the MCCS sub-array beams to allocate to this subarray. Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.			
		type	*array*		
		items	type	*integer*	
	• **station_ids**	IDs of MCCS stations to allocate to this sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.			
		type	*array*		
		items	type	*array*	
			items	type	*integer*
	• **channel_blocks**	Number of channel blocks to allocate to this sub-array beam. Maximum number of channel blocks = 48.			
		type	*array*		
		items	type	*integer*	
	additionalProperties	False			
• **sdp**	SDP configuration specification				
	type	*object*			
	properties				
	• interface	type	*string*		
		default	null		
	• transaction_id	type	*string*		
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
		default	null		
	• execution_block	Execution block			
		default	null		
		Execution block 0.4			

continues on next page

Table 32 – continued from previous page

	• resources	External resources				
		type	*object*			
		default	null			
		properties				
		• receptors	type	*array*		
			default	null		
			items	anyOf	type	*string*
					pattern	^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$
					type	*string*
					pattern	^[ENS]([1-9]|1[0-6])-[1-6]$
					type	*string*
					pattern	^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$
					type	*string*
					pattern	^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$
					type	*string*
					pattern	^MKT0([0-5][0-9]|6[0-3])$
		additionalProperties	True			
	• processing_blocks	Processing blocks				
		type	*array*			
		default	null			
		items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.			
			Processing block 0.4			
	additionalProperties	False				
• csp	CSP configuration specification					

continues on next page

Table 32 – continued from previous page

	type	*object*		
	default	null		
	properties			
	• **interface**	CSP Interface configuration		
		type	*string*	
	• **common**	type	*object*	
		properties		
		• **subarray_id**	type	*integer*
		additionalProperties	True	
	• **lowcbf**	type	*object*	
		properties		
		additionalProperties	True	
	additionalProperties	False		
additionalProperties	False			

Execution block 0.4

type	*object*			
properties				
• **eb_id**	type	*string*		
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
• **max_length**	type	*number*		
• **context**	Free-form information from OET, see ADR-54			
• **beams**	Beam parameters			
	type	*array*		
	items	Beam parameters for the purpose of the Science Data Processor.		
		Beam 0.4		
• **scan_types**	Scan types. Associates scans with per-beam fields & channel configurations			
	type	*array*		
	items	type	*object*	
		properties		
		• **scan_type_id**	type	*string*

continues on next page

Table 33 – continued from previous page

		• derive_from	type	*string*	
		• beams	type	*object*	
		additionalProperties	False		
• **channels**	Channels				
	type	*array*			
	items	Spectral windows per channel configuration.			
		Scan channels 0.4			
• **polarisations**	Polarisation definitions				
	type	*array*			
	items	Polarisation definition.			
		type	*object*		
		properties			
		• **polarisations_id**	type	*string*	
		• **corr_type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• **fields**	Fields / targets				
	type	*array*			
	items	Fields / Targets			
		type	*object*		
		properties			
		• **field_id**	type	*string*	
		• **phase_dir**	Phase direction		
			type	*object*	
			properties		
			• **ra**	type	*array*
				items	
			• **dec**	type	*array*
				items	
			• **reference_time**	type	*string*
			• **reference_frame**	const	ICRF3
			additionalProperties	False	

continues on next page

Table 33 – continued from previous page

		• pointing_fqdn	type	*string*
		additionalProperties	False	
additionalProperties	False			

Beam 0.4

Beam parameters for the purpose of the Science Data Processor.

type	*object*	
properties		
• **beam_id**	Name to identify the beam within the SDP configuration.	
	type	*string*
• **function**	Identifies the type and origin of the generated beam data. This corresponds to a certain kind of calibration or receive functionality SDP is meant to provide for it. Possible options: • *visibilities*: Correlated voltages from CBF used for calibration and imaging • *pulsar search*: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar search data products • *pulsar timing*: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar timing data products • *vlbi*: SDP provides calibrations for tied-array beam • *transient buffer*: SDP receives and delivers transient buffer data dumps	
	enum	visibilities, pulsar search, pulsar timing, vlbi, transient buffer
• search_beam_id	type	*integer*
	default	null
• timing_beam_id	type	*integer*
	default	null
• vlbi_beam_id	type	*integer*
	default	null
additionalProperties	False	

Scan channels 0.4

Spectral windows per channel configuration.

type	*object*			
properties				
• **channels_id**				
• **spectral_windows**	type	*array*		
	items	type	*object*	
		properties		
		• **spectral_window_id**		
		• **count**	Number of channels	
			type	*integer*
		• **start**	First channel ID	
			type	*integer*
		• stride	Distance between subsequent channel IDs	
			type	*integer*
			default	null
		• **freq_min**	Lower bound of first channel	
			type	*number*
		• **freq_max**	Upper bound of last channel	
			type	*number*
		• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
			type	*array*
			default	null
			items	
		additionalProperties	False	
additionalProperties	False			

Processing block 0.4

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

type	*object*	
properties		
• **pb_id**	Unique identifier for this processing block.	
	type	*string*
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$
• **script**	Specification of the workflow to be executed along with configuration parameters for the workflow.	
	type	*object*
	properties	

continues on next page

Table 34 – continued from previous page

<table>
<tr><td rowspan="8"></td><td rowspan="3">• kind</td><td colspan="4">The kind of processing script (realtime or batch)</td></tr>
<tr><td rowspan="2">allOf</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>enum</td><td colspan="2">realtime, batch</td></tr>
<tr><td rowspan="2">• name</td><td colspan="4">The name of the processing script</td></tr>
<tr><td>type</td><td colspan="3">string</td></tr>
<tr><td rowspan="2">• version</td><td colspan="4">Version of the processing script. Uses semantic versioning.</td></tr>
<tr><td>type</td><td colspan="3">string</td></tr>
<tr><td>additionalProperties</td><td colspan="4">False</td></tr>
<tr><td rowspan="3">• parameters</td><td colspan="5">Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.</td></tr>
<tr><td>type</td><td colspan="4">object</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="10">• dependencies</td><td colspan="5">A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that
1. The dependent processing block might only be able to start once the dependency has been fulfilled
2. Data associated with the dependency must be kept alive until the dependent processing block is finished.
As processing blocks might have many different outputs, the dependency “kind” can be used to specify how this dependency is meant to be interpreted (e.g. “visibilities”, “calibration”…)</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="7">items</td><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• pb_id</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td rowspan="2">• kind</td><td>type</td><td colspan="2">array</td></tr>
<tr><td>items</td><td>type</td><td>string</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
<tr><td rowspan="5">• sbi_ids</td><td colspan="5">Scheduling block instances that the processing block belongs to.</td></tr>
<tr><td>type</td><td colspan="4">array</td></tr>
<tr><td>default</td><td colspan="4">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td colspan="3">string</td></tr>
<tr><td>pattern</td><td colspan="3">^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td>additionalProperties</td><td colspan="5">False</td></tr>
</table>

Low TMC assign resources 2.0

Example JSON.

```
{
    "interface": "https://schema.skao.in/ska-low-tmc-assignresources/2.0",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "mccs": {
        "subarray_beam_ids": [1],
```

(continues on next page)

(continued from previous page)

```
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
    }
}
```

https://schema.skao.int/ska-low-tmc-assignresources/2.0			
type	*object*		
properties			
• **interface**	URI of JSON schema applicable to this JSON payload.		
	type	*string*	
• transaction_id	A transaction id specific to the command		
	type	*string*	
	default	null	
• **subarray_id**	ID of sub-array targeted by this resource allocation request		
	type	*integer*	
• **mccs**	MCCS specification for resource allocation.		
	type	*object*	
	properties		
	• **subarray_beam_ids**	IDs of the MCCS sub-array beams to allocate to this subarray. Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.	
		type: *array*	
		items: type	*integer*
	• **station_ids**	IDs of MCCS stations to allocate to this sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.	
		type: *array*	
		items: type: *array*	items: type: *integer*
	• **channel_blocks**	Number of channel blocks to allocate to this sub-array beam. Maximum number of channel blocks = 48.	
		type: *array*	
		items: type	*integer*
	additionalProperties	False	
additionalProperties	False		

Low TMC assign resources 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-tmc-assignresources/1.0",
    "subarray_id": 1,
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
    }
}
```

https://schema.skatelescope.org/ska-low-tmc-assignresources/1.0					
type	*object*				
properties					
• **interface**	URI of JSON schema applicable to this JSON payload.				
	type	*string*			
• transaction_id	A transaction id specific to the command				
	type	*string*			
	default	null			
• **subarray_id**	ID of sub-array targeted by this resource allocation request				
	type	*integer*			
• **mccs**	MCCS specification for resource allocation.				
	type	*object*			
	properties				
	• **subarray_beam_ids**	IDs of the MCCS sub-array beams to allocate to this subarray. Each ID must be between 1 and 48, the maximum number of sub-array beams. As of PI10, only one MCCS sub-array beam can be configured per allocation request. Multiple beams must be allocated via multiple allocation requests.			
		type	*array*		
		items	type	*integer*	
	• **station_ids**	IDs of MCCS stations to allocate to this sub-array beam. Each ID must be between 1 and 512, the maximum number of stations.			
		type	*array*		
		items	type	*array*	
			items	type	*integer*
	• **channel_blocks**	Number of channel blocks to allocate to this sub-array beam. Maximum number of channel blocks = 48.			
		type	*array*		
		items	type	*integer*	
	additionalProperties	False			
additionalProperties	False				

1.15.2 ska-low-tmc-configure

Low TMC configure 3.1

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-low-tmc-configure/3.1",
    "transaction_id": "txn-....-00001",
    "mccs": {
        "stations": [{
            "station_id": 1
        }, {
            "station_id": 2
        }],
        "subarray_beams": [{
            "subarray_beam_id": 1,
            "station_ids": [1, 2],
            "update_rate": 0.0,
            "channels": [
                [0, 8, 1, 1],
                [8, 8, 2, 1],
                [24, 16, 2, 1]
            ],
            "antenna_weights": [1.0, 1.0, 1.0],
            "phase_centre": [0.0, 0.0],
            "target": {
                "reference_frame": "HORIZON",
                "target_name": "DriftScan",
                "az": 180.0,
                "el": 45.0
            }
        }]
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
        "scan_type": "science_A"
    },
    "csp": {
        "interface": "https://schema.skao.int/ska-low-csp-configure/0.0",
        "common": {
            "config_id": "sbi-mvp01-20200325-00001-science_A"
        },
        "lowcbf": {
            "stations": {
                "stns": [
                    [1, 1],
                    [2, 1],
                    [3, 1],
                    [4, 1],
                    [5, 1],
                    [6, 1]
                ],
```


(continues on next page)

