RPW Operation Centre

ROC Engineering Guidelines For External Users

ROC-GEN-SYS-NTT-00019-LES
Iss.02, Rev.01

Prepared by:	Function:	Signature:	Date
Xavier Bonnin	RPW Ground Segment Project Manager		Dd/mm/yyyy
Verified by:	**Function:**	**Signature:**	**Date**
RPW RCS teams	Team Member #2		Dd/mm/yyyy
Approved by:	**Function:**	**Signature:**	**Date**
N/A	N/A		Dd/mm/yyyy
For application:	**Function:**	**Signature:**	**Date**
Name	Team Member #4		Dd/mm/yyyy

CLASSIFICATION **PUBLIC** ☒ **RESTRICTED** ☐

l'Observatoire de Paris LESIA
Laboratoire d'Études Spatiales et d'Instrumentation en Astrophysique

CNRS-Observatoire de PARIS
Section de MEUDON – LESIA
5, place Jules Janssen
92195 Meudon Cedex – France

Change Record

Issue	Rev.	Date	Authors	Modifications
01	00	08/10/2015	X.Bonnin	First issue
01	01	18/11/2015	X.Bonnin	Update the RCS general conventions Modify the Acronym list
02	00	17/11/2017	X.Bonnin	Second issue with major changes.
02	01		X.Bonnin	Section 2.2: • Rename to “Usage convention for RCS-related tools and services at ROC” • Remove sections “RCS descriptor file”, “RCS naming”, “RCS versioning”, “RCS identification” and “RCS input/output data identification” • Move sub-sections “RCS Git repository”, “ROC DataPool Git repository” from section 2.3 to section 2.2 • Add sub-sections “ROC development server”, “ROC data file system”, “ROC issue tracker tool” and “RCS-related information tools” Section 2.3: • Rename to “RCS testing, delivery and acceptance” • Add sub-sections “Provision of test data”, “Delivery of RCS CDF skeletons”, “Delivery of RPW calibration tables” Section 2.4: • Rename to “Deployment at ROC” • Remove sections 2.4.1, 2.4.2, 2.4.3, 2.4.4 and 2.4.5 Section 2.5: • Rename to “Maintenance of the RCS” • Remove sub-sections 2.5.1 and 2.5.2 Remove section 2.6

Acronym List

Acronym	Definition
ANT	Antenna
BASH	Bourne-Again SHell
BIA	BIAS unit
CDF	Common Data Format
ESA	European Space Agency
ESAC	European Space Astronomy Centre
ESOC	European Space Operation Centre
GIGL	Groupe Informatique Générale du LESIA
HFR	High Frequency Receiver
ID	Identifier
ISTP	International Solar Terrestrial Physics
LDPA	Lightweight Directory Access Protocol
LESIA	Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique
LFR	Low Frequency Receiver
NFS	Network File System
OS	Operating System
RCS	RPW Calibration Software
ROC	RPW Operation Centre
RODP	ROC Operations and Data Pipeline
RPW	Radio and Plasma Waves experiment
RSS	ROC Software System
SCM	Search-Coil Magnetometer
SSH	Secure SHell
SVN	SubVersioN
S/W	Software
TC	Telecommand
TDS	Time Domain Sampler
TM	Telemetry

TNR	Thermal Noise Receiver
URL	Uniform Resource Locator
VCS	Version Control System

Table of Contents

List of figures

List of figures

1 GENERAL

1.1 Scope of the Document

The present document addresses guidelines for the external users involved in the engineering activities of the RPW Operation Centre (ROC) [RD1].

In the framework of this document, the definition of external users covers the people that do not belong to the ROC engineering team in the LESIA site (Meudon, France). Especially, it includes the teams in charge of delivering to the ROC the RPW Calibration Software (RCS).

These guidelines are an extension of the ROC Engineering Guidelines (REG) [RD10], which concerns more specifically the ROC teams of the LESIA.

1.2 Applicable Documents

This document responds to the requirements of the documents listed in the following table:

Mark	Reference/Iss/Rev	Title of the document	Authors	Date
AD1				
AD2				

1.3 Reference Documents

This document is based on the documents listed in the following table:

Mark	Reference/Iss/Rev	Title of the document	Authors	Date
RD1	ROC-GEN-SYS-PLN-00002-LES/1/4	ROC Concept and Implementation Requirement Document (CIRD)	Y.de Conchy, X.Bonnin	17/11/2017
RD2	SOLO-RPWSY-IF-55-CNES_0401(=EID-B).pdf/04/01	Experiment Interface Document Part B (EID-B) for RPW	RPW Team	21/12/2012
RD3	SOL.EST.RCD.0050/05/00	Experiment Interface Document Part A (EID-A)	Solar Orbiter	03/08/2012
RD4	ROC-PRO-DAT-NTT-00006-LES/1/1	RPW Data Products (RDP)	X.Bonnin	17/11/2017
RD5	ROC-TST-GSE-SPC-00017/2/1	Data format and metadata definitions for the ROC-SGSE data	X.Bonnin	14/10/2016
RD6	ROC-PRO-PIP-ICD-00037-LES/01/01	RPW Calibration Software ICD (RCSICD)	M.Duarte, X.Bonnin	17/11/2017
RD7	ROC-GEN-SYS-PLN-00015-LES/02/02	ROC Software Development Plan	X.Bonnin	17/11/2017
RD8	http://nvie.com/posts/a-successful-git-	A Successful Git Branching Model	Vincent Driessen	05/01/2010

