

GS L1a-to-L1b RPWI Calibration Software Interface Control Document for Radio & Plasma Wave Investigation (RPWI) JUICE

Leading Funding Agency (LFA): Swedish National Space Board (SNSB)
Box 4006, SE-171 04 Solna, Sweden
Tel.: +46-(0)8-627 6480, Fax: N/A.
Email: rymdstyrelsen@snsb.se

LFA Contact: Johan Köhler, Email: Johan.Kohler@snsb.se

Jan-Erik Wahlund (PI)
Swedish Institute of Space Physics
Box 537, SE-751 21 Uppsala, Sweden
Tel.: +46-(0)18-471 5946
+46-(0)76-76 97 877
Fax: +46-(0)18-471 5905
Email: jwe@irfu.se

Jan E. S. Bergman (IM)
Swedish Institute of Space Physics
Box 537, SE-751 21 Uppsala, Sweden
Tel.: +46-(0)18-471 5916
+46-(0)70-746 65 69
Fax: +46-(0)18-471 5905
Email: jb@irfu.se

Baptiste Cecconi (Co-PI)
LESIA-Observatoire de Paris, 5 place Jules
Janssen, 92190 Meudon, France
Tel.: +33-(0)1-45 07 77 59
Fax: +33-(0)1-45 07 28 06
Email: baptiste.cecconi@obspm.fr

Ondrej Santolik (Co-PI)
Institute of Atmospheric Physics
Bocni II/1401, 12131 Prague 4-Sporilov
Czech Republic
Tel.: +420 267 103 083
Fax: +420 272 763 745

Email: os@ufa.cas.cz

Yasumasa Kasaba (Co-PI)

Planetary Plasma and Atmospheric Research
Center (PPARC), Tohoku University
Graduate School of Science, 6-3 Aramaki aza
Aoba, Aoba-ku, Sendai 980-8578, Japan
Tel.: +81-22-795 6738
Email: kasaba@pparc.gp.tohoku.ac.jp

Hanna Rothkaehl (Co-PI)

Space Research Centre of the Polish
Academy of Sciences
00-716 Bartycka 18A, Warsaw, Poland
Tel.: +48-22-4966-418
Fax: +48-22 8403-131
Email: hrot@cbk.waw.pl

Ingo Müller-Wodarg (Co-PI)

Space and Atmospheric Physics group
Imperial College London, London, UK
Tel.: +44-(0)20-7594 7674
Fax: N/A
Email: i.mueller-wodarg@imperial.ac.uk

APPROVAL

Title: RPWI Calibration Software Interface Control Document	
Issue: 0	Revision: 2
Author: Erik P G Johansson	Date: 2024-02-29
Approved by:	Date:
XXX Electronic signature of approver XXXX	

CHANGE LOG

Reason for change	Issue	Revision	Date
Initial draft version.	0	1	2024-02-28
IRF-internal review	0	2	2024-02-29

CHANGE RECORD

Issue X	Revision X	
Reason for change	Date	Paragraph(s)
Issue X+1	Revision X+1	
Reason for change	Date	Paragraph(s)
XXX Itemised description of change		
XXX Extend or delete sections of table as necessary, keeping change record for all issues/revisions		

DISTRIBUTION LIST

Issue 0	Revision 2	Date 2024-02-29
Recipient	Affiliation	Comments
JUICE/RPWI GS Team		
Issue X+1	Revision X+1	
Reason for change	Affiliation	Comments
XXX Extend or delete sections of table as necessary, keeping distribution list for all issues		

Table of Contents

1	INTRODUCTION.....	6
2	Applicable and Reference Documents.....	6
2.1	Applicable Documents.....	6
2.2	Reference Documents.....	6
3	GENERAL REQUIREMENTS.....	6
3.1	Software Format and Packaging.....	6
3.2	In-flight and Ground-test Data.....	7
3.3	Output Dataset Compliance with Archiving Standards.....	7
3.4	Error Handling.....	7
3.5	Log File.....	7
3.6	Unit tests: pytest.....	7
3.7	Source Code Documentation: sphinx.....	8
3.8	Code Style Validation: flake8 (PEP 8).....	8
3.9	Function for Calling the RCS.....	8
3.9.1	Dataset IDs.....	9
3.9.2	Processing Modes.....	9
3.9.3	Content of a Request File.....	9
3.9.4	Content of a Response File.....	11
4	DELIVERY OF THE RCS.....	11
4.1	Versioning.....	11
5	NOT YET ADDRESSED (TBD) REQUIREMENTS.....	12
6	RATIONALE.....	12
6.1	Using Separate Python Environments.....	12
6.2	Using JSON Files.....	12
6.3	Ability to Generate Empty Datasets.....	13
6.4	Potential Generalized Use of the Interface.....	13
7	EXAMPLE JSON FILES AND SCHEMAS.....	13
7.1	Example Request File.....	13
7.2	Example Response File.....	15
7.3	JSON Schema File for Request Files.....	16
7.4	JSON Schema File for Response Files.....	19
8	ACRONYMS.....	22