(continued from previous page)


```
                "stn_beams": [{
                    "stn_beam_id": 1,
                    "freq_ids": [400]
                }]
            },
            "vis": {
                "fsp": {
                    "function_mode": "vis",
                    "fsp_ids": [1]
                },
                "stn_beams": [{
                    "stn_beam_id": 1,
                    "host": [
                        [0, "192.168.1.00"]
                    ],
                    "port": [
                        [0, 9000, 1]
                    ],
                    "mac": [
                        [0, "02-03-04-0a-0b-0c"]
                    ],
                    "integration_ms": 849
                }]
            }
        }
    },
    "tmc": {
        "scan_duration": 10.0
    }
}
```


https://schema.skao.int/ska-low-tmc-configure/3.1				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• transaction_id	A transaction id specific to the command			
	type	*string*		
	default	null		
• **mccs**	MCCS configuration specification.			
	type	*object*		
	properties			
	• **stations**	IDs of the MCCS stations to configure. Maximum array size = 512, the maximum number of MCCS stations.		
		type	*array*	
		items	type	*object*
			properties	

continues on next page

Table 35 – continued from previous page

			• **station_id**	MCCS Station ID. Each ID must be between 1 and 512.			
				type	*integer*		
			additionalProperties	True			
	• **sub-array_beams**	MCCS sub-array beam configuration.					
		type	*array*				
		items	type	*object*			
			properties				
			• **sub-array_beam_id**	ID of MCCS sub-array beam to configure. ID must be an integer between 1 and 48.			
				type	*integer*		
			• **station_ids**	IDs of MCCS stations within this sub-array beamto configure. Array size must be less than 512, the maximum number of MCCS stations. Each item in the list must be an integer between 1 and 512.			
				type	*array*		
				items	type	*integer*	
			• **update_rate**	Update rate for pointing information. Value must be 0.0 or greater. TODO: clarify whether this is specified as a frequency or as a cadence, plus units.			
				type	*number*		
			• **channels**	Channel block configurations. Each item in the list is a channel block configuration, each specified as a list of 4 numbers as follows: [start channel, number of channels, beam index, sub-station index] Constraints are: 0 < start channel < 376 start channel must be a multiple of 8 8 < number of channels < 48 1 < beam index < 48 1 < sub-station index < 8			
				type	*array*		
				items	type	*array*	
					items	type	*integer*
			• **antenna_weights**	Antenna weights. Maximum array size = 512 (=256 antennas x2 pols per sub-array beam). Antenna signals can be weighted to modify the station beam, varying from 0.0 for full exclusion to potentially 256.0 for an antenna contribution compensated for the number of antennas in the beam. This value is an amplitude multiplier added to that antenna signal before adding into the sum. Weights apply to all channels assigned to a beam.			
				type	*array*		
				items	type	*number*	

continues on next page

Table 35 – continued from previous page

			• **phase_centre**	Phase centre offset for the station beam, in metres. The reference position for station phase must be modified to reflect antenna weighting and their contribution to the station beam. This offset can be can considered the desired centre of mass for the station. Constraints: array size = 2 -20 < phase centre value < 20		
				type	*array*	
				items	type	*number*
			• **target**	Target position for the sub-array beam. Only drift scan targets are currently implemented by MCCS, hence only azimuth and elevation are specified.		
				type	*object*	
				properties		
				• **reference_frame**	Co-ordinate system. Must be HORIZON for drift scan.	
					type	*string*
				• **target_name**	Name of target.	
					type	*string*
				• **az**	Pointing azimuth in degrees.	
					type	*number*
				• **el**	Pointing elevation in degrees.	
					type	*number*
				additionalProperties	False	
			additionalProperties	False		
	additionalProperties	False				
• **csp**	CSP configuration specification.					
	type	*object*				
	properties					
	• **interface**	CSP Interface configuration				
		type	*string*			
	• **common**	CSP Common configuration specification				
		type	*object*			
		properties				
		• **config_id**	CSP config id configuration			
			type	*string*		
		additionalProperties	False			

continues on next page

Table 35 – continued from previous page

		• **lowcbf**	LOWCBF configuration specification.					
			type	*object*				
			properties					
			• **stations**	Subarray Stations and station beam descriptions				
				type	*object*			
				properties				
				• **stns**	type	*array*		
					items	type	*array*	
						items	type	*integer*
				• **stn_beams**	type	*array*		
					items	type	*object*	
						properties		
						• **stn_beam_id**	station beam id	
							type	*integer*
						• **freq_ids**	list of station beam frequency ids	
							type	*array*
							items	type *integer*
						additionalProperties	False	
				additionalProperties	False			
			• **vis**	Visibility output descriptions				
				type	*object*			
				properties				
				• **fsp**	FSPs used for correlation			
					type	*object*		
					properties			
					• **function_mode**	Function mode name		
						type	*string*	
					• **fsp_ids**	List of IDs (integer)		
						type	*array*	
						items	type	*integer*
					additionalProperties	False		
				• **stn_beams**	SDP visibility destinations			
					type	*array*		
					items	type	*object*	
						properties		
						• **stn_beam_id**	Station Beam ID	
							type	*integer*
						• **integration_ms**	milliseconds integration	
							type	*integer*

continues on next page

Table 35 – continued from previous page

					• **host**	SDP channel & IP Address				
						type	*array*			
						items	type	*array*		
							items	anyOf	type	*integer*
									type	*string*
					• **port**	SDP chan & UDP port, stride				
						type	*array*			
						items	type	*array*		
							items	type	*integer*	
					• mac	SDP channel & server MAC				
						type	*array*			
						default	null			
						items	type	*array*		
							items	anyOf	type	*integer*
									type	*string*
					additionalProperties	False				
			additionalProperties	False						
		additionalProperties	False							
	additionalProperties	False								
• **sdp**	SDP configuration specification.									
	type	*object*								
	properties									
	• interface	type	*string*							
		default	null							
	• transaction_id	type	*string*							
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$							
		default	null							
	• **scan_type**	type	*string*							
	• new_scan_types	type	*array*							
		default	null							
		items	type	*object*						
			properties							

continues on next page

Table 35 – continued from previous page

			• **scan_type_id**	const	(any scan type)
			• derive_from	type	*string*
			• beams	type	*object*
			additionalProperties	False	
	additionalProperties	False			
• tmc	TMC configuration specification.				
	type	*object*			
	default	null			
	properties				
	• **scan_duration**	Scan duration in seconds. Value must be >= 0.0			
		type	*number*		
	additionalProperties	True			
additionalProperties	False				

Low TMC configure 3.0

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-configure/3.0",
    "transaction_id": "txn-....-00001",
    "mccs": {
        "stations": [{
            "station_id": 1
        }, {
            "station_id": 2
        }],
        "subarray_beams": [{
```

(continues on next page)

(continued from previous page)

```
            "subarray_beam_id": 1,
            "station_ids": [1, 2],
            "update_rate": 0.0,
            "channels": [
                [0, 8, 1, 1],
                [8, 8, 2, 1],
                [24, 16, 2, 1]
            ],
            "antenna_weights": [1.0, 1.0, 1.0],
            "phase_centre": [0.0, 0.0],
            "target": {
                "reference_frame": "HORIZON",
                "target_name": "DriftScan",
                "az": 180.0,
                "el": 45.0
            }
        }]
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
        "scan_type": "target:a"
    },
    "csp": {
        "interface": "https://schema.skao.int/ska-csp-configure/2.0",
        "subarray": {
            "subarray_name": "science period 23"
        },
        "common": {
            "config_id": "sbi-mvp01-20200325-00001-science_A"
        },
        "lowcbf": {
            "stations": {
                "stns": [
                    [1, 0],
                    [2, 0],
                    [3, 0],
                    [4, 0]
                ],
                "stn_beams": [{
                    "beam_id": 1,
                    "freq_ids": [64, 65, 66, 67, 68, 69, 70, 71],
                    "boresight_dly_poly": "url"
                }]
            },
            "timing_beams": {
                "beams": [{
                    "pst_beam_id": 13,
                    "stn_beam_id": 1,
                    "offset_dly_poly": "url",
                    "stn_weights": [0.9, 1.0, 1.0, 0.9],
                    "jones": "url",
                    "dest_ip": ["10.22.0.1:2345", "10.22.0.3:3456"],
```

(continues on next page)

(continued from previous page)