	branching-model/			
RD9	https://git-scm.com/	Git	Git developer community	30/05/2017
RD10	ROC-GEN-SYS-NTT-00008-LES/1/3	ROC Engineering Guidelines (REG)	X.Bonnin	17/11/2017
RD11	https://pypi.python.org/pypi/maser4py	maser4py: Python 3 module for the MASER portal	X.Bonnin	20/03/2017
RD12	cdf364ug.pdf	CDF User's Guide	SPDF-GSFC	20/03/2017
RD13	ROC-TST-SFT-SUM-00027-LES/1/1	ROC-SGSE calibration software validation tool user manual	M.Duarte	06/05/2016
RD14	https://about.gitlab.com/	Gitlab	Gitlab team	10/2017
RD15	TBD	ROC Software User Manual	ROC developer team	
RD16	https://pypi.python.org/pypi/maser4py	Maser4py library	Maser developer team	10/2017

2 GUIDELINES FOR THE RPW CALIBRATION SOFTWARE (RCS)

2.1 Context

The RCS play a key role in the processing of the RPW science data. They are delivered to the ROC in order to produce calibrated RPW science data files at the LESIA site. Especially, they must contain a dedicated interface that allows the ROC data processing pipelines to call them in a standard way.

The general convention and the way the RCS must be delivered are presented in the next sections. The interface between the RCS and the ROC pipelines is described in the RCS Interface Control Document (RCSICD) [RD6]

2.2 Usage convention for RCS-related tools and services at ROC

2.2.1 RCS Git repository

The ROC team uses Git [RD9] as a Version Control System (VCS) to store its S/W source codes on a dedicated Gitlab [RD14] server at the LESIA.

This system will be also used in order to ensure the delivery and archiving of the RCS instances run at the LESIA. Especially, each RCS team will have its own space - called "project" in Gitlab – on the ROC server to deliver and host its S/W.

The RCS project page, which includes a Git repository, must be created by a ROC system administrator only, inside the dedicated "ROC/RCS" Gitlab sub-group, and must be named as the RCS.

The URL of a given RCS project page on the ROC Gitlab server will hence look like:

https://gitlab.obspm.fr/ROC/RCS/[RCS_NAME]

Where [RCS_NAME] is the name of the RCS, namely:

- "THR_CALBAR", for THR RCS
- "TDS_CALBA", for TDS RCS
- "LFR_CALBUT", for LFR RCS
- "SCMCAL", for SCM RCS
- "BICAS", for Bias unit RCS

According to the role definition in Gitlab: the ROC system administrators have the "master" role permissions on the RCS repository, people of the RCS team have the "developer" role permissions and people from other RCS teams the "observer" role permissions.

The teams are free to use or not this project page to develop and test their RCS, however the following rules must be applied:

- **The "master" branch of the Git repository must only store versions of the S/W to be delivered to the ROC**. In another word, the "master" branch must not be used to commit development and/or test versions.
- **Any S/W version committed on the "master" branch of the Git repository must be tagged**. The name of the tag must be the version of the S/W release without the

"V" prefix (e.g., "2.1.3"). In the very special case where a modification in the "master" branch is required, but does not need to update the version of the S/W, a fourth integer can be appended as a suffix to the tag name (i.e., "X.Y.Z.k", where "X.Y.Z" is the S/W version number, and "k" is the integer that can be incremented by 1, starting at 1.

- The RCS Git repository must not contain any master CDF, calibration table or testing data. More generally, the repository should not be used to store binary files, except executables and eventually documentations (e.g., .docx).

If teams also use the Git repository to perform the S/W development and testing, the ROC developer team strongly recommends to protect the "master" branch, and to apply the Vincent Driessen's branching model [RD8].

In the case where teams use another VCS, or a different Git server, to store its RCS source code, a specific interface could be set up with the support of the ROC, in order to transfer the S/W into the ROC Gitlab server, with a minimal human intervention.

2.2.2 ROC DataPool Git repository

In addition to the RCS main Git repository, the teams needs an access to the ROC "DataPool" Git repository:

https://gitlab.obspm.fr/ROC/DataPool

This repository is used by the ROC team to archive, among others, the CDF skeleton tables of RPW science datasets.

NOTE: the access to the ROC "DataPool" Git repository is restricted, and the "master" branch is protected (i.e., only the "master" branch owners can modify its content on the Gitlab server). Nevertheless the teams have the possibility to use the "rcs" branch to modify the content of the repository. Be careful since the "rcs" branch is shared between the RCS teams.