1 INTRODUCTION

The RPWI GS requires the RPWI teams to contribute pieces of software for producing L1b datasets (CDF files) from relevant and pre-existing L1a datasets (CDF files), HK (CDF files), and SPICE kernels. Such an instance of a team-contributed s/w will here be referred to as an RPWI Calibration Software (RCS). This document describes requirements on the RCS's, in particular the interface for calling them. The main RPWI GS L1a-to-L1b s/w (TBD) will make use of this interface to call the RCS's.

This document does *not* describe the details of the actual processing of data from dataset archiving levels L1a to L1b, such as calibration methods or dataset design except for general references to metadata standards.

Any software which calls an RCS using the interface described in this document will be referred to as “the caller” in this document. Only the main RPWI GS L1a-to-L1b s/w is expected to actually call the RCS's using this interface.

2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable Documents

- AD-1 “Project Management Plan”, JUI-IRFU-RPWI-PL-011, Issue 3.1, dated 2016-05-26
- AD-2 “JUICE RPWI-SOC Data Processing Agreement”, JUI-ESAC-SGS-MO-002, Issue 1.0, dated 2019-10-31
- AD-3 “PSA PDS4 ARCHIVING GUIDE”, ESDC-PSA-TN-0002, Issue 2.5, dated 2019-11-18
- AD-4 “JUICE Annex to PSA PDS4 Archiving Guide”, JUI-ESAC-SGS-TN-029, Issue 1.1
- AD-5 “JUICE PI-SOC Interface Control Document”, JUI-ESAC-SGS-ICD-001, Issue 1.1, Section 5.2.3.2
- AD-6 “ISTP Guidelines Page”, https://spdf.gsfc.nasa.gov/istp_guide/
- AD-7 “Sphinx documentation”, <https://www.sphinx-doc.org/en/master/>
- AD-8 “Flake8: Your Tool for Style Guide Enforcement”, <https://flake8.pycqa.org/en/latest/>

2.2 Reference Documents

- RD-1 “Specification for Science Data Unpacking S/w (SDUS) for the RPWI pipeline”, https://www.space.irfu.se/juice/rpwi_pipeline/autoapi/rpwi/l0tol1a/sci/sss/template/index.html

3 GENERAL REQUIREMENTS

3.1 Software Format and Packaging

An RCS shall:

- Be a Python application in the form of one Python distribution package.
- Be able to run on Linux.

- Not use software which requires a commercial license.
 - Footnote: This is a requirement for running the s/w at SOC.
- Use a Python version specified by the RCS itself. The Python version must be 3.12 or later.
 - Note: The lowest allowed Python version should be expected to increase with time and the teams shall thus expect to need to upgrade the RCS over time to satisfy this requirement.
 - Note: The method for specifying the Python version for the main RPWI GS L1a-to-L1b s/w is TBD.
- Only place code in Python package `rpwi.l1atol1b.sss.<RCSID>` and/or below it, where `<RCSID>` is an import package name which is unique for every RCS. `<RCSID>` will be one of the following string constants which represents the corresponding RCS: `lp`, `lf`, `hf`, `mm`.
- Supply one function for calling the RCS as described in Section 3.9.

Note: An RCS is allowed to:

- Specify its own Python dependencies (other Python distribution packages) available on PyPI, including the exact versions of these dependencies, independent of the dependencies of other RCS's.

3.2 In-flight and Ground-test Data

An RCS shall be able to process both (a) in-flight data and (b) ground-test data.

Note:

- In-flight data: Requires using SPICE kernels for time conversions. Official and unofficial datasets will be generated from this data.
- Ground-test data: Requires (nominally) *not* using SPICE kernels for time conversions. No official datasets will be generated from this data.

3.3 Output Dataset Compliance with Archiving Standards

An RCS shall produce L1b datasets (CDF files) which are consistent with relevant archiving standards described in AD-4 (e.g. Section 5) and AD-3, including ISTP (AD-6).

- Note: Some ISTP global attribute *values* will be received from the caller. See “`cdf_global_attributes`” in Section 3.9.3.
- Note: The RCS shall not produce PDS4 label files, but instead provide metadata which will help the main L1a-to-l1b produce the same. The convention for doing this is TBD.

3.4 Error Handling

The RCS default behaviour shall be to raise exception when encountering data which it cannot process.