```
                    "dest_chans": [128, 256],
                    "rfi_enable": [true, true, true],
                    "rfi_static_chans": [1, 206, 997],
                    "rfi_dynamic_chans": [242, 1342],
                    "rfi_weighted": 0.87
                }]
            }
        }
    },
    "tmc": {
        "scan_duration": 10.0
    }
}
```

https://schema.skao.int/ska-low-tmc-configure/3.0					
type	*object*				
properties					
• **interface**	URI of JSON schema applicable to this JSON payload.				
	type	*string*			
• transaction_id	A transaction id specific to the command				
	type	*string*			
	default	null			
• **mccs**	MCCS configuration specification.				
	type	*object*			
	properties				
	• **stations**	IDs of the MCCS stations to configure. Maximum array size = 512, the maximum number of MCCS stations.			
		type	*array*		
		items	type	*object*	
			properties		
			• **station_id**	MCCS Station ID. Each ID must be between 1 and 512.	
				type	*integer*
			additionalProperties	True	
	• **subarray_beams**	MCCS sub-array beam configuration.			
		type	*array*		
		items	type	*object*	
			properties		
			• **subarray_beam_id**	ID of MCCS sub-array beam to configure. ID must be an integer between 1 and 48.	
				type	*integer*
			• **station_ids**	IDs of MCCS stations within this sub-array beamto configure. Array size must be less than 512, the maximum number of MCCS stations. Each item in the list must be an integer between 1 and 512.	
				type	*array*

continues on next page

Table 36 – continued from previous page

				items	type	*integer*	
			• **update_rate**	Update rate for pointing information. Value must be 0.0 or greater. TODO: clarify whether this is specified as a frequency or as a cadence, plus units.			
				type	*number*		
			• **channels**	Channel block configurations. Each item in the list is a channel block configuration, each specified as a list of 4 numbers as follows: [start channel, number of channels, beam index, sub-station index] Constraints are: 0 < start channel < 376 start channel must be a multiple of 8 8 < number of channels < 48 1 < beam index < 48 1 < sub-station index < 8			
				type	*array*		
				items	type	*array*	
					items	type	*integer*
			• **antenna_weights**	Antenna weights. Maximum array size = 512 (=256 antennas x2 pols per sub-array beam). Antennas signals can be weighted to modify the station beam, varying from 0.0 for full exclusion to potentially 256.0 for an antenna contribution compensated for the number of antennas in the beam. This value is an amplitude multiplier added to that antenna signal before adding into the sum. Weights apply to all channels assigned to a beam.			
				type	*array*		
				items	type	*number*	
			• **phase_centre**	Phase centre offset for the station beam, in metres. The reference position for station phase must be modified to reflect antenna weighting and their contribution to the station beam. This offset can be can considered the desired centre of mass for the station. Constraints: array size = 2 -20 < phase centre value < 20			
				type	*array*		
				items	type	*number*	
			• **target**	Target position for the sub-array beam. Only drift scan targets are currently implemented by MCCS, hence only azimuth and elevation are specified.			
				type	*object*		
				properties			
				• **reference_frame**	Co-ordinate system. Must be HORIZON for drift scan.		
					type	*string*	
				• **target_name**	Name of target.		
					type	*string*	
				• **az**	Pointing azimuth in degrees.		

continues on next page

Table 36 – continued from previous page

					type	*number*	
				• **el**	Pointing elevation in degrees.		
					type	*number*	
				additionalProperties	False		
			additionalProperties	False			
	additionalProperties	False					
• **csp**	CSP configuration specification.						
	type	*object*					
	properties						
	• **interface**	CSP Interface configuration					
		type	*string*				
	• **subarray**	Subarray description					
		type	*object*				
		properties					
		• **subarray_name**	type	*string*			
		additionalProperties	True				
	• **common**	CSP Common configuration specification					
		type	*object*				
		properties					
		• **config_id**	CSP config id configuration				
			type	*string*			
		additionalProperties	False				
	• **lowcbf**	LOWCBF configuration specification.					
		type	*object*				
		properties					
		• **stations**	Subarray Stations and station beam input descriptions				
			type	*object*			
			properties				
			• **stns**	type	*array*		
				items	type	*array*	
					items	type	*integer*
			• **stn_beams**	type	*array*		
				items	type	*object*	
					properties		
					• **beam_id**	station beam id	

continues on next page

Table 36 – continued from previous page

						type	*integer*	
					• **freq_ids**	list of station beam frequency ids		
						type	*array*	
						items	type	*integer*
					• **bore-sight_dly_poly**	URL		
						type	*string*	
					additionalProperties	False		
			additionalProperties	False				
		• timing_beams	PST beam outputs descriptions					
			type	*object*				
			default	null				
			properties					
			• **beams**	inner				
				type	*array*			
				items	type	*object*		
					properties			
					• **stn_beam_id**	Station beam ID for pst beamforming		
						type	*integer*	
					• **pst_beam_id**	PST beam ID		
						type	*integer*	
					• firmware	Firmware name		
						type	*string*	
						default	null	
					• **offset_dly_poly**	Delay polynomial source URI		
						type	*string*	
					• **dest_ip**	Beam destination [ip_addr:port]		
						type	*array*	
						items	type	*string*
					• **dest_chans**	Number of fine chans to a destination		
						type	*array*	
						items	type	*integer*
					• **jones**	Jones matrix source URI		
						type	*string*	
					• **stn_weights**	weights for each station		
						type	*array*	
						items	type	*number*
					• rfi_enable	Master enable for RFI flagging		
						type	*array*	
						default	null	
						items	type	*boolean*
					• rfi_static_chans	Freqency IDs to be always flagged		
						type	*array*	
						default	null	
						items	type	*integer*

continues on next page

Table 36 – continued from previous page

					• rfi_dynamic_chans	Frequency IDs to be dynamically flagged		
						type	*array*	
						default	null	
						items	type	*integer*
					• rfi_weighted	Parameter for dynamic flagging		
						type	*number*	
						default	null	
					additionalProperties	False		
			additionalProperties	False				
		• search_beams	PSS beam outputs descriptions					
			type	*string*				
			default	null				
		• visibilities	Visibility output descriptions					
			type	*string*				
			default	null				
		• zooms	Zoom visibility output descriptions					
			type	*string*				
			default	null				
		additionalProperties	False					
	additionalProperties	False						
• **sdp**	SDP configuration specification.							
	type	*object*						
	properties							
	• interface	type	*string*					
		default	null					
	• transaction_id	type	*string*					
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$					
		default	null					
	• **scan_type**	type	*string*					
	• new_scan_types	type	*array*					
		default	null					
		items	type	*object*				
			properties					

continues on next page

Table 36 – continued from previous page

			• **scan_type_id**	const	(any scan type)
			• derive_from	type	*string*
			• beams	type	*object*
			additionalProperties	False	
	additionalProperties	False			
• tmc	TMC configuration specification.				
	type	*object*			
	default	null			
	properties				
	• **scan_duration**	Scan duration in seconds. Value must be >= 0.0			
		type	*number*		
	additionalProperties	True			
additionalProperties	False				

Low TMC configure 2.0

Example JSON.

```
{
    "interface": "https://schema.skao.in/ska-low-tmc-configure/2.0",
    "transaction_id": "txn-....-00001",
    "mccs": {
        "stations": [{
            "station_id": 1
        }, {
            "station_id": 2
        }],
        "subarray_beams": [{
            "subarray_beam_id": 1,
            "station_ids": [1, 2],
            "update_rate": 0.0,
            "channels": [
                [0, 8, 1, 1],
```

(continues on next page)

(continued from previous page)

```
                [8, 8, 2, 1],
                [24, 16, 2, 1]
            ],
            "antenna_weights": [1.0, 1.0, 1.0],
            "phase_centre": [0.0, 0.0],
            "target": {
                "reference_frame": "HORIZON",
                "target_name": "DriftScan",
                "az": 180.0,
                "el": 45.0
            }
        }]
    },
    "tmc": {
        "scan_duration": 10.0
    }
}
```

https://schema.skao.int/ska-low-tmc-configure/2.0					
type	*object*				
properties					
• **interface**	URI of JSON schema applicable to this JSON payload.				
	type	*string*			
• transaction_id	A transaction id specific to the command				
	type	*string*			
	default	null			
• **mccs**	MCCS configuration specification.				
	type	*object*			
	properties				
	• **stations**	IDs of the MCCS stations to configure. Maximum array size = 512, the maximum number of MCCS stations.			
		type	*array*		
		items	type	*object*	
			properties		
			• **station_id**	MCCS Station ID. Each ID must be between 1 and 512.	
				type	*integer*
			additionalProperties	True	
	• **subarray_beams**	MCCS sub-array beam configuration.			
		type	*array*		
		items	type	*object*	
			properties		
			• **subarray_beam_id**	ID of MCCS sub-array beam to configure. ID must be an integer between 1 and 48.	
				type	*integer*

continues on next page

Table 37 – continued from previous page

			• **station_ids**	IDs of MCCS stations within this sub-array beamto configure. Array size must be less than 512, the maximum number of MCCS stations. Each item in the list must be an integer between 1 and 512.			
				type	*array*		
				items	type	*integer*	
			• **update_rate**	Update rate for pointing information. Value must be 0.0 or greater. TODO: clarify whether this is specified as a frequency or as a cadence, plus units.			
				type	*number*		
			• **channels**	Channel block configurations. Each item in the list is a channel block configuration, each specified as a list of 4 numbers as follows: [start channel, number of channels, beam index, substation index] Constraints are: 0 < start channel < 376 start channel must be a multiple of 8 8 < number of channels < 48 1 < beam index < 48 1 < sub-station index < 8			
				type	*array*		
				items	type	*array*	
					items	type	*integer*
			• **antenna_weights**	Antenna weights. Maximum array size = 512 (=256 antennas x2 pols per sub-array beam). Antennas signals can be weighted to modify the station beam, varying from 0.0 for full exclusion to potentially 256.0 for an antenna contribution compensated for the number of antennas in the beam. This value is an amplitude multiplier added to that antenna signal before adding into the sum. Weights apply to all channels assigned to a beam.			
				type	*array*		
				items	type	*number*	
			• **phase_centre**	Phase centre offset for the station beam, in metres. The reference position for station phase must be modified to reflect antenna weighting and their contribution to the station beam. This offset can be can considered the desired centre of mass for the station. Constraints: array size = 2 -20 < phase centre value < 20			
				type	*array*		
				items	type	*number*	
			• **target**	Target position for the sub-array beam. Only drift scan targets are currently implemented by MCCS, hence only azimuth and elevation are specified.			
				type	*object*		
				properties			

continues on next page

Table 37 – continued from previous page

				• **reference_frame**	Co-ordinate system. Must be HORIZON for drift scan.	
					type	*string*
				• **target_name**	Name of target.	
					type	*string*
				• **az**	Pointing azimuth in degrees.	
					type	*number*
				• **el**	Pointing elevation in degrees.	
					type	*number*
				additionalProperties	False	
			additionalProperties	False		
	additionalProperties	False				
• tmc	TMC configuration specification.					
	type	*object*				
	default	null				
	properties					
	• **scan_duration**	Scan duration in seconds. Value must be >= 0.0				
		type	*number*			
	additionalProperties	True				
additionalProperties	False					

Low TMC configure 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-tmc-configure/1.0",
    "mccs": {
        "stations": [{
            "station_id": 1
        }, {
            "station_id": 2
        }],
        "subarray_beams": [{
            "subarray_beam_id": 1,
            "station_ids": [1, 2],
            "update_rate": 0.0,
            "channels": [
                [0, 8, 1, 1],
                [8, 8, 2, 1],
                [24, 16, 2, 1]
            ],
            "antenna_weights": [1.0, 1.0, 1.0],
            "phase_centre": [0.0, 0.0],
            "target": {
```

(continues on next page)

(continued from previous page)

```
                "system": "HORIZON",
                "name": "DriftScan",
                "az": 180.0,
                "el": 45.0
            }
        }]
    },
    "tmc": {
        "scan_duration": 10.0
    }
}
```

<table>
<tr><td colspan="2">https://schema.skatelescope.org/ska-low-tmc-configure/1.0</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• interface</td><td>URI of JSON schema applicable to this JSON payload.
<table><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>• transaction_id</td><td>A transaction id specific to the command
<table><tr><td>type</td><td>string</td></tr><tr><td>default</td><td>null</td></tr></table></td></tr>
<tr><td>• mccs</td><td>MCCS configuration specification.
<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• stations</td><td>IDs of the MCCS stations to configure.
Maximum array size = 512, the maximum number of MCCS stations.
<table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td>
<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• station_id</td><td>MCCS Station ID.
Each ID must be between 1 and 512.
<table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>additionalProperties</td><td>True</td></tr>
</table></td></tr>
</table></td></tr>
<tr><td>• subarray_beams</td><td>MCCS sub-array beam configuration.
<table>
<tr><td>type</td><td>array</td></tr>
<tr><td>items</td><td>
<table>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• subarray_beam_id</td><td>ID of MCCS sub-array beam to configure.
ID must be an integer between 1 and 48.
<table><tr><td>type</td><td>integer</td></tr></table></td></tr>
<tr><td>• station_ids</td><td>IDs of MCCS stations within this sub-array beamto configure.
Array size must be less than 512, the maximum number of MCCS stations.
Each item in the list must be an integer between 1 and 512.
<table><tr><td>type</td><td>array</td></tr><tr><td>items</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr></table></td></tr>
</table></td></tr>
</table></td></tr>
</table></td></tr>
</table>

continues on next page

Table 38 – continued from previous page

			• **update_rate**	Update rate for pointing information. Value must be 0.0 or greater. TODO: clarify whether this is specified as a frequency or as a cadence, plus units.			
				type	*number*		
			• **channels**	Channel block configurations. Each item in the list is a channel block configuration, each specified as a list of 4 numbers as follows: [start channel, number of channels, beam index, substation index] Constraints are: 0 < start channel < 376 start channel must be a multiple of 8 8 < number of channels < 48 1 < beam index < 48 1 < sub-station index < 8			
				type	*array*		
				items	type	*array*	
					items	type	*integer*
			• **antenna_weights**	Antenna weights. Maximum array size = 512 (=256 antennas x2 pols per sub-array beam). Antennas signals can be weighted to modify the station beam, varying from 0.0 for full exclusion to potentially 256.0 for an antenna contribution compensated for the number of antennas in the beam. This value is an amplitude multiplier added to that antenna signal before adding into the sum. Weights apply to all channels assigned to a beam.			
				type	*array*		
				items	type	*number*	
			• **phase_centre**	Phase centre offset for the station beam, in metres. The reference position for station phase must be modified to reflect antenna weighting and their contribution to the station beam. This offset can be can considered the desired centre of mass for the station. Constraints: array size = 2 -20 < phase centre value < 20			
				type	*array*		
				items	type	*number*	
			• **target**	Target position for the sub-array beam. Only drift scan targets are currently implemented by MCCS, hence only azimuth and elevation are specified.			
				type	*object*		
				properties			
				• **system**	Co-ordinate system. Must be HORIZON for drift scan.		
					type	*string*	
				• **name**	Name of target.		
					type	*string*	
				• **az**	Pointing azimuth in degrees.		
					type	*number*	
				• **el**	Pointing elevation in degrees.		