To get an access to the "DataPool" Git repository, people has to send an email to the roc.support@sympa.obspm.fr mailing list. The access is also possible using SSH-Key mechanism on demand.

2.2.3 ROC development server

The ROC development server is accessible to the RCS teams on demand. Especially, the RCS teams can use it to:

- Test the compilation/execution of their RCS in the ROC software environment
- Test the RCS compliance with the RCSICD, using the dedicated ROC RCS interface validation tool [RD13]

Besides, they are free to use the allocated space for their own purpose (e.g., developing the RCS), but they must respect the following rules:

- The ROC server policy defined in the section 3.2 must be followed
- There must be one user account by person
- The server is only reachable from the Paris Observatory Intranet, using the SSH mechanism

2.2.4 ROC data file system

Data files received from external collaborators or generated by the ROC are all stored in a dedicated file system. The RCS teams can access this file system from the ROC development server, via an NFS-like mounting mechanism.

Especially, they can deliver their RPW Calibration Tables (RCT) (see section 2.3.2) and RCS test data package (see section 2.3.3) on specific folders.

The table below gives the list of main paths relevant to the RCS in the ROC data file system

Path	Description	Env. variable
`/volumes/plasma/rpw/roc/data/https/private/users/roc_sgse/data/incoming/`	Used to drop incoming files for the ROC_SGSE	ROC_SGSE_DATA_IN_DIR
`/volumes/plasma/rpw/roc/data/https/private/users/roc_rodp/data/incoming`	Used to drop incoming files for the RODP	ROC_RODP_DATA_IN_DIR

Table 1. RCS-relevant paths in the ROC file system.

Following the policy in the section 3.3, the teams can also use the file system to store their own data in the dedicated space.

2.2.5 ROC issue tracker tool

The ROC uses JIRA software as a main issue tracker tool. Especially the RCS teams must submit issues (e.g. tasks, features, bugs, etc.) on the dedicated ROC-DATAPROD JIRA project:

https://jira-lesia.obspm.fr/projects/ROCDATPRO

When creating an issue, the following items must at least provided:

- Summary
- Description
- Type of issue
- Priority
- Assignee
- Components (if any)

Additionally, the following convention shall be applied:

Entity concerned by the issue	Issue summary prefix	Mandatory component(s)
BIAS	[BIA] -	BIAS
LFR	[LFR] -	LFR

SCM	[SCM] -	SCM
TDS	[TDS] -	TDS
THR	[THR] -	THR
ROC	[ROC] -	ROC
All entities	[ALL] -	BIAS, THR, SCM, TDS, LFR, ROC

Table 2. JIRA RCS-related issue conventions.

2.2.6 RCS-related information tools

The ROC and RCS teams can use the dedicated roc.rcs@sympa.obspm.fr to discuss about the RCS.

Additionally, the activity about the RCS can be traced from the ROC Wiki page:

https://confluence-lesia.obspm.fr/display/ROC/RPW+Calibration+Software+Engineering

2.3 RCS testing, delivery and acceptance

2.3.1 Provision of test data

2.3.1.1 Context

The ROC needs to check that the RCS output files produced by the pipelines are as expected. This verification must be done, systematically and in an autonomous way, by the ROC as a part of the acceptance phase, i.e., just after each RCS delivery.

To perform this test, each team must provide to the ROC a so-called “RCS test data package”, which must contain a set of input data files and their corresponding expected output files, for each of the RCS function to test. The input data set should be as much as possible exhaustive and have to be representative enough, in order to be fully confident about the S/W data production behaviour.

Any new RCS delivery must be preceded by the provision of a new test data package; even if the latter does not need any change. This mechanism ensures that a RCS is always delivered with the corresponding test data in mind and that both are always compatible.

2.3.1.2 RCS test data package content description

The RCS test data package must be organized as shown in the figure below.

