RPW Operation Centre

ROC Software Development Plan

ROC-GEN-SYS-PLN-00015-LES
Iss.02, Rev.03

Prepared by:	Function:	Signature:	Date
Xavier Bonnin	RPW Ground Segment Project Manager	Signature numérique de Xavier Bonnin DN : cn=Xavier Bonnin, o=LESIA, ou, email=xavier.bonnin@obspm.fr, c=FR Date : 2019.12.19 10:44:51 +01'00'	17/11/2017
Verified by:	**Function:**	**Signature:**	**Date**
Desi Raulin Stéphane Papais	RPW Ground Segment Development Support ROC Software Product Assurance Manager		Dd/mm/yyyy
Approved by:	**Function:**	**Signature:**	**Date**
Name	Team Member #3		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
00	00	23/12/2014	Xavier Bonnin	First draft
00	01	17/02/2015	Xavier Bonnin	Second draft
00	02	25/06/2015	Xavier Bonnin	Third draft
01	00	30/06/2015	Xavier Bonnin	First release
01	01	15/10/2015	Xavier Bonnin	Update the software responsibilities and deliveries
02	01	22/07/2016	Xavier Bonnin	Major updates: Re-organize sections to be consistent with ECSS-E-ST-40C(6March2009).
02	02	27/09/2016	Xavier Bonnin	Complete missing sections
02	03	17/11/2017	Xavier Bonnin	- Add information about continuous integration tools (Jenkins and gitlab) - Describe software environments at LESIA and ESOC - Update software product overview and delivery schedule (with RSS release)

Acronym List

Acronym	Definition
AIT	Assembly, Integration Tests
AIV	Assembly, Integration Validations
API	Application Programming Interface
CDF	Common Data Format
CIRD	Concept and Implementation Requirements Document
CNES	Centre National d'Etudes Spatiales
CoI	Co-Investigator
CP	Cruise Phase
DAS	DPU Application Software
DBS	DPU Boot Software
DPS	Data Processing System
DPU	Data Processing Unit

ECSS	European Cooperation for Space Standardization
EDDS	EGOS Data Dissemination System
E-FECS	Enhanced-Flight Events Communications Skeleton
EM	Engineering Model
ESA	European Space Agency
ESAC	European Space Astronomy Centre
ESOC	European Space Operation Centre
Faust	Flight Operation Request Editor
FCT	Flight Control Team
Figaro	Flight Operation Procedure Editor
FM	Flight Model
FOP	Flight Operation Plan
GIGL	Groupe Informatique Générale du LESIA
GSE	Ground Support Equipment
GUI	Graphical User Interface
HF	High Frequency
HFR	High Frequency Receiver
IAP	Institute of Atmospheric Physics
ICD	Interface Control Document
IDB	Instrument Database
IDL	Interactive Data Language
IOR	Instrument Operation Request
IRF	Institutet för rymdfysik
IT	Instrument Team
I/O	Input/Output
JSON	JavaScript Object Notation
LESIA	Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique
LF	Low Frequency
LFR	Low Frequency Receiver
LL	Low Latency
LPC2E	Laboratoire de Physique et Chimie de l'Environnement et de l'Espace
LPP	Laboratoire de Physique des Plasmas
MADAWG	Modeling And Data Analysis Working Group
MCS	Monitoring and Control System
MDOR	Memory Direct Operation Request

MEB	Main Electronic Box
MIB	Mission Information Base
MOC	Mission Operation Centre
MUsIC	MCS User Interfaces
NECP	Near Earth Commissioning Phase
NP	Nominal Phase
OPera	Operation Planning Interface
ORM	Object Relational Mapping
OS	Operating System
PDOR	Payload Direct Operation Request
PFM	Preliminary Flight Model
PI	Principal Investigator
PMP	Project Management Plan
POPPy	Plugin-Oriented Pipeline for Python
PA	Product Assurance
QA	Quality Assurance
RCS	RPW Calibration Software
RDBMS	Relational Database Management System
REG	ROC Engineering Guidelines
REGU	ROC Engineering Guidelines for External Users
RFP	RPW Flight Procedure
RGS	RPW Ground Segment
ROC	RPW Operation Centre
RODS	ROC Operation and Data System
ROT	RPW Operation Toolkit
RPL	RPW Packet parsing Library
RPW	Radio and Plasma Waves
RSS	ROC Software System
SAP	Science Activity Plan
SBM	Selected Burst Mode
SCM	Search-Coil Magnetometer
SDD	Software Design Document
SDDS	Solar Orbiter EDDS
SDP	Software Development Plan
SGSE	Software Ground Support Equipment