3.5 Log File

The log file generated by the RCS must conform to the log file format specified in AD-5.

3.6 Unit tests: pytest

An RCS shall contain relevant unit tests using pytest.

3.7 Source Code Documentation: sphinx

The source code for an RCS shall be (reasonably) documented using documentation strings which can be parsed by sphinx (AD-7).

3.8 Code Style Validation: flake8 (PEP 8)

The source code to an RCS code shall pass the code style validation tests of flake8 (AD-8).

3.9 Function for Calling the RCS

An RCS must supply one callable Python function:

```
rpwi.l1atol1b.sss.<RCSID>.rcs.process_request(request_file_path: str)
-> None:
```

where <RCSID> is described in Section 3. Argument `request_file_path` is a string path to an existing “request file” created by the caller and described in Section 3.9.3. The function must have no Python-language return value but must instead create a corresponding response file as described in Section 3.9.4. The path to the response file is described in the request file. See Figure 1.

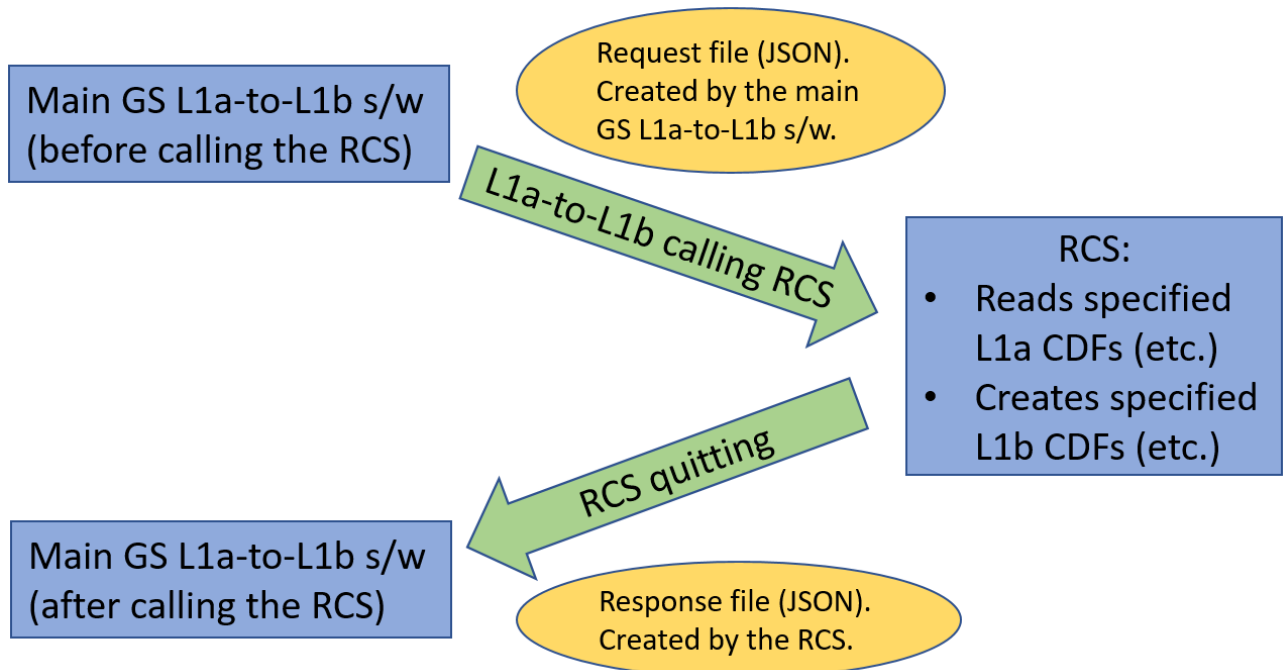


Figure 1. Main GS L1a-to-L1b s/w calling an RCS for processing datasets. JSON files are used for exchanging information between both sides.

If an error occurs in the RCS during the call to this function, then the RCS should raise an appropriate Python exception with a relevant human-readable text message. See Section 3.4.

3.9.1 Dataset IDs

A “Dataset ID” is a string constant which represents one particular type of dataset. This is the same concept as in the interface between the main TM-to-L1a s/w and SDUS’s (RD-1).

Note: It is expected that these will be closely related to, or identical to, ISTP global attribute Logical_source and some similar PDS4 variable. It is therefore likely that these will at some point need to be redefined (replaced) to be consistent with ISTP and PDS4 for simplicity.

3.9.2 Processing Modes

The RCS shall organize the available forms of processing as “processing modes”. A “processing mode” defines a way where the RCS reads N_{in} input datasets (with unique pre-defined dataset IDs) and from them produces N_{out} output datasets (with unique pre-defined dataset IDs).