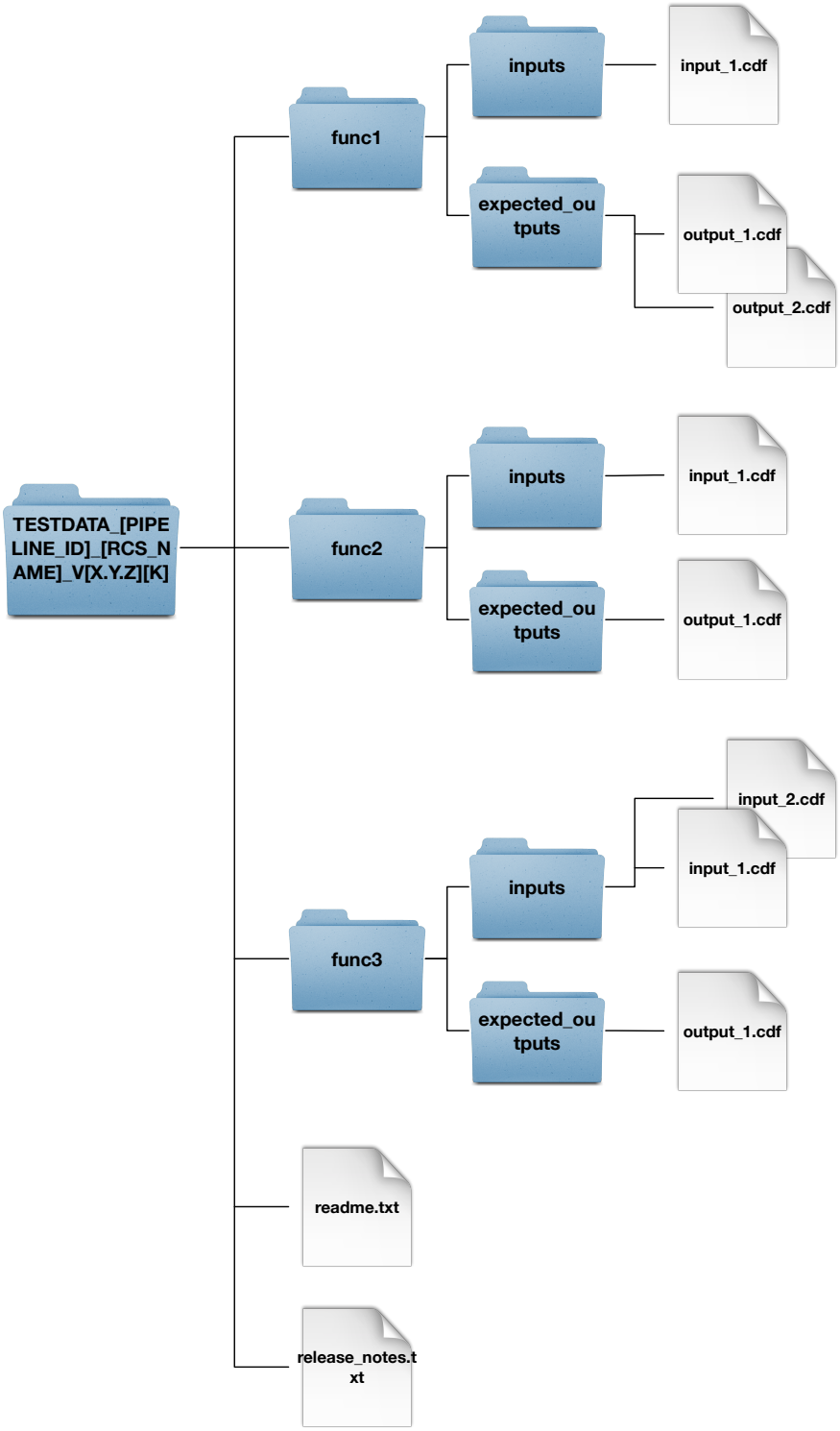

Figure 1. RCS test data package structure tree.

The main directory shall be named as follows:

TESTDATA_[PIPELINE_ID]_[RCS_NAME]_V[X.Y.Z][K]

Where [PIPELINE_ID] is the identifier of the ROC pipeline ("RODP" or "RGTS"), [RCS_NAME] and [X.Y.Z] are respectively the name and version of the RCS, as defined in

the `identification.identifier` and `release.version` attributes of the S/W descriptor file (see RCSICD for details about the related convention).

The [K] field must be a letter indicating the current version of the test data package; in the case where several versions of the package are released for the same RCS version. The first version of the package must have [K]="A", second version [K]="B", third version [K]="C", ...[1]

Inside this directory, there must be one sub-directory per RCS function, i.e., mode, to test. The name of the sub-directory must be the value of the `"modes.name"` attribute in the descriptor file. Each sub-directory must contain the following items:

- An "`inputs`" sub-directory, containing the input RPW data files to be used to run the test for the given function.
- An "`expected_outputs`" sub-directory, containing the expected output RPW data files. These files will be used by the ROC, in order to compare with the files generated by the RCS inside the ROC pipelines.

The name of the test data files - inputs and expected outputs – must follow the convention defined for the corresponding ROC pipelines products: into [RD4] for the RODP and [RD5] for the ROC-SGSE respectively.

Notes:

- A same file can be used to test several functions. In this case, there must be as many as copies of this file than the number of the functions to test (i.e., one copy per function directory)
- When comparing files, the ROC does not compare the meta-data that are not stateless relative to the time and software instance: "Generated_by", "Generation_date", "JOB_ID" and "FILE_UUID".

Additionally, the main directory should also contains:

- A *readme.txt file*, providing at least:
  - A short description of the package content
  - The name, affiliation and contact of the person in charge
  - Any Caveat
- A *release_notes.txt* text file, providing an up-to-date history list of the versions, date/time, author(s) and modifications of the test data package

2.3.1.3 RCS test data package delivery mechanism

The teams must upload their RCS test data package, as a zip archive file, into the dedicated directories in the ROC file system. The list of directories is given in the table below (The actual values of the $ROC_SGSE_DATA_IN_DIR" and "$ROC_RODP_DATA_IN_DIR" environment variables are given in the Table 1). The zip file must have the same name that the test data main directory (see previous section) plus the ".zip" extension. There is one delivery directory per sub-system and per ROC pipeline.