continues on next page

Table 38 – continued from previous page

					type	*number*
				additional-Properties	False	
			additional-Properties	False		
	additional-Properties	False				
• tmc	TMC configuration specification.					
	type	*object*				
	default	null				
	properties					
	• **scan_duration**	Scan duration in seconds. Value must be >= 0.0				
		type	*number*			
	additional-Properties	True				
additional-Properties	False					

1.15.3 ska-low-tmc-releaseresources

Low TMC resource release 3.0

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-low-tmc-releaseresources/3.0",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "release_all": true
}
```

https://schema.skao.int/ska-low-tmc-releaseresources/3.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **subarray_id**	ID of the sub-array which should release resources.	
	type	*integer*
• **release_all**	true to release all resources, false to release only the resources defined in this payload. Note: partial resource release for SKA LOW is not implemented and the identification of the resources to release is not yet part of the schema.	
	type	*boolean*
additionalProperties	False	

Low TMC resource release 2.0

Example JSON.


```
{
    "interface": "https://schema.skao.in/ska-low-tmc-releaseresources/2.0",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "release_all": true
}
```


https://schema.skao.int/ska-low-tmc-releaseresources/2.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **subarray_id**	ID of the sub-array which should release resources.	
	type	*integer*
• **release_all**	true to release all resources, false to release only the resources defined in this payload. Note: partial resource release for SKA LOW is not implemented and the identification of the resources to release is not yet part of the schema.	
	type	*boolean*
additionalProperties	False	

Low TMC resource release 1.0

Example JSON.


```
{
    "interface": "https://schema.skatelescope.org/ska-low-tmc-releaseresources/1.0",
    "subarray_id": 1,
    "release_all": true
}
```


https://schema.skatelescope.org/ska-low-tmc-releaseresources/1.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• **subarray_id**	ID of the sub-array which should release resources.	
	type	*integer*
• **release_all**	true to release all resources, false to release only the resources defined in this payload. Note: partial resource release for SKA LOW is not implemented and the identification of the resources to release is not yet part of the schema.	
	type	*boolean*
additionalProperties	False	

1.15.4 ska-low-tmc-scan

Low TMC scan 4.0

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-low-tmc-scan/4.0",
    "transaction_id": "txn-....-00001",
    "scan_id": 1,
    "subarray_id": 1
}
```


https://schema.skao.int/ska-low-tmc-scan/4.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **scan_id**	Scan ID to associate with the data. The scan ID and SBI ID are used together to uniquely associate the data taken with the telescope configuration in effect at the moment of observation.	
	type	*integer*
• **subarray_id**	ID of the sub-array which should release resources.	
	type	*integer*
additionalProperties	False	

Low TMC scan 3.0

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-low-tmc-scan/3.0",
    "transaction_id": "txn-....-00001",
    "scan_id": 1
}
```

https://schema.skao.int/ska-low-tmc-scan/3.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **scan_id**	Scan ID to associate with the data. The scan ID and SBI ID are used together to uniquely associate the data taken with the telescope configuration in effect at the moment of observation.	
	type	*integer*
additionalProperties	False	

Low TMC scan 2.0

Example JSON.


```
{
    "interface": "https://schema.skao.in/ska-low-tmc-scan/2.0",
    "transaction_id": "txn-....-00001",
    "scan_id": 1
}
```


https://schema.skao.int/ska-low-tmc-scan/2.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **scan_id**	Scan ID to associate with the data. The scan ID and SBI ID are used together to uniquely associate the data taken with the telescope configuration in effect at the moment of observation.	
	type	*integer*
additionalProperties	False	

Low TMC scan 1.0

Example JSON.


```
{
    "interface": "https://schema.skatelescope.org/ska-low-tmc-scan/1.0",
    "scan_id": 1
}
```

https://schema.skatelescope.org/ska-low-tmc-scan/1.0		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **scan_id**	Scan ID to associate with the data. The scan ID and SBI ID are used together to uniquely associate the data taken with the telescope configuration in effect at the moment of observation.	
	type	*integer*
additionalProperties	False	

1.15.5 ska-low-tmc-assignedresources

Low TMC assigned resources 1.0

Example JSON.

```
{
    "interface": "https://schema.skatelescope.org/ska-low-tmc-assignedresources/1.0",
    "mccs": {
        "subarray_beam_ids": [1],
        "station_ids": [
            [1, 2]
        ],
        "channel_blocks": [3]
    }
}
```

<table>
<tr><td colspan="2">https://schema.skatelescope.org/ska-low-tmc-assignedresources/1.0</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• interface</td><td><table><tr><td colspan="2">URI of JSON schema applicable to this JSON payload.</td></tr><tr><td>type</td><td>string</td></tr></table></td></tr>
<tr><td>• mccs</td><td><table>
<tr><td colspan="2">Specification of the MCCS resources allocated to this sub-array.</td></tr>
<tr><td>type</td><td>object</td></tr>
<tr><td colspan="2">properties</td></tr>
<tr><td>• subarray_beam_ids</td><td><table><tr><td colspan="2">IDs of the MCCS sub-array beams allocated to this subarray.
Each ID must be between 1 and 48, the maximum number of sub-array beams.</td></tr><tr><td>type</td><td>array</td></tr><tr><td>items</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr></table></td></tr>
<tr><td>• station_ids</td><td><table><tr><td colspan="2">IDs of MCCS stations allocated to each MCCS sub-array beam.
Each ID must be between 1 and 512, the maximum number of stations.</td></tr><tr><td>type</td><td>array</td></tr><tr><td>items</td><td><table><tr><td>type</td><td>array</td></tr><tr><td>items</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr></table></td></tr></table></td></tr>
<tr><td>• channel_blocks</td><td><table><tr><td colspan="2">Number of channel blocks allocated per sub-array beam.
Maximum number of channel blocks = 48.</td></tr><tr><td>type</td><td>array</td></tr><tr><td>items</td><td><table><tr><td>type</td><td>integer</td></tr></table></td></tr></table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table></td></tr>
<tr><td>additionalProperties</td><td>False</td></tr>
</table>

1.15.6 ska-tmc-assignresources

Mid TMC assign resources 2.1

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-tmc-assignresources/2.1",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "dish": {
        "receptor_ids": ["0001"]
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-assignres/0.4",
        "execution_block": {
            "eb_id": "eb-mvp01-20210623-00000",
            "max_length": 100.0,
            "context": {},
            "beams": [{
                "beam_id": "vis0",
                "function": "visibilities"
            }, {
                "beam_id": "pss1",
```


(continues on next page)

(continued from previous page)


```
            "search_beam_id": 1,
            "function": "pulsar search"
        }, {
            "beam_id": "pss2",
            "search_beam_id": 2,
            "function": "pulsar search"
        }, {
            "beam_id": "pst1",
            "timing_beam_id": 1,
            "function": "pulsar timing"
        }, {
            "beam_id": "pst2",
            "timing_beam_id": 2,
            "function": "pulsar timing"
        }, {
            "beam_id": "vlbi1",
            "vlbi_beam_id": 1,
            "function": "vlbi"
        }],
        "scan_types": [{
            "scan_type_id": ".default",
            "beams": {
                "vis0": {
                    "channels_id": "vis_channels",
                    "polarisations_id": "all"
                },
                "pss1": {
                    "field_id": "pss_field_0",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "pss2": {
                    "field_id": "pss_field_1",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "pst1": {
                    "field_id": "pst_field_0",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "pst2": {
                    "field_id": "pst_field_1",
                    "channels_id": "pulsar_channels",
                    "polarisations_id": "all"
                },
                "vlbi": {
                    "field_id": "vlbi_field",
                    "channels_id": "vlbi_channels",
                    "polarisations_id": "all"
                }
            }
```


(continues on next page)

(continued from previous page)

```
            }, {
                "scan_type_id": "target:a",
                "derive_from": ".default",
                "beams": {
                    "vis0": {
                        "field_id": "field_a"
                    }
                }
            }],
            "channels": [{
                "channels_id": "vis_channels",
                "spectral_windows": [{
                    "spectral_window_id": "fsp_1_channels",
                    "count": 744,
                    "start": 0,
                    "stride": 2,
                    "freq_min": 350000000.0,
                    "freq_max": 368000000.0,
                    "link_map": [
                        [0, 0],
                        [200, 1],
                        [744, 2],
                        [944, 3]
                    ]
                }, {
                    "spectral_window_id": "fsp_2_channels",
                    "count": 744,
                    "start": 2000,
                    "stride": 1,
                    "freq_min": 360000000.0,
                    "freq_max": 368000000.0,
                    "link_map": [
                        [2000, 4],
                        [2200, 5]
                    ]
                }, {
                    "spectral_window_id": "zoom_window_1",
                    "count": 744,
                    "start": 4000,
                    "stride": 1,
                    "freq_min": 360000000.0,
                    "freq_max": 361000000.0,
                    "link_map": [
                        [4000, 6],
                        [4200, 7]
                    ]
                }]
            }, {
                "channels_id": "pulsar_channels",
                "spectral_windows": [{
                    "spectral_window_id": "pulsar_fsp_channels",
                    "count": 744,
```

(continues on next page)