The RCS should support three commands (“command_id” in the request/response files):

- GET_ALL_PROCESSING_MODES: The RCS shall return a list of available processing modes and corresponding metadata.
- PROCESS_DATASETS: The RCS processes data using one of the processing modes advertised in the response to GET_ALL_PROCESSING_MODES.
- GENERATE_EMPTY_DATASETS: Generate empty datasets for the specified dataset IDs. This only refers to those datasets (dataset IDs) which are specified as *output* datasets in all of the supported processing modes combined. The generated datasets must be consistent with the same datasets when generated with command PROCESS_DATASETS (CDF metadata, set of zVariables, relevant dimensions of zVariables, number of dimensions of zVariables).

3.9.3 Content of a Request File

A “request file” is a valid JSON file which is created by the caller and is submitted to the RCS by submitting the path to it. The file shall be read by (and not modified by) the RCS. The request file contains a list of commands which the RCS shall execute in order. Depending on the type of command, each command is associated with a number of parameters which further describe what the RCS is expected to do, e.g. which output CDF files to create.

The exact content of a request file is described by the example in Section 7.1. A request file must validate against the corresponding JSON schema found in Section 7.3 which thus also describes the exact format of request files.

Remarks on the content of request files (see Section 7.1):

- request_format_version: Non-negative integer which specifies a specific version of the request file format. This number should be incremented every time the request file format is officially changed.
 - An RCS can optionally check this number before trying to parse the rest of the request file. Note that an RCS can potentially simultaneously support multiple request formats at the same time, both with and without looking at this value.
 - It should be enough to support the most recent version to be compatible with the most recent version of the main L1a-to-L1b s/w.
- response_file_path: The path to the response file which the RCS shall create.

- `spice_kernel_file_paths`: Array of paths to SPICE kernel files. The RCS must load the specified SPICE kernel files *in the specified order* if it uses SPICE. It is expected that there will normally only be one SPICE kernel file (a metakernel) specified but there could be more.
- `center_file_index`: The caller may supply the RCS with multiple consecutive input datasets for the same dataset ID if they are available (`file_paths`), despite that the RCS should only create one output dataset (for each output dataset ID). The time interval of the output dataset should correspond to the time interval of the (relevant) input dataset numbered `center_file_index`, where 0=first input dataset in array.
 - This feature can be used by calibration algorithms which need to consider a larger time interval of data than covered in a single input L1a datasets in order to produce one output L1b output dataset(s). This can help with correctly calibrating data at the time interval edges using e.g. transfer functions. Calibration which does not do this can ignore input datasets not numbered `center_file_index`.
- `pds4_vid`: PDS4 “Version identifier” (“Version ID”, or “VID”). Dataset version on format *m.n*, where *m* and *n* are nonnegative integers. See AD-3, Section 4.4.
 - Note: It is currently unclear how or if this will be used by the RCS, but it will likely be used for some CDF metadata.
- `rsc_log_execution_id`: String “Execution ID” needed for generating log file on the correct format. See Section 3.5.
- `custom_rsc_settings`: List of key-value pairs describing optional custom RCS settings. Such settings can optionally be defined by the RCS and optionally be passed on from the caller to the RCS.
 - This feature can be useful for e.g. debugging or for custom-processing such as enabling unofficial processing modes, enabling unofficial output datasets, disabling calibration, enabling experimental implementations, enabling mitigations, disabling failing/incomplete functionality etc.
 - Note: An RCS can ignore this feature if not used.
 - Note: Nominal (official) processing of datasets from L1a to L1b will not pass on any custom settings to the RCS. This feature, if used, is only intended for unofficial processing.
- `cdf_global_attributes`: Describes CDF global attributes which the RCS must write to the corresponding output CDF file *in addition to* any global attributes which the RCS itself will generate.
 - Note: If both the RCS itself (internally) and the request file specifies values for the same global attribute, then the request file’s global attribute should take precedence.
 - A global attribute value specified may only be an array of strings. This may be expanded if needed.
 - The exact set of global attributes which will be set by the main L1a-to-L1b s/w for official processing in this way is TBD, but is likely to include CDF attributes which are constant over all datasets, e.g. ISTP’s “PI_name”, and global attributes which follow simple rules which apply to all datasets, e.g. global attributes which correlate with official filenames and dataset versions.

3.9.4 *Content of a Response File*

A “response file” is a valid JSON file which is created by the RCS. It contains a list of entries, each entry containing information on the outcome of each of the corresponding commands specified in the corresponding the request file, and in the same order. The path to the response file is specified in the request file (Section 3.9.3).

The exact content of a response file is described by the example in Section 7.2. A response file must validate against the corresponding JSON schema found in Section 7.4 which thus also describes the exact format of response files.