[1] The number of the test data packages, which can be delivered for a given "X.Y.Z" RCS version, is limited to 26.

Test data files used by	Delivery path for the ROC-SGSE-related RCS test data files	Delivery path for the RODP-related RCS test data files
BICAS	$ROC_SGSE_DATA_IN_DIR/BIA/TestData	$ROC_RODP_DATA_IN_DIR/BIA/TestData
LFR_CALBUT	$ROC_SGSE_DATA_IN_DIR /LFR/TestData	$ROC_RODP_DATA_IN_DIR /LFR/TestData
SCMCAL	$ROC_SGSE_DATA_IN_DIR /SCM/TestData	$ROC_RODP_DATA_IN_DIR /SCM/TestData
TDS_CALBA	$ROC_SGSE_DATA_IN_DIR /TDS/TestData	$ROC_RODP_DATA_IN_DIR /TDS/TestData
THR_CALBAR	$ROC_SGSE_DATA_IN_DIR /THR/TestData	$ROC_RODP_DATA_IN_DIR /THR/TestData

Table 3. RCS test data delivery paths.

Notes:

- The ROC must be informed each time a new package has been released. It ensures that the ROC always uses the right package in order to perform the verification tests.
- The delivery directories above are used as a file exchange interface between the ROC and the RCS teams. Especially, the ROC team reserves the right to remove the old test data package if needed.

2.3.1 Delivery of RCS CDF skeletons

The production of the RPW science data files by the ROC pipelines, including the RCS, is performed applying the CDF generation mechanism (see section 4.1 for more details). This mechanism requires using master CDF binary files as templates, in order to generate output CDF files.

The CDF skeleton table files, used to create the master CDF files for the RCS output CDF files, must be delivered to the ROC by the teams in charge.

The teams must upload their CDF skeleton files in the "rcs" branch of the ROC "DataPool" Git repository (see section 2.2.2):

- In the dedicated `ROADS/RODP/CDF/Excel` directory, for the CDF skeletons related to the RODP pipeline.
- In the `GSE/ROC-SGSE/CDF/Excel` directory, for the CDF skeletons related to the ROC-SGSE pipeline.

The CDF skeleton file format must be as required by the "maser4py" Python library [RD16] (i.e., Excel 2007 file format).

Once skeleton files have been uploaded, the teams must use the roc.rcs mailing list, in order to inform other teams. The ROC will then generate the new skeleton tables (.skt) and master CDF binary files (.cdf), and merge the "rcs" branch into the "master". At the end, the ROC will inform all of the teams if the process has ended correctly or not.

Notes:

- The provision and validation of the CDF skeletons must be performed prior to the RCS delivery, in order to ensure that the ROC pipelines always load the right master CDF files when executing the RCS.

2.3.2 Delivery of RPW calibration tables (RCT)

The RPW calibration tables (RCT) are CDF files, needed by the RCS to convert science "engineering" data into calibrated data at receiver and sensor levels. The convention concerning the RCS is defined in the RCSICD.

The teams must upload their RCT files into the ROC file system directories given in the table below (The actual values of the $ROC_SGSE_DATA_IN_DIR" and "$ROC_RODP_DATA_IN_DIR" environment variables are given in the Table 1). Note that there is one directory per sub-system and per ROC pipeline.

RCT CDF files used by	Delivery path for the ROC-SGSE-related RCT files	Delivery path for the RODP-related RCT files
BICAS	$ROC_SGSE_DATA_IN_DIR/BIA/RCT	$ROC_RODP_DATA_IN_DIR/BIA/RCT
LFR_CALBUT	$ROC_SGSE_DATA_IN_DIR /LFR/RCT	$ROC_RODP_DATA_IN_DIR /LFR/RCT
SCMCAL	$ROC_SGSE_DATA_IN_DIR /SCM/RCT	$ROC_RODP_DATA_IN_DIR /SCM/RCT
TDS_CALBA	$ROC_SGSE_DATA_IN_DIR /TDS/RCT	$ROC_RODP_DATA_IN_DIR /TDS/RCT
THR_CALBAR	$ROC_SGSE_DATA_IN_DIR /THR/RCT	$ROC_RODP_DATA_IN_DIR /THR/RCT

Table 4. RCT delivery paths.

Notes:

- As soon as a team has delivered RCT files, it must inform the other teams via the roc.rcs mailing list.
- Using the CDF production mechanism to generate the RPW calibration tables (RCT) is strongly recommended by the ROC team. In this case, the corresponding CDF skeletons must be delivered as specified in the section 2.3.1.
- The delivery directories above are used as a file exchange interface between the ROC and the RCS teams. Especially, the ROC team reserves the right to remove the old RCT files if needed.

2.3.3 Delivery of RCS

The teams in charge must follow the procedures described in this section, each time they plan to deliver a new version of their RCS.