SISSI	SBM Interactive Selection Software Interface
SOC	Science Operation Centre
SolO	Solar Orbiter
SOOP	Solar Orbiter Operation Plan
SOV	System Operations Validation
SRDB	Spacecraft Reference Database
SSD	Software Specification Document
SSH	Secure SHell
SSMM	State Solid Mass Memory
SSS	Software System Specification
SUM	Software User Manual
SVP	System Validation Plan
SVT	System Validation Tests
S/C	Spacecraft
S/W	Software
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TC	Telecommand
TDS	Time Domain Sampler
THR	Thermal Noise and High Frequency Receivers
TM	Telemetry
TMC	TM Corridor
TNR	Thermal Noise Receiver
TV	TM/TC Viewer
URD	User Requirement Document
WF	Waveform
XML	eXtended Markup Language

Table of Contents

List of figures

List of tables

1 GENERAL

1.1 Scope of the Document

This document presents the software development plan (SDP) of the RPW Operation Centre (ROC).

According to [RD26], the purpose of the SDP is "*to describe the established management and development approach for the software items to be defined by a software supplier to set up a software project in accordance with the customer requirements*".

The present document is a tailored version of the SDP as defined above, but covering all of the software units of the ROC software system (RSS).

This SDP addresses more specifically the needs to be foreseen in terms of:

- RSS development, validation and maintenance organisation and responsibilities
- RSS data production, validation and distribution organisation and responsibilities
- RSS-related management and collaboration tools
- RSS-related development approach, environment and integrated logistic support
- RSS-related identified risks and the expected mitigation processes

These needs must meet the requirements initially listed in the ROC Concept and Implementation Requirements Document (CIRD) [AD1] in terms of: technical developments, software tools, data format, and functional facilities to support the ROC science and operational activities. Nevertheless the content of the SDP is expected to follow the evolution of the RSS specification and design.

The SDP must comply with the approach described in the ROC Project Management Plan (PMP) [AD3] and the ROC Software Products Assurance Plan (SPAP) [AD4].

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	ROC-GEN-SYS-PLN-00002-LES/1/4	ROC Concept and Impelement Requirements Document (CIRD)	Yvonne de Conchy, Xavier Bonnin	17/11/2017
AD2	ROC-GEN-OTH-NTT-00045-LES/1/0	ROC Project Glossary of terms	Xavier Bonnin	24/01/2017
AD3	ROC-GEN-MGT-PLN-00026-LES/1/4	ROC Project Management Plan (PMP)	Yvonne de Conchy Xavier Bonnin	17/11/2017
AD4	ROC-GEN-MGT-QAD-00033-LES/1/1	ROC Software Assurance /Product Assurance Plan (SPAP)	Stephane Papais	07/11/2017

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	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
RD2	SOL.EST.RCD.0050/03/00	Experiment Interface Document Part A (EID-A)	Solar Orbiter Team	03/08/2012
RD3	RPW-PRO-DAT-NTT-00006-LES/01/01	RPW Data Products	Xavier Bonnin	17/11/2017
RD4	RPW-TST-GSE-NTT-00017-LES/02/01	Data format and metadata definition for the ROC-SGSE	Xavier Bonnin	14/10/2016
RD5	SOL-SGS-TN-0003/1/2	Solar Orbiter Low-Latency Data: Concept and Implementation	Anik de Groof	19/09/2017
RD6	SOL-SGS-TN-0009/2/2	Metadata Definition for Solar Orbiter Science Data	Solar Orbiter MADA WG	23/07/2015
RD7	SOL-SGS-ICD-0002_DPICD/0/2	Data Producer to archive Interface Control Document	Luis Sanchez	05/09/2014
RD8	SOL-SGS-ICD-0003/1/0	Solar Orbiter Instrument Operation Request Interface Control Document (IOR ICD)	Christopher Watson	13/03/2017
RD9	SOL-SGS-ICD-0004/1/3	Solar Orbiter Interface Control Document for Low Latency Data CDF files	Andrew Walsh	09/02/2017
RD10	SOL-SGS-TN-0006/1/2	SOC Engineering Guidelines for External Users (SEGU)	Richard Carr	03/08/2017
RD11	SOL-ESC-IF-05011/1/0	Solar Orbiter Data Delivery Interface Control Document	Luca Michienzi	10/09/2013
RD12		Deleted		
RD13	SOL-ESC-PL-10001/1/2	Solar Orbiter FOP Preparation Plan	B. Sousa	18/01/2017
RD14	SOL-ESC-TN-12000/1/2	Solar Orbiter Mission Planning Concept (MPC)	Ignacio Tanco	27/06/2014
RD15	SOL-SGS-ICD-0006/1/2	Solar Orbiter Enhanced-Flight Events Communications Skeletons Interface Control Document	Christopher Watson	31/10/2017
RD16	ROC-GEN-SYS-NTT-00008-LES/1/3	ROC Engineering Guidelines (REG)	Xavier Bonnin	17/11/2016