Remarks on the content of response files (see Section 7.2):

- `response_format_version`: Non-negative integer which specifies a specific version of the response file format. This number should be incremented every time the response file format is changed.
- `archiving_level`: The response to a command request `GET_ALL_PROCESSING_MODES` should specify the input and output dataset archiving levels. This should nominally always be L1a (input) or L1b (output) (and possibly “HK” for HK input). It is included in case the interface is generalized to support more archiving levels in the future (see Section 6.4).
- `time_of_generation_utc`: The time the JSON response file is generated (roughly), i.e. after that bulk processing has completed.

4 DELIVERY OF THE RCS

An RCS must be delivered to IRF-U using a git repo in the GitLab instance hosted on IRFU server `spis.irfu.se`. The git repo shall be named:

`rpwi_pipeline_l1a-to-l1b_<RCSID>` (“RPWI Pipeline L1a-to-L1b <RCSID>”)

where <RCSID> is described in Section 3.

4.1 Versioning

Each formal delivery of an RCS from the teams to IRFU shall be associated with a version number on the standard form X.Y.Z, where

- X=major version (major changes or backward-incompatible changes),
- Y=minor version (backward compatible), and
- Z=minor revision (backward compatible; typically bugfixes).

This version number shall at the very least be used for:

- Tagging (git; annotated tag) the corresponding git commit for official deliveries of the s/w intended to be used in the main RPWI GS L1a-to-L1b pipeline. Tags shall be on the format “vX.Y.Z” (notice the “v”). Such tags shall be pushed to the server along with the corresponding commit.

- Setting the Python distribution package version number (typically set in file setup.cfg though there may be other ways). This shall always be consistent with the most recent git-tagged version.
- Setting the sphinx documentation version number in docs/source/conf.py. This shall always be consistent with the most recent git-tagged version.

All official s/w versions shall be on the git repo master branch.

5 NOT YET ADDRESSED (TBD) REQUIREMENTS

This ICD does not yet address the following TBD requirements:

- How the RCS shall specify its Python version. See Section 3.1.
- How to specify which calibration data the RCS shall use (if external to the code).
- Which CDF global attributes shall be specified by the caller in the request file (for official processing). See Section 3.9.3.

6 RATIONALE

6.1 Using Separate Python Environments

Allowing each RCS to specify both (1) its own Python version, and (2) Python dependencies, including versions of dependencies, has some important disadvantages for the main L1a-to-L1b s/w and but also advantages which should make the project easier to manage over the long run, given (1) the number of teams involved, and (2) the long time horizon over which s/w needs to be maintained.

Advantages:

- RCS's and main L1a-to-L1b s/w are allowed to have conflicting dependencies (Python distribution packages), in particular dependency conflicting version requirements.
- RCS's and the main L1a-to-L1b s/w are allowed to have conflicting Python version requirements.
- The main L1a-to-L1b s/w can potentially simultaneously run older and newer RCS's.

Disadvantages:

- Requires the main GS L1a-to-L1b s/w to provide separate Python environments for each RCS.
- Makes calls from the main GS L1a-to-L1b s/w to the RCS's harder to implement.
- Add adds overhead to every call from the main GS L1a-to-L1b s/w to an RCS.

6.2 Using JSON Files

Using JSON files for communicating with the RCS has multiple advantages over using e.g. command-line arguments and stdout:

- Easy to send complex hierarchical data structures with named variables between caller and RCS.
- Easy and safe to update the interface if needed, including updating s/w to support interface updates. Easier to handle backward-compatibility if needed.

- Easy to log the communication between RCS and caller by simply saving the request and response files.
- The format of the JSON files can be described (documented) by, as well programmatically verified by, using JSON schemas.

6.3 Ability to Generate Empty Datasets

Being able to generate empty datasets are useful for the GS maintaining an overview of *all* supported datasets and their basic design, e.g. for documentation and inspection of metadata.

6.4 Potential Generalized Use of the Interface

The s/w interface described in this document is designed for the purpose of automatizing the production of L1b datasets (output) from relevant L1a datasets etc. (input). It is however designed with the intent of also being reusable for automatizing the production of higher-level data products if convenient. It is not anticipated that such processing will take place at SOC, but within the RPWI consortium, or some subset of the RPWI consortium.

7 EXAMPLE JSON FILES AND SCHEMAS

The JSON files listed in this section can also be found in git repo `rpwi_pipeline`, subdirectory `src/rpwi/llatol1b/data/` in GitLab at `spis.irfu.se`. The exact version may differ for different git commits and branches depending on what the main L1a-to-L1b s/w supports in that exact version. Below JSON files come from commit `3dc2b9a5`.