(continued from previous page)

```
                    "start": 0,
                    "freq_min": 350000000.0,
                    "freq_max": 368000000.0
                }]
            }],
            "polarisations": [{
                "polarisations_id": "all",
                "corr_type": ["XX", "XY", "YY", "YX"]
            }],
            "fields": [{
                "field_id": "field_a",
                "phase_dir": {
                    "ra": [123, 0.1],
                    "dec": [80, 0.1],
                    "reference_time": "...",
                    "reference_frame": "ICRF3"
                },
                "pointing_fqdn": "low-tmc/telstate/0/pointing"
            }]
        },
        "processing_blocks": [{
            "pb_id": "pb-mvp01-20210623-00000",
            "sbi_ids": ["sbi-mvp01-20200325-00001"],
            "script": {
                "kind": "realtime",
                "name": "vis_receive",
                "version": "0.1.0"
            },
            "parameters": {}
        }, {
            "pb_id": "pb-mvp01-20210623-00001",
            "sbi_ids": ["sbi-mvp01-20200325-00001"],
            "script": {
                "kind": "realtime",
                "name": "test_realtime",
                "version": "0.1.0"
            },
            "parameters": {}
        }, {
            "pb_id": "pb-mvp01-20210623-00002",
            "sbi_ids": ["sbi-mvp01-20200325-00002"],
            "script": {
                "kind": "batch",
                "name": "ical",
                "version": "0.1.0"
            },
            "parameters": {},
            "dependencies": [{
                "pb_id": "pb-mvp01-20210623-00000",
                "kind": ["visibilities"]
            }]
        }, {
```

(continues on next page)

(continued from previous page)


```
            "pb_id": "pb-mvp01-20210623-00003",
            "sbi_ids": ["sbi-mvp01-20200325-00001", "sbi-mvp01-20200325-00002"],
            "script": {
                "kind": "batch",
                "name": "dpreb",
                "version": "0.1.0"
            },
            "parameters": {},
            "dependencies": [{
                "pb_id": "pb-mvp01-20210623-00002",
                "kind": ["calibration"]
            }]
        }],
        "resources": {
            "csp_links": [1, 2, 3, 4],
            "receptors": ["FS4", "FS8", "FS16", "FS17", "FS22", "FS23", "FS30", "FS31", "FS32", "FS33", "FS36", "FS52", "FS56", "FS57", "FS59", "FS62", "FS66", "FS69", "FS70", "FS72", "FS73", "FS78", "FS80", "FS88", "FS89", "FS90", "FS91", "FS98", "FS108", "FS111", "FS132", "FS144", "FS146", "FS158", "FS165", "FS167", "FS176", "FS183", "FS193", "FS200", "FS345", "FS346", "FS347", "FS348", "FS349", "FS350", "FS351", "FS352", "FS353", "FS354", "FS355", "FS356", "FS429", "FS430", "FS431", "FS432", "FS433", "FS434", "FS465", "FS466", "FS467", "FS468", "FS469", "FS470"],
            "receive_nodes": 10
        }
    }
}
```


https://schema.skao.int/ska-tmc-assignresources/2.1				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• transaction_id	A transaction id specific to the command			
	type	*string*		
	default	null		
• **subarray_id**	ID of sub-array targeted by this resource allocation request			
	type	*integer*		
• **dish**	Mid Telescope specification for Dish allocation.			
	type	*object*		
	properties			
	• **receptor_ids**	Receptor ids of dishes		
		type	*array*	
		items	type	*string*
	additionalProperties	False		
• **sdp**	sdp block for assignres version 0.4			
	type	*object*		
	properties			
	• interface	type	*string*	
		default	null	
	• transaction_id	type	*string*	

continues on next page

Table 39 – continued from previous page

		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$			
		default	null			
	• execution_block	Execution block				
		default	null			
		Execution block 0.4				
	• resources	External resources				
		type	*object*			
		default	null			
		properties				
		• receptors	type	*array*		
			default	null		
			items	anyOf	type	*string*
					pattern	^C([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-1][0-9]|22[0-4])$
					type	*string*
					pattern	^[ENS]([1-9]|1[0-6])-[1-6]$
					type	*string*
					pattern	^FS([1-9]|[1-9][0-9]|[1-4][0-9][0-9]|50[0-9]|51[0-2])(\.\S+)?$
					type	*string*
					pattern	^SKA((?!000)0[0-9][0-9]|1[0-2][0-9]|13[0-3])$
					type	*string*
					pattern	^MKT0([0-5][0-9]|6[0-3])$
		additionalProperties	True			
	• processing_blocks	Processing blocks				
		type	*array*			
		default	null			

continues on next page

Table 39 – continued from previous page

		items	A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress. PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.
			Processing block 0.4
	additional-Properties	False	
additional-Properties	False		

Execution block 0.4

type	*object*		
properties			
• **eb_id**	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
• **max_length**	type	*number*	
• **context**	Free-form information from OET, see ADR-54		
• **beams**	Beam parameters		
	type	*array*	
	items	Beam parameters for the purpose of the Science Data Processor.	
		Beam 0.4	
• **scan_types**	Scan types. Associates scans with per-beam fields & channel configurations		
	type	*array*	
	items	type	*object*
		properties	
		• **scan_type_id**	type: *string*
		• de-rive_from	type: *string*
		• beams	type: *object*

continues on next page

Table 40 – continued from previous page

		additionalProperties	False		
• **channels**	Channels				
	type	*array*			
	items	Spectral windows per channel configuration.			
		Scan channels 0.4			
• **polarisations**	Polarisation definitions				
	type	*array*			
	items	Polarisation definition.			
		type	*object*		
		properties			
		• **polarisations_id**	type	*string*	
		• **corr_type**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• **fields**	Fields / targets				
	type	*array*			
	items	Fields / Targets			
		type	*object*		
		properties			
		• **field_id**	type	*string*	
		• **phase_dir**	Phase direction		
			type	*object*	
			properties		
			• **ra**	type	*array*
				items	
			• **dec**	type	*array*
				items	
			• **reference_time**	type	*string*
			• **reference_frame**	const	ICRF3
			additionalProperties	False	
		• pointing_fqdn	type	*string*	
		additionalProperties	False		
additionalProperties	False				

Beam 0.4

Beam parameters for the purpose of the Science Data Processor.

type	*object*	
properties		
• **beam_id**	Name to identify the beam within the SDP configuration.	
	type	*string*
• **function**	Identifies the type and origin of the generated beam data. This corresponds to a certain kind of calibration or receive functionality SDP is meant to provide for it. Possible options: • *visibilities*: Correlated voltages from CBF used for calibration and imaging • *pulsar search*: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar search data products • *pulsar timing*: SDP provides calibrations for tied-array beam as well as post-processes and delivers pulsar timing data products • *vlbi*: SDP provides calibrations for tied-array beam • *transient buffer*: SDP receives and delivers transient buffer data dumps	
	enum	visibilities, pulsar search, pulsar timing, vlbi, transient buffer
• search_beam_id	type	*integer*
	default	null
• timing_beam_id	type	*integer*
	default	null
• vlbi_beam_id	type	*integer*
	default	null
additionalProperties	False	

Scan channels 0.4

Spectral windows per channel configuration.

type	*object*			
properties				
• **channels_id**				
• **spectral_windows**	type	*array*		
	items	type	*object*	
		properties		
		• **spectral_window_id**		
		• **count**	Number of channels	
			type	*integer*
		• **start**	First channel ID	
			type	*integer*
		• stride	Distance between subsequent channel IDs	
			type	*integer*
			default	null
		• **freq_min**	Lower bound of first channel	
			type	*number*
		• **freq_max**	Upper bound of last channel	
			type	*number*
		• link_map	Channel map that specifies which network link is going to get used to send channels to SDP. Intended to allow SDP to optimise network and receive node configuration.	
			type	*array*
			default	null
			items	
		additionalProperties	False	
additionalProperties	False			

Processing block 0.4

A Processing Block is an atomic unit of data processing for the purpose of SDP's internal scheduler. Each PB references a processing script and together with the associated execution block provides all parameters necessary to carry out scheduling - both on TM's side for observation planning and on SDP's side - as well as enable processing to locate all required inputs once it is in progress.

PBs are used for both real-time and deferred, batch, processing. An execution block will often contain many Processing Blocks, for example for ingest, self-calibration and Data Product preparation.

type	*object*			
properties				
• **pb_id**	Unique identifier for this processing block.			
	type	*string*		
	pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
• **script**	Specification of the workflow to be executed along with configuration parameters for the workflow.			
	type	*object*		
	properties			
	• **kind**	The kind of processing script (realtime or batch)		
		allOf	type	*string*
			enum	realtime, batch
	• **name**	The name of the processing script		

continues on next page

Table 41 – continued from previous page

		type	*string*		
	• **version**	Version of the processing script. Uses semantic versioning.			
		type	*string*		
	additionalProperties	False			
• parameters	Configuration parameters needed to execute the workflow. As these parameters will be workflow specific, this is left as an object to be specified by the workflow definition.				
	type	*object*			
	default	null			
• dependencies	A dependency between processing blocks means that one processing block requires something from the other processing block to run - typically an intermediate Data Product. This generally means that 1. The dependent processing block might only be able to start once the dependency has been fulfilled 2. Data associated with the dependency must be kept alive until the dependent processing block is finished. As processing blocks might have many different outputs, the dependency “kind” can be used to specify how this dependency is meant to be interpreted (e.g. “visibilities”, “calibration”…)				
	type	*array*			
	default	null			
	items	type	*object*		
		properties			
		• **pb_id**	type	*string*	
			pattern	^pb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
		• **kind**	type	*array*	
			items	type	*string*
		additionalProperties	False		
• sbi_ids	Scheduling block instances that the processing block belongs to.				
	type	*array*			
	default	null			
	items	type	*string*		
		pattern	^sbi\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
additionalProperties	False				

1.15.7 ska-tmc-configure

Mid TMC configure 2.2

Example JSON.

```
{
    "interface": "https://schema.skao.int/ska-tmc-configure/2.2",
    "transaction_id": "txn-....-00001",
    "pointing": {
        "target": {
            "reference_frame": "ICRS",
            "target_name": "Polaris Australis",
            "ra": "21:08:47.92",
```

(continues on next page)

(continued from previous page)


```
            "dec": "-88:57:22.9",
            "ca_offset_arcsec": 0.0,
            "ie_offset_arcsec": 0.0
        }
    },
    "dish": {
        "receiver_band": "1"
    },
    "csp": {
        "interface": "https://schema.skao.int/ska-csp-configure/2.0",
        "subarray": {
            "subarray_name": "science period 23"
        },
        "common": {
            "config_id": "sbi-mvp01-20200325-00001-science_A",
            "frequency_band": "1",
            "subarray_id": 1
        },
        "cbf": {
            "fsp": [{
                "fsp_id": 1,
                "function_mode": "CORR",
                "frequency_slice_id": 1,
                "integration_factor": 1,
                "zoom_factor": 0,
                "channel_averaging_map": [
                    [0, 2],
                    [744, 0]
                ],
                "channel_offset": 0,
                "output_link_map": [
                    [0, 0],
                    [200, 1]
                ]
            }, {
                "fsp_id": 2,
                "function_mode": "CORR",
                "frequency_slice_id": 2,
                "integration_factor": 1,
                "zoom_factor": 1,
                "channel_averaging_map": [
                    [0, 2],
                    [744, 0]
                ],
                "channel_offset": 744,
                "output_link_map": [
                    [0, 4],
                    [200, 5]
                ],
                "zoom_window_tuning": 650000
            }],
            "vlbi": {}
```


(continues on next page)