RD17	ROC-GEN-SYS-NTT-00019-LES/2/0	ROC Engineering Guidelines for External Users (REGU)	Xavier Bonnin	17/11/2017
RD18	TN_GSE-017/01/02	RPW MEB SGSE Science Data Format	Loïc Guegen	23/06/2015
RD19	ROC-TST-GSE-SUM-00024-LES/1/1	ROC-SGSE User Manual	M.Duarte	16/12/2016
RD20	SOL-SGS-0006-TS/1/0	Solar Orbiter Instrument Teams – SOC Test Specification	Nana Bach, Chris Watson	30/08/2017
RD21	ROC-PRO-PIP-ICD-00037-LES/1/1	RPW Calibration Software Interface Control Document (RCS ICD)	Manuel Duarte, Xavier Bonnin	12/10/2017
RD22	ROC-TST-GSE-SPC-00004-LES/01/01	ROC-SGSE Software Design Document	Xavier Bonnin	15/04/2016
RD23	https://www.atlassian.com/software/jira	JIRA software	Atlassian team	N/A
RD24	http://nvie.com/posts/a-successful-git-branching-model/	A successful Git branching model	Vincent Driessen	05/01/2010
RD25	SOL-SGS-ICD-0007/1/0	Solar Orbiter Telemetry Corridor Interface Control Document	Christopher J. Watson	14/03/2017
RD26	ECSS-E-ST-40C(6March2009)/3	Space engineering: Software	ECSS team	06/03/2016
RD27	SOL-ESC-RP-05500 - Issue 3r1 20121012 - Solar Orbiter CReMA Issue 3 Rev 1	Solar Orbiter: Consolidated Report on Mission Analysis	José Manuel Sanchez Perez	2012-10-12
RD28	SOLO-RPWSY-PT-1235-CNES/01/00	RPW Instrument Calibration Plan	RPW teams	2014-11-12
RD29	2A- SOL-ESC-HO-05014/1/1	Instrument Command Workshop, ESOC : Commanding Interface and Testing	I.Tanco	05/09/2016
RD30	LL-pipelines at SOC schedule.pptx	LL-Pipelines@SOC Proposed schedule	Chris Watson	06/07/2015
RD31	ROC-TST-GSE-SUM-00035-LES/1/1	Plugin-Oriented Pipeline for Python (POPPy) framework User Manual	Manuel Duarte	26/06/2016
RD32	https://www.djangoproject.com/	Django	Django software foundation	2016
RD33	https://docs.python.org/3/library/venv.html	Creation of virtualenv	Python software	2016

			foundation	
RD34	https://docs.python.org/3/install/	Installing Python modules (Legacy Version)	Python software foundation	2016
RD35	Deleted			
RD36	ROC-GEN-SYS-SPC-00026-LES/01/01	ROC Software System Specification (RSSS)	X.Bonnin	17/11/2017
RD37	ROC-GEN-SYS-SPC-00036-LES/01/01	ROC Software System Design Document (RSSDD)	X.Bonnin, S.Lion	17/11/2017
RD38	SOL-ESC-IF-10002/2/0	Solar Orbiter Instrument FOP Procedure Input Interface Control Document	D. Lakey	12/06/2014
RD39	ROC-GEN-SYS-NTT-00038-LES/01/02	ROC Mission Database Description Document (MDBDD)	X.Bonnin, Sonny Lion	24/11/2017
RD40	RPW-SYS-MEB-GSE-SPC-00125-LES/01/01	RPW MEB GSE Description	Loic Led Guegue n	26/11/2012
RD41	http://maser.lesia.obspm.fr/?lang=en	Measuring Analyzing & Simulating Emissions in Radio frequencies	MASER team	18/07/2017
RD42	https://jenkins.io/	Jenkins	Jenkins developer team	18/07/2017
RD43	SOL-SGS-PL-0009/2/0	Solar Orbiter Archive Plan	Pedro Osuna	01/09/2017
RD44	SOL-ESC-IF-05010/1/2	Planning Interface Control Document	L. Michienzi	07/2015
RD45	https://www.python.org/dev/peps/pep-0008/	PEP 8 -- Style Guide for Python Code	Guido van Rossum, Barry Warsaw, Nick Coghlan	01/08/2013

1.4 About this document

1.4.1 Access policy

The present document is accessible without any restriction.

Any modification of this document must be approved by the RPW Ground Segment Project Manager (GSPM) before publication.

1.4.2 Terminology

All terms used in this document, and which are not listed in the table below must follow the definition in [AD2].

Name	Definition
External team/person/user	Team/person/user that does not belong to the ROC team at LESIA.
(POPPy) Plugin	A software unit designed to be used in the POPPy framework

Table 1. Terminology.