Note: The specific combination of commands in the request and response example files in Sections 7.1 and 7.2 make little sense from the caller's perspective, but the combination is legal and the commands illustrate the usage of the commands separately.

Note: The JSON schema files require the exact set of specified properties in most cases, but not all of them.

7.1 Example Request File

```
{
  "request_format_version": 0,
  "time_of_generation_utc": "2032-02-10T00:00:00Z",
  "target_rcs_id": "LP",
  "rcs_log_file_path": "/rcs_logs/rcs_log_file_20320210T000000.log",
  "rcs_log_execution_id": "0123456789",
  "response_file_path": "/response_files/response_file_20320210T000000.json",
  "spice_kernel_file_paths": [
    "/data/juice/SPICE/kernels/mk/juice_ops.tm",
    "/data/juice/SPICE/kernels/lsc/naif0012.tls"
  ],
  "custom_rcs_settings": {
    "LP_RCS_CUSTOM_SETTING_BOOLEAN": false,
    "LP_RCS_CUSTOM_SETTING_INTEGER": 123,
    "LP_RCS_CUSTOM_SETTING_STRING": "string",
    "LP_RCS_CUSTOM_SETTING_ARRAY": [0, 1, 2, 3]
  },
  "commands": [
    {
```

```
"command_id": "GET_ALL_PROCESSING_MODES",
"command_arguments": {}
},
{
  "command_id": "PROCESS_DATASETS",
  "command_arguments": {
    "processing_mode_id": "EXAMPLE_PROCESSING_MODE_1",
    "input_datasets": {
      "DATASET_ID_11": {
        "center_file_index": 1,
        "file_paths": [
          "/input_datasets/DATASET_ID_11_20320101.cdf",
          "/input_datasets/DATASET_ID_11_20320102.cdf",
          "/input_datasets/DATASET_ID_11_20320103.cdf"
        ]
      }
    },
    "output_datasets": {
      "DATASET_ID_12": {
        "path": "/output_datasets/DATASET_ID_12_20320102.cdf",
        "pds4_vid": "0.1",
        "cdf_global_attributes": {}
      },
      "DATASET_ID_13": {
        "path": "/output_datasets/DATASET_ID_13_20320102.cdf",
        "pds4_vid": "0.2",
        "cdf_global_attributes": {
          "PI_name": [
            {
              "value": "Wahlund, J.-E.",
              "cdf_data_type": "CDF_CHAR"
            }
          ]
        },
        "GA_multiple_strings": [
          {
            "value": "GA string value 1",
            "cdf_data_type": "CDF_CHAR"
          },
          {
            "value": "GA string value 2",
            "cdf_data_type": "CDF_CHAR"
          }
        ]
      }
    }
  }
},
{
  "command_id": "GENERATE_EMPTY_DATASETS",
  "command_arguments": {
    "output_datasets": {
      "DATASET_ID_21": {
        "path": "/output_datasets_empty/DATASET_ID_23_EMPTY.cdf",
        "pds4_vid": "0.1",
        "cdf_global_attributes": {}
      },
      "DATASET_ID_22": {
        "path": "/output_datasets_empty/DATASET_ID_23_EMPTY.cdf",
        "pds4_vid": "0.2",
        "cdf_global_attributes": {
          "PI_name": [
            {

```

```
        "value": "Wahlund, J.-E.",
        "cdf_data_type": "CDF_CHAR"
    },
    ],
    "GA_multiple_strings": [
        {
            "value": "GA string value 1",
            "cdf_data_type": "CDF_CHAR"
        },
        {
            "value": "GA string value 2",
            "cdf_data_type": "CDF_CHAR"
        }
    ]
}
}
```

7.2 Example Response File

```
{
  "response_format_version": 0,
  "time_of_generation_utc": "2032-02-11T00:00:00Z",
  "source_rcs_id": "LP",
  "commands": [
    {
      "command_id": "GET_ALL_PROCESSING_MODES",
      "command_return_values": {
        "processing_modes": {
          "EXAMPLE_PROCESSING_MODE_1": {
            "input_datasets": {
              "DATASET_ID_11": {
                "archiving_level": "L1a"
              }
            },
            "output_datasets": {
              "DATASET_ID_12": {
                "archiving_level": "L1b"
              },
              "DATASET_ID_13": {
                "archiving_level": "L1b"
              }
            }
          },
          "EXAMPLE_PROCESSING_MODE_2": {
            "input_datasets": {
              "DATASET_ID_21": {
                "archiving_level": "L1a"
              },
              "DATASET_ID_22": {
                "archiving_level": "L1a"
              }
            },
            "output_datasets": {
              "DATASET_ID_23": {
                "archiving_level": "L1b"
              }
            }
          }
        }
      }
    }
  ]
}
```

```
    }  
  },  
  {  
    "command_id": "PROCESS_DATASETS",  
    "command_return_values": {}  
  },  
  {  
    "command_id": "GENERATE_EMPTY_DATASETS",  
    "command_return_values": {}  
  }  
]  
}
```