(continued from previous page)

```
        },
        "pss": {},
        "pst": {}
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
        "scan_type": "science_A"
    },
    "tmc": {
        "scan_duration": 10.0,
        "partial_configuration": false
    }
}
```

https://schema.skao.int/ska-tmc-configure/2.2				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• transaction_id	A transaction id specific to the command			
	type	*string*		
	default	null		
• **pointing**	Pointing configuration specification.			
	type	*object*		
	properties			
	• **target**	Target configuration coordinates		
		type	*object*	
		properties		
		• reference_frame	standard celestial reference system such as ICRS	
			type	*string*
			default	null
		• target_name	celestial source	
			type	*string*
			default	null
		• ra	Pointing Right Ascension coordinates.	
			type	*string*
			default	null
		• dec	Pointing Declination coordinates.	
			type	*string*
			default	null
		• ca_offset_arcsec	Cross-elevation offset in arcseconds from the central pointing pointing defined by target's ra+dec. This is an optional field; if omitted, an offset of 0 arcseconds can be assumed.	
			type	*number*
			default	null
		• ie_offset_arcsec	Elevation offset in arcseconds from the central pointing position defined by the ra+dec pair. This is an optional field; if omitted, an offset of 0 arcseconds can be assumed.	
			type	*number*

continues on next page

Table 42 – continued from previous page

			default	null
		additionalProperties	False	
	additionalProperties	False		
• dish	Dish band configuration			
	type	*object*		
	default	null		
	properties			
	• **receiver_band**	Dish Receiver band configuration		
		type	*string*	
	additionalProperties	True		
• csp	CSP configuration specification.			
	type	*object*		
	default	null		
	properties			
	• **interface**	type	*string*	
	• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
		type	*object*	
		properties		
		• **subarray_name**	Name and scope of current subarray the sub-array.	
			type	*string*
		additionalProperties	False	
	• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
		Common CSP config 2.0		
	• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
		CBF config 2.0		
	• pss	default	null	
		PSS configuration 2.0		
	• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
		default	null	
		PST configuration 2.0		
	additionalProperties	False		
• sdp	SDP configuration specification.			
	type	*object*		
	default	null		
	properties			
	• interface	type	*string*	
		default	null	
	• transaction_id	type	*string*	
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	

continues on next page

Table 42 – continued from previous page

		default	null		
	• **scan_type**	type	*string*		
	• new_scan_types	type	*array*		
		default	null		
		items	type	*object*	
			properties		
			• **scan_type_id**	const	(any scan type)
			• derive_from	type	*string*
			• beams	type	*object*
			additionalProperties	False	
	additionalProperties	False			
• tmc	TMC Mid TMC configuration specification.				
	type	*object*			
	default	null			
	properties				
	• scan_duration	Scan duration in seconds. Value must be >= 0.0			
		type	*number*		
		default	null		
	• partial_configuration	Partial Configuration Flag. Partial configurations assume that previously set state is maintained, and undergo less strict JSON validation.			
		type	*boolean*		
		default	null		
	additionalProperties	False			
additionalProperties	False				

Common CSP config 2.0

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

type	*object*		
properties			
• config_id	type	*string*	
	default	null	
• **frequency_band**	Frequency band applies for all the receptors (VCCs) that belong to the sub-array.		
	type	*string*	
	pattern	^(1|2|3|4|5(a|b))$	
• band_5_tuning	Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4). Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream. The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.		
	type	*array*	
	default	null	
	items	type	*number*
• eb_id	Execution block ID to associate scan configs to an observation. This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation. This ID does not have to be unique for a scan configuration but should be unique for different observations. For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.		
	type	*string*	
	pattern	^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$	
	default	null	
• **subarray_id**	Subarray number		
	type	*integer*	
additionalProperties	False		

CBF config 2.0

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*		
properties			
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)		
	type	*integer*	
	default	null	
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*		
	type	*integer*	
	default	null	
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.		
	type	*string*	
	default	null	
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.		
	type	*string*	
	default	null	
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).		
	type	*object*	
	default	null	
	properties		
	additionalProperties	True	
• **fsp**	type	*array*	
	items	*FSP config 2.0*	
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.		
	default	null	
	VLBI config 2.0		
• search_window	type	*array*	
	default	null	
	items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming.	
		Search window config 2.0	
additionalProperties	False		

FSP config 2.0

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3]))| (MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 43 – continued from previous page

• channel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• channel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• output_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• output_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 43 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.0

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.0

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.0

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

PST configuration 2.0

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

Mid TMC configure 2.1

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-tmc-configure/2.1",
    "transaction_id": "txn-....-00001",
    "pointing": {
        "target": {
            "reference_frame": "ICRS",
            "target_name": "Polaris Australis",
            "ra": "21:08:47.92",
            "dec": "-88:57:22.9"
        }
    },
    "dish": {
        "receiver_band": "1"
    },
    "csp": {
        "interface": "https://schema.skao.int/ska-csp-configure/2.0",
        "subarray": {
            "subarray_name": "science period 23"
        },
        "common": {
            "config_id": "sbi-mvp01-20200325-00001-science_A",
            "frequency_band": "1",
            "subarray_id": 1
        },
        "cbf": {
            "fsp": [{
                "fsp_id": 1,
                "function_mode": "CORR",
                "frequency_slice_id": 1,
                "integration_factor": 1,
                "zoom_factor": 0,
                "channel_averaging_map": [
                    [0, 2],
                    [744, 0]
                ],
                "channel_offset": 0,
                "output_link_map": [
```


(continues on next page)

(continued from previous page)