2.3.3.1 Verifications to be done by the teams before delivery

Before proceeding with the delivery of a RCS, the team in charge shall ensure that:

- The RCS delivery package has the expected content (see section below)
- RCS can be compiled and run on the ROC development server
- The interface with the ROC pipelines complies the specification in the RCSICD. For this purpose, the teams can use the dedicated interface validation tool installed in the ROC development server (see [RD13] for more details)
- The required materials: RCS test data package, CDF skeletons and RCT files have been already delivered to the ROC, and are applicable with the new S/W version.

2.3.3.2 Expected content of a RCS delivery package

Any RCS release must be at least delivered with the following items:

- The S/W source code files
- The S/W descriptor file(s), as defined in the RCSICD.
- Any script or executable file used to run S/W on the ROC servers. This executable must comply the specification defined in the RCSICD.
- If required, the S/W configuration file
- Any additional files required to run the S/W without error.
- If required, the scripts to activate/deactivate the execution environment (see RCSICD).
- Context files (e.g., readme.txt, release_notes.txt, etc.), placed at the root of the S/W directory
- If required, the list of additional libraries needed to compile and/or to run S/W. This list can be provided in a specific text file (e.g., requirements.txt for Python) or in the README.rst as default
- If required, any script or program used to compile S/W on the ROC servers (e.g., makefile or ant build file for instance).
- If it exists, any script or executable file that could permit to test the S/W execution (e.g., script to launch unit tests for instance).
- The corresponding up-to-date documentation. Especially a user manual describing in details the S/W in terms of organization, installation and use. This document must be compliant with the ROC documentation conventions defined in [RD10].

The organization of the S/W main directory may look like to the ROC proposal in [RD10].

2.3.3.3 Procedure to deliver a new RCS version

Each time a new version of RCS is available, the team in charge must:

1. Upload (i.e. “push”) the content of this new version in the corresponding RCS Git repository in the ROC Gitlab server. In practice, it should only consist of performing a push on the “master” branch. Make sure that the pushed commit is always tagged with the RCS version; a pushed commit that is not tagged, or badly tagged, will not be accepted by the ROC.

2. Inform the ROC and RCS teams, by sending a message to the roc.rcs mailing list. The object of the mail must contain the S/W name, version and the purpose of the message as a prefix (e.g., "NAME -- VERSION -- NEW RELEASE DELIVERED:").

Note that if the changes in the new version of the RCS have impacts on its data products, the integration of the S/W might also require to upgrade other RCS, using these data as inputs (for instance, the SCM RCS uses L1R waveform data products of the LFR/TDS RCS as inputs). In this case, the procedure might be longer if discussions are needed.

2.3.4 Acceptance

After each RCS delivery, the ROC team will then deploy an instance in the dedicated ROC testing pipeline and launch the verification tests in the following order:

1. Check the compliance of the RCS interface with the RCSICD
2. Run the RCS with the associated test data and compare output files with expected input files. Additionally, the structure and content of the output files will be checked.

If at least one test has failed, then the ROC will then stop the deployment process and inform the team in charge.

2.4 Deployment at ROC

The deployment of a newly delivered RCS (which has passed acceptance testing as described) to the ROC operational infrastructure (replacing any previously existing one) will take place at the discretion of the ROC Development/Operations manager. The teams will be informed when the RCS version running at the ROC changes.

2.5 Maintenance of the RCS

The ROC team plans to monitor the behaviour of its pipelines, including the RCS execution, and to alert the teams in case of anomalies. However, it is outside of its scope to modify the RCS source code or to perform upgrades, e.g., in case of bugs or new calibrations.

It is hence the role of the teams in charge to ensure the maintenance of their S/W, including the compliance with the RCS ICD and the optimization of the RCS science data quality.

3 GUIDELINES FOR THE USAGE OF THE ROC ENGINEERING INFRASTRUCTURE

Depending of the need, a part of the ROC infrastructure is accessible to external collaborators. The accessibility concerns more specifically the collaboration tools, but also servers and data disks.

This section gives the rules to be followed by external users relative to this infrastructure.

All requests/questions concerning these tools must be done using the roc.support@sympa.obspm.fr list.

3.1 Usage of ROC team collaboration tools

3.1.1 ROC Gitlab server

The ROC uses git as a main VCS to store its source codes. The ROC git repositories are managed using a Gitlab server:

https://gitlab.obspm.fr/ROC

The ROC can provide an access to its Gitlab server on demand.

If the demand concerns software to be implemented in the RODP, the Vincent Driessen's branching model [RD8] shall be applied, and only tagged master branch will be used by the ROC.

3.1.2 ROC SVN repository

The ROC can provide an access to its SVN repository on demand. Nevertheless, it must be noticed that the ROC team does not use the SVN repo. for its software development and execution.

3.1.3 JIRA Issue tracker tool

The ROC team uses JIRA as an issue tracker tool. This tool is accessible on demand to external users.

3.1.4 ROC Documentation Manager System

External users can ask for an access to the ROC Documentation Manager System (DMS), which contains the documentation of the ROC.

Especially, any document relative to the RCS must be archived on the ROC DMS.

