Proposal for a small change in Configure Json

C.Baffa, E.Giani, S.Vrcic, 2020 Jul 31

The current structure for the Configuration of CSP can be, in our view, improved a little bit . The json string has a fairly flat structure, in which all commands sent to an element are placed at or near the upper level.
This is a general proposal, but we will give CSP based examples.

Now a configure command is structured as:

```
{
        "id": "sbi-mvp01-20200325-00001-science_A",
        "frequencyBand": "1",           // CSP parameter
        "fsp": [
                {
                        "fspID": "1",
                        ….
                }
        ],
        "APSSParameter: "0",            // PSS parameter
        "PSSbeam": [
                "id": "1",
                "PSSConfigureParam": "0"
        ]
        "APSTParameter: "0",            // PST parameter
        "PSTbeam": [
                "PSTConfigureParam": "0"
        ]
}
```

We propose the structure of configuration json reflects the real structure of the target [sub]element. As an example the CSP configure would have a structure as:

```
{
        "csp": {        // portion common to the whole CSP
                        "frequencyBand": "1",           // CSP parameter
                        …
        }
        "cbf": {        // Portion specific to CSP.CBF
                        ….
                         "fsp": [       // Portion specific to CSP.CBF.FSB subsystems
                                {
                                        "fspID": "1",
                                        ….
                                }
                        ],
        },
        "pst": {        // Portion specific to CSP.PST
                …
        },
        …
        "pss":{         // Portion specific to CSP.PSS
                …
        },
        …
}
```

The proposed structure, while a little more complex has the following advantages:

1. For the producing entity (in our example OET) it is simpler to organize and maintain the code which configure a specific part. The code relative to a [sub]sub-element only acts on a [sub]branch of the complete json, so there is a gain in code independence, with obvious gain in code clarity and testability.
2. For the target entity the gain is analogous, as the sub-elements decode portion does need to know only a specific ('local') portion of the possible commands and the global (CSP-portion) parameters. This simplify the code with improvement in code clarity and testability. At the CSP level the portion which forwards the

commands to sub-elements do not any more decode every specific command to know its destination element, it only consists of the extraction of a specific element from a larger structure. There is still the need for a CBF-PSS and CBF-PST consistency check. But this is already in place.

3. For testing and human inspection it is much simpler to collect all related item in a single branch of the complete json. With this structure it is simpler to perform a content and protocol check. Also if correctly formatted the human inspection, for documentation or debug, gains from the single location of various related items.

As final suggestion we note the present day and the proposed format can coexist without harming each other. It is so possible to foresee a transition phase in which both forms are present and a not updated sub-element can continue to function also during the period in which the system migrate from a format to another.

Appendix 1: CSP-specific coherence checks

We put here some note on CSP specific sub-element cross check. This portion is common to present day and to the proposed structure. However in the second case it is somehow simpler to implement. We report, as an example, the Pulsar Search configuration:

a) Search Window bandwidth is fixed (300MHz and does not need to be specified). If PSS software requires bandwidth to be specified, then CSP.LMC could define the Search Window bandwidth as constant and use that constant to insert BW in the PSS sub-element configuration.
b) The same applies for the number of channels per beam - Mid.CBF will always produce the same number of channels (4096).
c) The placement of the Search Window within the observed band is configurable (and is part of Mid.CBF configuration). Mid.CBF can form up to two 300 MHz Search Windows, each can be independently placed anywhere within the observed band. For each beam to be formed Mid.CBF has to be told which Search Window to use as input. CSP.LMC should verify that Search Beam centre frequency is consistent in CBF and PSS configuration (for each beam).
d) To reduce the output data rate Mid.CBF implements averaging in time. Integration factor for CBF time averaging is configurable parameter and should be consisted with 'time resolution' specified in PSS configuration.
e) In some cases only a sub-set of antennas that belong to the subarray will be used for Pulsar Search, the list should be the same in CBF configuration (provided for each FSP) and in PSS configuration

Appendix 2: present day configuration format

```
// NEW FROM ADR-4
// "id" is an ID for this CSP configuration that may be used by CSP to aid in tracing any problems.
// It is comprised of the SBI ID concatenated with the scan_type ID from the SDP resource setup that
// directly relates that SDP setup to this CSP configuration.
// Note that this configuration might be re-used within an SBI execution. The ID will remain the
same.
{
    "id": "sbi-mvp01-20200325-00001-science_A",
    // Should we omit CSP frequencyBand or DISH frequencyBand? One could be
    // derived from the other.
    "frequencyBand": "1",
    "fsp": [
      {
        "fspID": 1,
        "functionMode": "CORR",  // Set FSP to correlation mode
        // Since receptors are not given, FSP uses all receptors assigned to the
        // subarray
```

```
      "frequencySliceID": 1,   // Tell FSP to process frequency slice #1
      "integrationTime": 1400  // Set FSP to 1400ms integration time.
      "corrBandwidth": 0       // Correlate the entire frequency slice
      //
      // Send the minimum possible number of channels to SDP by averaging
      // the first 744 fine channels down to 372 channels (=744/2). Do not
      // send any other fine channel groups to SDP (=<chan ID>,0).
      // THIS HAS NOW ADDED THE OUTPUTLINKMAP INFORMATION. IT HAS ALSO COMPRESSED THE FORMAT:
      // FROM ADR-4: "So the link map would work very similarly to the averaging map: To find
      // the link for a (fine) channel select the value associated with the last channel ID in
      // the map that is less than or equal to the looked-up channel ID. In contrast to the
      // averaging map, channel IDs would not have to correspond to align to channel groups
      // (here 744). This should be a relatively compact way to represent the map."
      //
      "channelAveragingMap": [[1,2], [745,0]],
      "outputLinkMap": [[1,0], [201,1]]
    }, // end FSP 1 configuration
    {
      "fspID": 2,
      "functionMode": "CORR",
      "frequencySliceID": 2,
      "integrationTime": 1400,
      "corrBandwidth": 0
      // Since channel averaging has not been configured, no products from FSP
      // 2 are sent to SDP
    }, // end FSP 2 configuration
  ] // end FSP configuration
}
```