```
                    [0, 0],
                    [200, 1]
                ]
            }, {
                "fsp_id": 2,
                "function_mode": "CORR",
                "frequency_slice_id": 2,
                "integration_factor": 1,
                "zoom_factor": 1,
                "channel_averaging_map": [
                    [0, 2],
                    [744, 0]
                ],
                "channel_offset": 744,
                "output_link_map": [
                    [0, 4],
                    [200, 5]
                ],
                "zoom_window_tuning": 650000
            }],
            "vlbi": {}
        },
        "pss": {},
        "pst": {}
    },
    "sdp": {
        "interface": "https://schema.skao.int/ska-sdp-configure/0.4",
        "scan_type": "science_A"
    },
    "tmc": {
        "scan_duration": 10.0
    }
}
```

https://schema.skao.int/ska-tmc-configure/2.1				
type	*object*			
properties				
• **interface**	URI of JSON schema applicable to this JSON payload.			
	type	*string*		
• transaction_id	A transaction id specific to the command			
	type	*string*		
	default	null		
• **pointing**	Pointing configuration specification.			
	type	*object*		
	properties			
	• **target**	Target configuration coordinates		
		type	*object*	
		properties		
		• reference_frame	standard celestial reference system such as ICRS	
			type	*string*
			default	null

continues on next page

Table 44 – continued from previous page

		• target_name	celestial source	
			type	*string*
			default	null
		• ra	Pointing Right Ascension coordinates.	
			type	*string*
			default	null
		• dec	Pointing Declination coordinates.	
			type	*string*
			default	null
		additionalProperties	False	
	additionalProperties	False		
• dish	Dish band configuration			
	type	*object*		
	default	null		
	properties			
	• **receiver_band**	Dish Receiver band configuration		
		type	*string*	
	additionalProperties	True		
• csp	CSP configuration specification.			
	type	*object*		
	default	null		
	properties			
	• **interface**	type	*string*	
	• **subarray**	subarray section, containing the parameters relevant only for the current sub-array device. This section is not forwarded to any subelement.		
		type	*object*	
		properties		
		• **subarray_name**	Name and scope of current subarray the sub-array.	
			type	*string*
		additionalProperties	False	
	• **common**	Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.		
		Common CSP config 2.0		
	• **cbf**	Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD		
		CBF config 2.0		
	• pss	default	null	
		PSS configuration 2.0		
	• pst	Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.		
		default	null	
		PST configuration 2.0		
	additionalProperties	False		

continues on next page

Table 44 – continued from previous page

• sdp	SDP configuration specification.				
	type	*object*			
	default	null			
	properties				
	• interface	type	*string*		
		default	null		
	• transaction_id	type	*string*		
		pattern	^txn\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$		
		default	null		
	• **scan_type**	type	*string*		
	• new_scan_types	type	*array*		
		default	null		
		items	type	*object*	
			properties		
			• **scan_type_id**	const	(any scan type)
			• derive_from	type	*string*
			• beams	type	*object*
			additionalProperties	False	
	additionalProperties	False			
• tmc	TMC Mid TMC configuration specification.				
	type	*object*			
	default	null			
	properties				
	• scan_duration	Scan duration in seconds. Value must be >= 0.0			
		type	*number*		
		default	null		
	additionalProperties	False			
additionalProperties	False				

Common CSP config 2.0

Common section, containing the parameters and the sections belonging to all CSP sub elements. This section is forwarded to all sub-elements.

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td rowspan="2">• config_id</td><td>type</td><td colspan="2">string</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="3">• frequency_band</td><td colspan="3">Frequency band applies for all the receptors (VCCs) that belong to the sub-array.</td></tr>
<tr><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^(1|2|3|4|5(a|b))$</td></tr>
<tr><td rowspan="4">• band_5_tuning</td><td colspan="3">Center frequency for the Band-of-Interest. Required if Band is 5a or 5b; not specified for other Bands (not configurable for Band 1, 2, 3 and 4).
Input for Band 5a and 5b consists of two 2.5 GHz streams; the center frequency can be independently tuned for each stream.
The following nomenclature is used to refer to Band 5a and 5b streams: 5a1, 5a2, 5b1, 5b2.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td>items</td><td>type</td><td>number</td></tr>
<tr><td rowspan="4">• eb_id</td><td colspan="3">Execution block ID to associate scan configs to an observation.
This ID is used for associating generated data, especially data products, for a given observation. Multiple scans can be linked to one observation and this ID is used as metadata to associate the data products from all scans of the same observation.
This ID does not have to be unique for a scan configuration but should be unique for different observations.
For example, all the data and weights files will have an EB_ID header value populated with the value supplied in this field.</td></tr>
<tr><td>type</td><td colspan="2">string</td></tr>
<tr><td>pattern</td><td colspan="2">^eb\-[a-z0-9]+\-[0-9]{8}\-[a-z0-9]+$</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• subarray_id</td><td colspan="3">Subarray number</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>additionalProperties</td><td colspan="3">False</td></tr>
</table>

CBF config 2.0

Correlator and Beamformer specific parameters. This section contains the parameters relevant only for CBF sub-element. This section is forwarded only to CBF subelement. Most of it to be borrowed from IICD

type	*object*		
properties			
• frequency_band_offset_stream1	Optionally, an offset can be specified so that the entire observed band is shifted (to accommodate a Zoom Window that crosses a 'natural' Frequency Slice boundary). If specified, applies for all the receptors in the sub-array. Bands 1, 2, 3 and 4: input from the receptor consists of a single data stream; the Frequency Band Offset (FBO) should be specified for Stream 1 only. Bands 5a and 5b: input from the receptor consists of two data streams; the FBO can be specified for each stream independently. Note: For Band 5a and 5b the frequency shift is performed by the receptor (DISH). Note: This is optional and does not need to be implemented in PI3, but would be great for demo; if Team Buttons is looking for opportunities to showcase interesting GUIs, Zoom Windows are perfect opportunity (would require TMC and CSP to support these two parameters, corrBandwidth values > 0 and zoom window tuning.)		
		type	*integer*
		default	null
• frequency_band_offset_stream2	See *frequencyBandOffsetStream1*		
		type	*integer*
		default	null
• delay_model_subscription_point	FQDN of TMC.DelayModel TANGO attribute which exposes delay values for all the dishes assigned to a Subarray in JSON format. Delay values are updated every 10 seconds.		
		type	*string*
		default	null
• doppler_phase_corr_subscription_point	The same model applies for all receptors that belong to the subarray. Delivered by TMC using publish-subscribe mechanism (see ICD Section 3.8.8.5.3). The Doppler phase correction, by default, applies only to the CSP_Mid Processing Mode Correlation; optionally may apply to other Processing Modes as well.		
		type	*string*
		default	null
• rfi_flagging_mask	Specified as needed in advance of the scan start and/or during the scan. Delivered using publish-subscribe mechanism (see ICD Section 3.8.8.5.7).		
		type	*object*
		default	null
		properties	
		additionalProperties	True
• **fsp**		type	*array*
		items	*FSP config 2.0*
• vlbi	Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.		
		default	null
	VLBI config 2.0		
• search_window		type	*array*
		default	null
		items	Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beamforming. *Search window config 2.0*
additionalProperties	False		

FSP config 2.0

<table>
<tr><td>type</td><td colspan="3">object</td></tr>
<tr><td colspan="4">properties</td></tr>
<tr><td>• fsp_id</td><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• function_mode</td><td rowspan="2">allOf</td><td>type</td><td>string</td></tr>
<tr><td>enum</td><td>CORR, PSS-BF, PST-BF, VLBI</td></tr>
<tr><td rowspan="5">• receptors</td><td colspan="3">Optionally a subset of receptors to be correlated can be specified. If not specified, all receptors that belong to the subarray are cross-correlated (i.e. visibilities for all the baselines in the subarray are generated and transmitted to SDP).
Valid receptor IDs include: SKA dishes: “SKAnnn”, where nnn is a zero padded integer in the range of 001 to 133. MeerKAT dishes: “MKTnnn”, where nnn is a zero padded integer in the range of 000 to 063.</td></tr>
<tr><td>type</td><td colspan="2">array</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">items</td><td>type</td><td>string</td></tr>
<tr><td>pattern</td><td>^(SKA(00[1-9]|0[1-9][0-9]|1[0-2][0-9]|13[0-3])|
(MKT(0[0-5][0-9]|06[0-3]))$</td></tr>
<tr><td rowspan="2">• frequency_slice_id</td><td colspan="3">Frequency Slice to be processed on this FSP (valid range depends on the Frequency Band).</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="2">• zoom_factor</td><td colspan="3">Bandwidth to be correlated calculated as FSBW/2n, where n is in range [0..6].
When n=0 the full Frequency Slice bandwidth is correlated.
BW > 0 implies ‘Zoom Window’ configuration; the spectral Zoom Window tuning must be specified.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td rowspan="3">• zoom_window_tuning</td><td colspan="3">The Zoom Window tuning provided in absolute terms as RF center frequency. Based on that, CSP_Mid calculates tuning within the data stream received from the receptor. Must be selected so that the entire Zoom Window is within the Frequency Slice. If partially out of the FS a warning is generated. If completely outside of the FS an exception is generated.
Step size <= 0.01MHz.
The Frequency Band Offset can be used to shift the entire observed band in order to accommodate a Zoom Window that spans across a Frequency Slice boundary.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
<tr><td>default</td><td colspan="2">null</td></tr>
<tr><td rowspan="2">• integration_factor</td><td colspan="3">Integration time for the correlation products, defines multiple of 140 milliseconds.</td></tr>
<tr><td>type</td><td colspan="2">integer</td></tr>
</table>

continues on next page

Table 45 – continued from previous page

• channel_averaging_map	Table of up to 20 x 2 integers. Each of entries contains: • Start channel ID, and • averaging factor. Explanation: Each FSP produces 14880 (TBC) fine channels across the correlated bandwidth (Frequency Slice or Zoom Window). Channels are evenly spaced in frequency. TM shall provide the table that for each FSP and each group of 744 channels (there are 20 groups per FSP) indicates the channel averaging factor. More precisely, for each group the TMC provided table specifies: • the channel ID (integer) of the first channel, and • the averaging factor, as follows: – 0 means do not send channels to SDP, – 1 means no averaging, – 2 means average two adjacent channels, – 3 means average three adjacent channels, and so on. If no entry is present for an FSP, the averaging settings of the previous FSP are still applicable.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• channel_offset	Channel ID to use for visibilities of the first channel produced by this FSP. For example, if the channel offset is 5000 the first channel group would span IDs 5000-5743. Note that this offset does not apply to channel maps in this structure (such as *channelAveragingMap* or *outputHost*).				
	type	*integer*			
	default	null			
• output_link_map	Output links to emit visibilities on for every channel, given as a list of start channel ID to link ID. Where no value is given for concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_host	Output host to send visibilities to for every channel, given as a list of start channel ID to host IP addresses in dot-decimal notation. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
• output_port	Output port to send visibilities to for every channel, given as a list of start channel ID to port number. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			
	items	type	*array*		
		items	type	*integer*	
• output_mac	Output MAC address to send visibilities to for every channel, given as a list of start channel ID to IEEE 802 MAC addresses. Where no value is given for a concrete channel, the previous value should be used.				
	type	*array*			
	default	null			

continues on next page

Table 45 – continued from previous page

	items	type	*array*		
		items	anyOf	type	*integer*
				type	*string*
additionalProperties	False				

VLBI config 2.0

Very Long Baseline Interferometry specific parameters. To be borrowed from IICD This section contains the parameters relevant only for VLBI. This section is forwarded only to CSP subelement.

type	*object*	
properties		
• dummy_param	type	*string*
additionalProperties	False	

Search window config 2.0

Up to two 300 MHz Search Windows can be optionally configured and used as input for Transient Data Capture and/or Pulsar Search beam-forming.

type	*object*			
properties				
• **search_window_id**	Identifier of the 300MHz Search Window. Unique within a sub-array.			
	type	*integer*		
• **search_window_tuning**	The Search Window tuning is provided in absolute terms as RF center frequency. The Search Window must be placed within the observed band. If partially out of the observed Band a warning is generated. If completely outside of the observed Band an exception is generated.			
	type	*integer*		
• **tdc_enable**	Enable / disable Transient Data Capturefor the Search Window.			
	type	*boolean*		
• tdc_num_bits	Number of bits per sample (for the Transient Data Capture). Required if TDC is enabled, otherwise not specified.			
	type	*integer*		
	default	null		
• tdc_period_before_epoch	Users can trade the period of time for which data are saved and transmitted for the sample bit-width and/or the number of Search Windows. The exact information regarding the memory capacity per receptor and supported range will be provided in construction. The epoch is specified in the command that triggers TDC off-loading (transmission of data).			
	type	*integer*		
	default	null		
• tdc_period_after_epoch	see *tdcPeriodBeforeEpoch*			
	type	*integer*		
	default	null		
• tdc_destination_address	Destination addresses (MAC, IP, port) for off-loading of the content of the Transient Data Capture Buffer, specified per receptor. The destination addresses for the content of the Transient Data Capture can be provided either as a part of the scan configuration or by the command that triggers transmission of the captured data. The latter, if provided, overrides previously set addresses. Required if TDC is enabled, otherwise not specified.			
	type	*array*		
	default	null		
	items	anyOf	type	*integer*
			type	*string*
additionalProperties	False			

PSS configuration 2.0

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

PST configuration 2.0

Pulsar Timing specific parameters. To be borrowed from IICD This section contains the parameters relevant only for PST. This section is forwarded only to PST subelement.

type	*object*	
properties		
• dummy_param	type	*string*
	default	null
additionalProperties	False	

1.15.8 ska-tmc-releaseresources

Mid TMC resource release 2.1

Example JSON.

```
{
    "interface": "https://schema.skao.in/ska-tmc-releaseresources/2.1",
    "transaction_id": "txn-....-00001",
    "subarray_id": 1,
    "release_all": true,
    "receptor_ids": []
}
```

https://schema.skao.int/ska-tmc-releaseresources/2.1			
type	*object*		
properties			
• **interface**	URI of JSON schema applicable to this JSON payload.		
	type	*string*	
• transaction_id	A transaction id specific to the command		
	type	*string*	
	default	null	
• **subarray_id**	ID of the sub-array which should release resources.		
	type	*integer*	
• **release_all**	Scan ID to associate with the data. true to release all resources, false to release only the resources defined in this payload. Note: partial resource release for SKA Mid is not implemented and the identification of the resources to release is not yet part of the schema.		
	type	*boolean*	
• receptor_ids	empty list of receptor_ids when release_all is true		
	type	*array*	
	default	null	
	items	type	*string*
additionalProperties	False		

1.15.9 ska-tmc-scan

Mid TMC scan 2.1

Example JSON.


```
{
    "interface": "https://schema.skao.int/ska-tmc-scan/2.1",
    "transaction_id": "txn-....-00001",
    "scan_id": 1
}
```


https://schema.skao.int/ska-tmc-scan/2.1		
type	*object*	
properties		
• **interface**	URI of JSON schema applicable to this JSON payload.	
	type	*string*
• transaction_id	A transaction id specific to the command	
	type	*string*
	default	null
• **scan_id**	Scan ID to associate with the data.	
	type	*integer*
additionalProperties	False	

1.16 Telescope Layout schemas

1.16.1 ska-telmodel-layout

Telescope Layout 1.1

Example


```
{
    "interface": "https://schema.skao.int/ska-telmodel-layout/1.1",
    "telescope": "ska1_low",
    "receptors": [{
        "interface": "https://schema.skao.int/ska-telmodel-layout-receptor/1.1",
        "station_label": "FS001",
        "station_id": 1,
        "diameter": 38.0,
        "location": {
            "interface": "https://schema.skao.int/ska-telmodel-layout-location/1.0",
            "geocentric": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geocentric/1.0",
                "coordinate_frame": "ITRF",
                "x": -2563226.960308,
                "y": 5081884.949807,
                "z": -2878357.951618
            },
```


(continues on next page)

(continued from previous page)