3.1.5 ROC Wiki page

The ROC project Wiki page is available in:

https://confluence-lesia.obspm.fr/display/ROC/ROC

This Wiki page is run under an Atlassian Confluence server, in order to share information concerning the RPW ground segment project.

Note that the restricted area of the site is currently limited to 25 users.

3.1.6 RPW Web portal

There is no specific rule concerning the RPW Web portal, which is publicly accessible from Internet at https://rpw.lesia.obspm.fr.

3.1.7 ROC intranet site

The ROC intranet Web site is only accessible by the ROC team at LESIA.

3.1.8 ROC mailing lists

External users can use one of the two following mailing lists to communicate inside the ROC project:

- roc.teams@sympa.obspm.fr, must be used for sending messages for all people involved in the RPW ground segment activities.
- roc.lesia@sympa.obspm.fr, must be used for sending messages only to the ROC team at the LESIA.
- roc.cal@sympa.obspm.fr, must be used for sending messages related to the RPW calibrations activities
- roc.sgse@sympa.obspm.fr, must be used for sending messages related to the ROC-SGSE engineering activities

- roc.rcs@sympa.obspm.fr, used for discussions related to the RCS development and integration activities.
- roc.support@sympa.obspm.fr, must be used for sending messages if an assistance is needed when using ROC infrastructure, data products or software.

3.2 Usage of ROC servers

The description of the ROC servers is given in the "ROC Software Development Plan" document [RD7].

3.2.1 Access policy

External collaborators can ask to open a user account on the ROC development server *roc-dev.obspm.fr*.

The ROC development server is only accessible from the Observatoire de Paris intranet and using the SSH protocol only. This requires users to have both a valid LDAP account at the Observatoire de Paris and a user account on the development server.

Note that the users must connect to the *styx.obspm.fr* server first with their LDAP login and password in order to reach the intranet.

Any request concerning an access to the ROC development server must be addressed to the roc.support@sympa.obspm.fr mailing list.

3.2.2 Usage policy

The default user account access privileges are defined by the ROC team, but can be changed if required.

In the same, the space quota per user must be limited to **20 Gigabytes**, but can be also extended on demand. Especially, large data might not be stored directly in the ROC server. They shall be read/saved from/to the dedicated data disks visible on the server (see next section). This requirement avoids exceeding the disk quota on the server, which is dedicated to data processing.

3.3 Usage of ROC data disks

The description of the ROC data disks is given in the "ROC Software Development Plan" document [RD7].

3.3.1 Access policy

The ROC has 16 Terabytes data disk mounted on the ROC servers, and accessible through the "/volumes" local directory.

Each user on the ROC development server has a dedicated space on this disk, reachable at the path:

```
/volumes/plasma/rpw/roc/data/https/private/users/[name_of_user]
```

Where `[name_of_user]` is the name of the user account of the server.

Note that this space can also be visible from Internet via the URL:

https://rpw.lesia.obspm.fr/roc/data/private/users/[name_of_user]

Where [name_of_user] is the name of the user account of the server. The access is restricted and will require to login using the LDAP info.

3.3.2 Usage policy

Each user has the right to read/write inside its dedicated space only. The total amount of data stored in the space must not exceed **50 Gigabytes**. This default configuration can be changed on demand.

4 APPENDIX

4.1 RPW science CDF file production mechanism

All of the RPW science CDF data files generated at LESIA, including the RCS outputs, must be produced using *master* CDF binary files [RD12].

Figure below presents the concept of the CDF file production mechanism.