7.3 JSON Schema File for Request Files

```
{  
  "type": "object",  
  "properties": {  
    "request_format_version": {  
      "const": 0  
    },  
    "time_of_generation_utc": {  
      "type": "string",  
      "pattern": "^20[0-9][0-9]-[01][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9]:[0-6][0-9]Z$"   
    },  
    "target_rcs_id": {  
      "enum": ["LP", "LF", "HF", "MM"]  
    },  
    "response_file_path": {  
      "type": "string"  
    },  
    "rcs_log_file_path": {  
      "type": "string"  
    },  
    "rcs_log_execution_id": {  
      "type": "string",  
      "pattern": "^([0-9]{10})$"   
    },  
    "spice_kernel_file_paths": {  
      "type": "array",  
      "items": {  
        "type": "string"  
      }  
    },  
    "custom_rcs_settings": {  
      "type": "object",  
      "patternProperties": {  
        "^[A-Z0-9._]+$": {}  
      },  
      "additionalProperties": false  
    },  
    "commands": {  
      "type": "array",  
      "items": {  
        "oneOf": [  
          {  
            "type": "object",  
            "properties": {  
              "command_id": {  
                "const": "GET_ALL_PROCESSING_MODES"  
              },  
              "command_arguments": {  

```

```

        "type": "object",
        "properties": {},
        "required": [],
        "additionalProperties": false
    },
    },
    "required": ["command_arguments", "command_id"],
    "additionalProperties": false
},
{
    "type": "object",
    "properties": {
        "command_id": {
            "const": "PROCESS_DATASETS"
        },
        "command_arguments": {
            "type": "object",
            "properties": {
                "processing_mode_id": {
                    "type": "string",
                    "pattern": "^[A-Za-z][A-Za-z0-9_-]*$"
                },
                "input_datasets": {
                    "type": "object",
                    "patternProperties": {
                        "^[A-Za-z][A-Za-z0-9_-]*$": {
                            "type": "object",
                            "properties": {
                                "center_file_index": {
                                    "type": "integer",
                                    "minimum": 0
                                },
                                "file_paths": {
                                    "type": "array",
                                    "items": {
                                        "type": "string"
                                    }
                                }
                            }
                        }
                    },
                    "required": ["center_file_index", "file_paths"],
                    "additionalProperties": false
                }
            },
            "additionalProperties": false
        },
        "output_datasets": {
            "type": "object",
            "patternProperties": {
                "^[A-Za-z][A-Za-z0-9_-]*$": {
                    "type": "object",
                    "properties": {
                        "path": {
                            "type": "string"
                        },
                        "pds4_vid": {
                            "type": "string",
                            "pattern": "^[0-9]+\\. [0-9]+$"
                        }
                    },
                    "cdf_global_attributes": {
                        "type": "object",
                        "patternProperties": {
                            "^[a-zA-Z0-9_]+$": {
                                "type": "array",

```

```
        "items": {
          "type": "object",
          "properties": {
            "value": {
              "type": "string"
            },
            "cdf_data_type": {
              "type": "string"
            }
          },
          "required": ["cdf_data_type", "value"],
          "additionalProperties": false
        }
      },
      "additionalProperties": false
    },
    "required": [
      "cdf_global_attributes",
      "path",
      "pds4_vid"
    ],
    "additionalProperties": false
  },
  "additionalProperties": false
},
"required": [
  "input_datasets",
  "output_datasets",
  "processing_mode_id"
],
"additionalProperties": false
},
"required": ["command_arguments", "command_id"],
"additionalProperties": false
},
{
  "type": "object",
  "properties": {
    "command_id": {
      "const": "GENERATE_EMPTY_DATASETS"
    },
    "command_arguments": {
      "type": "object",
      "properties": {
        "output_datasets": {
          "type": "object",
          "patternProperties": {
            "^[A-Za-z][A-Za-z0-9_-]*$": {
              "type": "object",
              "properties": {
                "path": {
                  "type": "string"
                }
              },
              "pds4_vid": {
                "type": "string",
                "pattern": "^[0-9]+\\.\\.[0-9]+$"
              }
            },
            "cdf_global_attributes": {
```

```

        "type": "object",
        "patternProperties": {
          "^[a-zA-Z0-9_]+$": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "value": {
                  "type": "string"
                },
                "cdf_data_type": {
                  "type": "string"
                }
              },
              "required": ["cdf_data_type", "value"],
              "additionalProperties": false
            }
          }
        },
        "additionalProperties": false
      },
      "required": [
        "cdf_global_attributes",
        "path",
        "pds4_vid"
      ],
      "additionalProperties": false
    },
    "additionalProperties": false
  },
  "required": ["output_datasets"],
  "additionalProperties": false
},
"required": ["command_arguments", "command_id"],
"additionalProperties": false
}
]
}
},
"required": [
  "commands",
  "custom_rcs_settings",
  "rcs_log_execution_id",
  "rcs_log_file_path",
  "request_format_version",
  "response_file_path",
  "spice_kernel_file_paths",
  "target_rcs_id",
  "time_of_generation_utc"
],
"additionalProperties": false
}