```
            "geodetic": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                "coordinate_frame": "WGS84",
                "lat": 0.01,
                "lon": 0.01,
                "h": 1.0
            },
            "local": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-local/
↪1.0",
                "coordinate_frame": "local",
                "east": 100.0,
                "north": 10.0,
                "up": 1.0,
                "reference": {
                    "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                    "coordinate_frame": "WGS84",
                    "lat": 0.01,
                    "lon": 0.01,
                    "h": 1.0
                }
            }
        },
        "fixed_delays": [{
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }, {
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }],
        "niao": 0.0
    }, {
        "interface": "https://schema.skao.int/ska-telmodel-layout-receptor/1.1",
        "station_label": "FS001",
        "station_id": 1,
        "diameter": 38.0,
        "location": {
            "interface": "https://schema.skao.int/ska-telmodel-layout-location/1.0",
            "geocentric": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geocentric/1.0",
                "coordinate_frame": "ITRF",
```


(continues on next page)

(continued from previous page)


```
                "x": -2563226.960308,
                "y": 5081884.949807,
                "z": -2878357.951618
            },
            "geodetic": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                "coordinate_frame": "WGS84",
                "lat": 0.01,
                "lon": 0.01,
                "h": 1.0
            },
            "local": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-local/
↪1.0",
                "coordinate_frame": "local",
                "east": 100.0,
                "north": 10.0,
                "up": 1.0,
                "reference": {
                    "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                    "coordinate_frame": "WGS84",
                    "lat": 0.01,
                    "lon": 0.01,
                    "h": 1.0
                }
            }
        },
        "fixed_delays": [{
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }, {
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }],
        "niao": 0.0
    }]
}
```


Contains information required to populate a delay model used determine the relative delay between stations. Includes information such as station location, and fixed delays such as cable lengths.

https://schema.skao.int/ska-telmodel-layout/1.1		
type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **telescope**	SKA Telescope	
	type	*string*
• **receptors**	Receptors	
	type	*array*
	items	Identification, location and delay information for a receptor
		Receptor 1.1
additionalProperties	False	

Receptor 1.1

Identification, location and delay information for a receptor

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **station_label**	Receptor or station label	
	type	*string*
• **station_id**	Receptor or station identifier	
	type	*integer*
• **diameter**	Receptor or station nominal diameter (m)	
	type	*number*
• **location**	Location of receptors coordinates	
	Coordinate Locations 1.1	
• **fixed_delays**	Fixed delays	
	type	*array*
	items	A fixed delay representation, these are delays that are fixed to the station, such as cable lengths, electronic delays. This is configured to be per polarisation and the delay model can contain multiple delays and they can be stored in length or time.
		Fixed Delay 1.1
• **niao**	non-intersecting axis offset - between az and el axes	
	type	*number*
additionalProperties	False	

Coordinate Locations 1.1

A representation of the receptor position. Multiple representations are supported.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **geocentric**	Geocentric Location	
	ECEF - XYZ 1.1	
• geodetic	Geodetic location	
	default	null
	Geodetic - lat,lon,h 1.1	
• local	Local Geodetic location	
	default	null
	Local Geodetic - east, north, up 1.1	
additionalProperties	False	

ECEF - XYZ 1.1

Earth Centred Earth Fixed - Geocentric position (x,y,z) in meters. The centre of the Earth is defined by a given frame, usually a particular realisation of ITRF.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame for positions (ITRF)	
	type	*string*
• **x**	ECEF X coordinate (m)	
	type	*number*
• **y**	ECEF Y coordinate (m)	
	type	*number*
• **z**	ECEF Z coordinate (m)	
	type	*number*
additionalProperties	False	

Geodetic - lat,lon,h 1.1

Global Geodetic position schema, Geodetic coordinate systems are based on a reference ellipsoid the coordinates are geodetic latitude (rad), longitude (rad) and height (m).

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame or datum (e.g. ITRF or WGS84)	
	type	*string*
• **lat**	Geodetic latitude (rad)	
	type	*number*
• **lon**	Geodetic longitude (rad)	
	type	*number*
• **h**	height (m)	
	type	*number*
additionalProperties	False	

Local Geodetic - east, north, up 1.1

Local Geodetic position schema. Local Geodetic coordinate systems are based on a reference ellipsoid and a geodetic reference position. They are generally specified in East (E), North (N), and Up (U) in meters

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame or datum (e.g. ITRF or WGS84)	
	type	*string*
• **east**	Local Geodetic East (m)	
	type	*number*
• **north**	Local Geodetic North (m)	
	type	*number*
• **up**	Local Geodetic Height (m)	
	type	*number*
• **reference**	The geodetic reference position	
	Geodetic - lat,lon,h 1.1	
additionalProperties	False	

Fixed Delay 1.1

A fixed delay representation, these are delays that are fixed to the station, such as cable lengths, electronic delays. This is configured to be per polarisation and the delay model can contain multiple delays and they can be stored in length or time.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **fixed_delay_id**	Identification for the delay	
	type	*string*
• **polarisation**	Which polarisation this delay is applied to	
	type	*integer*
• **units**	Units for the delay (seconds, metres)	
	type	*string*
• **delay**	The delay	
	type	*number*
additionalProperties	False	

Telescope Layout 1.0

Example


```
{
    "interface": "https://schema.skao.int/ska-telmodel-layout/1.0",
    "telescope": "ska1_low",
    "receptors": [{
        "interface": "https://schema.skao.int/ska-telmodel-layout-receptor/1.0",
        "station_name": "FS001",
        "diameter": 38.0,
        "location": {
            "interface": "https://schema.skao.int/ska-telmodel-layout-location/1.0",
            "geocentric": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geocentric/1.0",
                "coordinate_frame": "ITRF",
                "x": -2563226.960308,
                "y": 5081884.949807,
                "z": -2878357.951618
            },
            "geodetic": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                "coordinate_frame": "WGS84",
                "lat": 0.01,
                "lon": 0.01,
                "h": 1.0
            },
            "local": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-local/
↪1.0",
                "coordinate_frame": "local",
                "east": 100.0,
                "north": 10.0,
                "up": 1.0,
                "reference": {
```


(continues on next page)

(continued from previous page)


```
                    "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                    "coordinate_frame": "WGS84",
                    "lat": 0.01,
                    "lon": 0.01,
                    "h": 1.0
                }
            }
        },
        "fixed_delays": [{
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }, {
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }],
        "niao": 0.0
    }, {
        "interface": "https://schema.skao.int/ska-telmodel-layout-receptor/1.0",
        "station_name": "FS001",
        "diameter": 38.0,
        "location": {
            "interface": "https://schema.skao.int/ska-telmodel-layout-location/1.0",
            "geocentric": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geocentric/1.0",
                "coordinate_frame": "ITRF",
                "x": -2563226.960308,
                "y": 5081884.949807,
                "z": -2878357.951618
            },
            "geodetic": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                "coordinate_frame": "WGS84",
                "lat": 0.01,
                "lon": 0.01,
                "h": 1.0
            },
            "local": {
                "interface": "https://schema.skao.int/ska-telmodel-layout-location-local/
↪1.0",
                "coordinate_frame": "local",
                "east": 100.0,
```


(continues on next page)

(continued from previous page)

```
                "north": 10.0,
                "up": 1.0,
                "reference": {
                    "interface": "https://schema.skao.int/ska-telmodel-layout-location-
↪geodetic/1.0",
                    "coordinate_frame": "WGS84",
                    "lat": 0.01,
                    "lon": 0.01,
                    "h": 1.0
                }
            }
        },
        "fixed_delays": [{
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }, {
            "interface": "https://schema.skao.int/ska-telmodel-layout-receptor-fixed-
↪delay/1.0",
            "fixed_delay_id": "FIX_H",
            "polarisation": 0,
            "units": "m",
            "delay": 100.0
        }],
        "niao": 0.0
    }]
}
```

Contains information required to populate a delay model used determine the relative delay between stations. Includes information such as station location, and fixed delays such as cable lengths.

https://schema.skao.int/ska-telmodel-layout/1.0		
type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **telescope**	SKA Telescope	
	type	*string*
• **receptors**	Receptors	
	type	*array*
	items	Identification, location and delay information for a receptor
		Receptor 1.0
additionalProperties	False	

Receptor 1.0

Identification, location and delay information for a receptor

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **station_name**	Receptor or station label	
	type	*string*
• **diameter**	Receptor or station nominal diameter (m)	
	type	*number*
• **location**	Location of receptors coordinates	
	Coordinate Locations 1.0	
• **fixed_delays**	Fixed delays	
	type	*array*
	items	A fixed delay representation, these are delays that are fixed to the station, such as cable lengths, electronic delays. This is configured to be per polarisation and the delay model can contain multiple delays and they can be stored in length or time.
		Fixed Delay 1.0
• **niao**	non-intersecting axis offset - between az and el axes	
	type	*number*
additionalProperties	False	

Coordinate Locations 1.0

A representation of the receptor position. Multiple representations are supported.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **geocentric**	Geocentric Location	
	ECEF - XYZ 1.0	
• geodetic	Geodetic location	
	default	null
	Geodetic - lat,lon,h 1.0	
• local	Local Geodetic location	
	default	null
	Local Geodetic - east, north, up 1.0	
additionalProperties	False	

ECEF - XYZ 1.0

Earth Centred Earth Fixed - Geocentric position (x,y,z) in meters. The centre of the Earth is defined by a given frame, usually a particular realisation of ITRF.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame for positions (ITRF)	
	type	*string*
• **x**	ECEF X coordinate (m)	
	type	*number*
• **y**	ECEF Y coordinate (m)	
	type	*number*
• **z**	ECEF Z coordinate (m)	
	type	*number*
additionalProperties	False	

Geodetic - lat,lon,h 1.0

Global Geodetic position schema, Geodetic coordinate systems are based on a reference ellipsoid the coordinates are geodetic latitude (rad), longitude (rad) and height (m).

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame or datum (e.g. ITRF or WGS84)	
	type	*string*
• **lat**	Geodetic latitude (rad)	
	type	*number*
• **lon**	Geodetic longitude (rad)	
	type	*number*
• **h**	height (m)	
	type	*number*
additionalProperties	False	

Local Geodetic - east, north, up 1.0

Local Geodetic position schema. Local Geodetic coordinate systems are based on a reference ellipsoid and a geodetic reference position. They are generally specified in East (E), North (N), and Up (U) in meters

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **coordinate_frame**	Coordinate frame or datum (e.g. ITRF or WGS84)	
	type	*string*
• **east**	Local Geodetic East (m)	
	type	*number*
• **north**	Local Geodetic North (m)	
	type	*number*
• **up**	Local Geodetic Height (m)	
	type	*number*
• **reference**	The geodetic reference position	
	Geodetic - lat,lon,h 1.0	
additionalProperties	False	

Fixed Delay 1.0

A fixed delay representation, these are delays that are fixed to the station, such as cable lengths, electronic delays. This is configured to be per polarisation and the delay model can contain multiple delays and they can be stored in length or time.

type	*object*	
properties		
• **interface**	Interface version	
	type	*string*
• **fixed_delay_id**	Identification for the delay	
	type	*string*
• **polarisation**	Which polarisation this delay is applied to	
	type	*integer*
• **units**	Units for the delay (seconds, metres)	
	type	*string*
• **delay**	The delay	
	type	*number*
additionalProperties	False	

PYTHON MODULE INDEX

S

INDEX

R

S

T

V