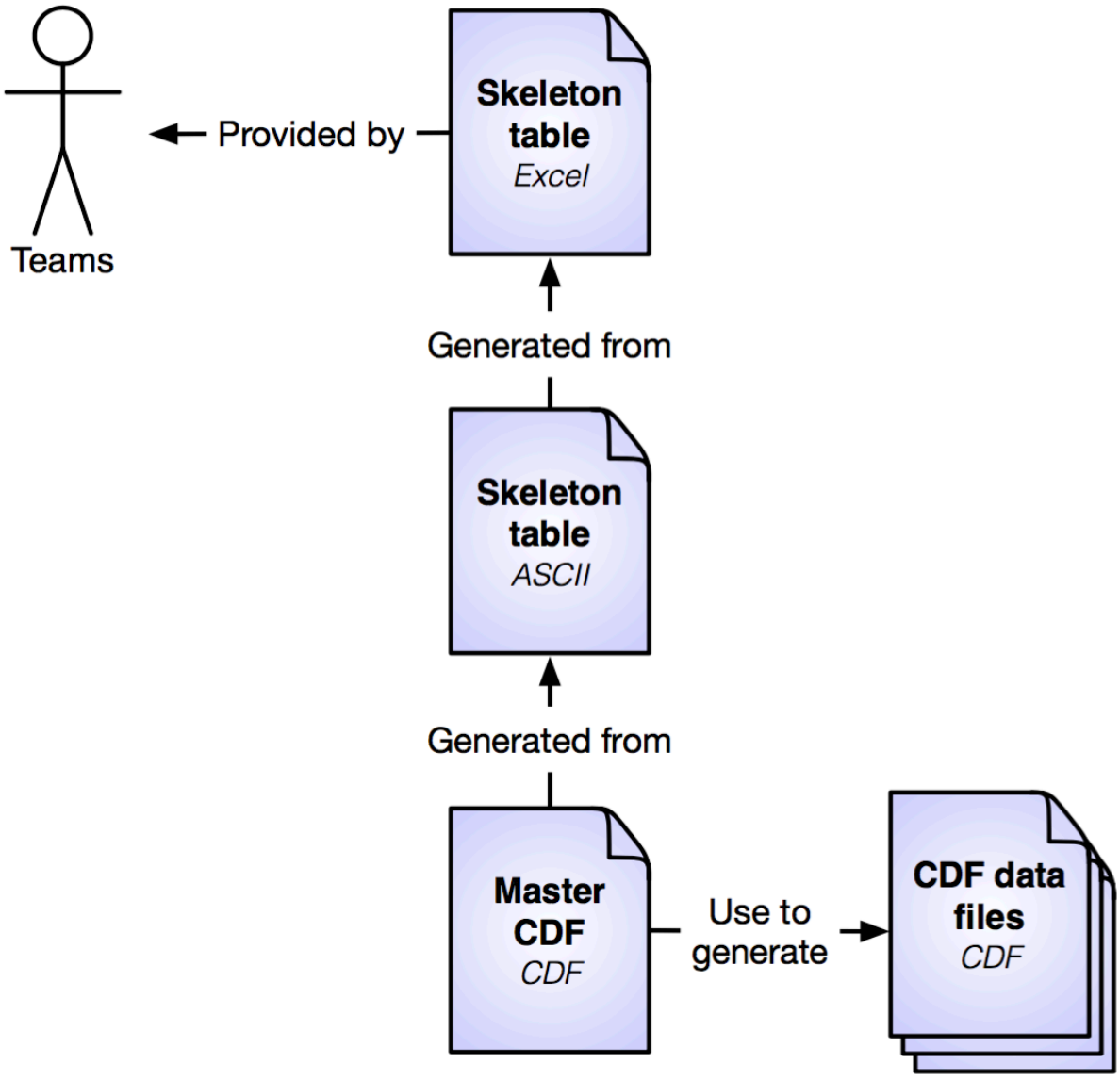

Figure 2. RPW science CDF file production mechanism.

The main steps are:

1. Each RCS team writes the CDF skeleton files for their RPW sub-system data sets. It must be one skeleton file per data set. The format uses to save these files is the Excel 2007 format (i.e., .xlsx).
2. From these skeleton Excel format files, the corresponding CDF skeleton tables in the ASCII format are generated by the ROC, using the maser4py Python package [RD11].
3. The master CDF binary files are then created by ROC, using the "skeletoncdf" tool of the NASA CDF software [RD12]
4. The resulting CDF master files can be then used by the ROC pipelines at LESIA to produce the CDF data files.

A copy of the (Excel, skeleton table, master CDF binary) file set for each RPW science data set is archived in the ROC "DataPool" Git repository. This repository is accessible from teams. Additionally, teams are free to use maser4py to convert master CDF files for their own purposes.

5 LIST OF TBC/TBD/TBWS

TBC/TBD/TBW			
Reference/Page/Location	Description	Type	Status

6 DISTRIBUTION LIST

LISTS See Contents lists in "Baghera Web": Project's informations / Project's actors / RPW_actors.xls and tab with the name of the list or NAMES below		
		Tech_LESIA
		Tech_MEB
		Tech_RPW
		[Lead-]Cols
		Science-Cols

INTERNAL

LESIA CNRS	x	M. MAKSIMOVIC
	x	Y. DE CONCHY
	x	X. BONNIN
	x	QN. NGUYEN
	x	**S. LION**
		L. GUEGUEN
		P. PLASSON
		A. BOUBACAR AMADOU

LESIA CNRS		

EXTERNAL (To modify if necessary)

CNES		C. FIACHETTI
		C. LAFFAYE
		R.LLORCA-CEJUDO
		E.LOURME
		M-O. MARCHE
		E.GUILHEM
		J.PANH
		B.PONTET
IRFU		L. BYLANDER
		C.CULLY
		A.ERIKSSON
		SE.JANSSON
	x	A.VAIVADS
LPC2E		P. FERGEAU
	x	G. JANNET
		T.DUDOK de WIT
	x	M. KRETZSCHMAR
		V. KRASNOSSELSKIKH
SSL		S.BALE

AsI/CSRC		J.BRINEK
		P.HELLINGER
		D.HERCIK
		P.TRAVNICEK
IAP		J.BASE
		J. CHUM
		I. KOLMASOVA
		O.SANTOLIK
	x	J. SOUCEK
	x	L.UHLIR
IWF		G.LAKY
		T.OSWALD
		H. OTTACHER
		H. RUCKER
		M.SAMPL
		M. STELLER
LPP	x	T.CHUST
		A. JEANDET
		P.LEROY
		M.MORLOT
	x	B.KATRA