```

7.4 JSON Schema File for Response Files

```

{
  "type": "object",
  "properties": {

```

```

"response_format_version": {
  "const": 0
},
"time_of_generation_utc": {
  "type": "string",
  "pattern": "^20[0-9][0-9]-[01][0-9]-[0-3][0-9]T[0-2][0-9]:[0-5][0-9]:[0-6][0-9]Z$"
},
"source_rcs_id": {
  "enum": ["LP", "LF", "HF", "MM"]
},
"commands": {
  "type": "array",
  "items": {
    "oneOf": [
      {
        "type": "object",
        "properties": {
          "command_id": {
            "const": "GET_ALL_PROCESSING_MODES"
          },
          "command_return_values": {
            "type": "object",
            "properties": {
              "processing_modes": {
                "type": "object",
                "properties": {},
                "required": [],
                "patternProperties": {
                  "^[A-Za-z][A-Za-z0-9_-]*$": {
                    "type": "object",
                    "properties": {
                      "input_datasets": {
                        "type": "object",
                        "additionalProperties": false,
                        "patternProperties": {
                          "^[A-Za-z][A-Za-z0-9_-]*$": {
                            "type": "object",
                            "properties": {
                              "archiving_level": {
                                "oneOf": [
                                  {
                                    "const": "L1a"
                                  },
                                  {
                                    "const": "L1b"
                                  }
                                ]
                              }
                            }
                          }
                        },
                        "required": ["archiving_level"],
                        "additionalProperties": false
                      }
                    }
                  }
                }
              }
            }
          }
        }
      },
      {
        "type": "object",
        "properties": {
          "output_datasets": {
            "type": "object",
            "patternProperties": {
              "^[A-Za-z][A-Za-z0-9_-]*$": {
                "type": "object",
                "properties": {
                  "archiving_level": {
                    "oneOf": [
                      {
                        "const": "L1a"
                      },
                      {
                        "const": "L1b"
                      }
                    ]
                  }
                }
              }
            }
          }
        }
      }
    ]
  }
}

```

```
        "const": "L1a"
      },
      {
        "const": "L1b"
      }
    ]
  },
  "required": ["archiving_level"],
  "additionalProperties": false
},
"additionalProperties": false
},
"required": ["input_datasets", "output_datasets"],
"additionalProperties": false
},
"additionalProperties": false
},
"required": ["processing_modes"],
"additionalProperties": false
},
"required": ["command_id", "command_return_values"],
"additionalProperties": false
},
{
  "type": "object",
  "properties": {
    "command_id": {
      "const": "PROCESS_DATASETS"
    },
    "command_return_values": {
      "type": "object",
      "properties": {},
      "required": [],
      "additionalProperties": false
    }
  },
  "required": ["command_id", "command_return_values"],
  "additionalProperties": false
},
{
  "type": "object",
  "properties": {
    "command_id": {
      "const": "GENERATE_EMPTY_DATASETS"
    },
    "command_return_values": {
      "type": "object",
      "properties": {},
      "required": [],
      "additionalProperties": false
    }
  },
  "required": ["command_id", "command_return_values"],
  "additionalProperties": false
}
]
}
```

```
    }  
  },  
  "required": [  
    "commands",  
    "response_format_version",  
    "source_rcs_id",  
    "time_of_generation_utc"  
  ],  
  "additionalProperties": false  
}
```

8 ACRONYMS

HF	High Frequency
IM	Instrument Manager
JSON	JavaScript Object Notation (generic file format)
JUICE	JUpiter Icy moons Explorer
IRF-U, IRFU	Institutet för rymdfysik (Swedish Institute of Space Physics), Uppsala department, Sweden
LF	Low Frequency
LP	Langmuir Probe
MM	Mutual IMpedance Experiment (MIME)
PDS4	Planetary Data System 4
PI	Principal Investigator
RCS	RPWI Calibration Software
RPWI	Radio & Plasma Wave Investigation
SCM	Search Coil Magnetometer
TBD	To Be Defined