2 ROC SOFTWARE PRODUCTS OVERVIEW

2.1 ROC Software System (RSS)

2.1.1 RSS product tree

The ROC Software System (RSS) definition gathers all of the engineering systems required to reach the ROC functionalities defined in the CIRD. Figure 1 shows the RSS product tree, which is composed of two main systems:

- The ROC Ground Equipment Support (ROC GSE), which regroups the sub-systems in support to tests performed on-ground before and after the launch, namely: the ROC-SGSE and the SBM Algorithms Validation software (SAVs)
- The ROC Operations And Data System (ROADS), which concerns sub-systems relative to the instrument monitoring, commanding and data processing capabilities during the Solar Orbiter mission, namely: the Monitoring and Control Subsystem (MCS) and the Data Processing Subsystem (DPS).

The ROC GSE and ROADS are presented in the next sections. The software environments, i.e., development, testing, validation and execution, as well as the delivery schedule are given in the sections 5 and 6 respectively.

The specification requirements of the RSS can be read in the “ROC Software System Specification” document (RSSS) [RD36], and the RSS design in the “ROC Software System Design Document” (RSSDD) [RD37].

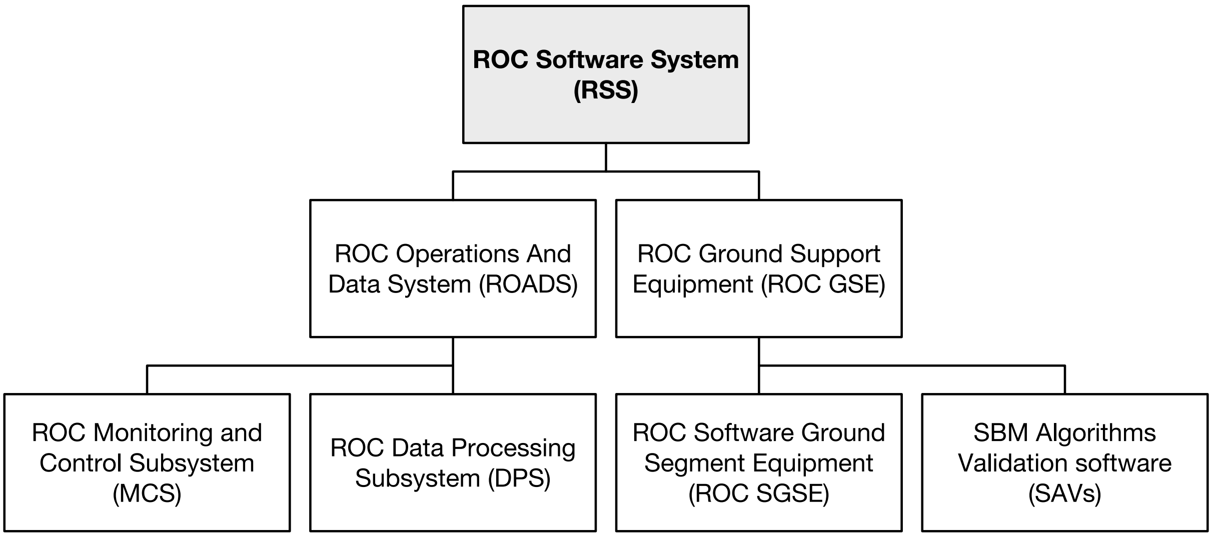

Figure 1. ROC Software System product tree.

2.1.2 ROC Operations And Data System (ROADS)

Figure 2 shows the ROADS software product tree in details.

The ROADS are six main software tools, regrouped into the MCS and DPS sub-systems:

- The MCS User Interface (MUSIC), a Web tool allowing ROC operators to view the mission planning, prepare and submit the operations requests, but also monitoring downlink/uplink TM/TC data flows and analysing incoming RPW data.
- The ROC Operations and Data Pipeline (RODP), the main data processing pipeline to retrieve and process RPW-related data, i.e., TM raw and ancillary data, and operation inputs.
- The RPW Calibration Software (RCS), which actually gathers the five programs in charge of producing RPW science calibrated data. In practice, the RCS are automatically run by the RODP using a specific interface.
- The RPW Low Latency Virtual Machine (LLVM), in charge of processing low latency data for RPW at the Solar Orbiter Science Operation Centre (SOC) site.
- The RPW Data Access Layer (DAL), the infrastructure to give an access to the ROC data.
- The RPW Data Archive (DArc), the infrastructure related to the RPW data archiving.

Besides, the ROADS software tools rely on several databases to work, as explained in the section 2.2.

The main instance of the ROADS, also named "primary" instance, will be hosted in the Laboratoire d'Etudes Spatiales et d'instrumentation en Astrophysique (LESIA) in Meudon, (France), which is the main ROC site. Secondary instances will have also to be deployed and run during the commissioning phase, but at the ESA Space Operations Centre (ESOC) in Darmstadt (Germany), which is the Mission Operation Centre (MOC) for the Solar Orbiter mission.

The next sections give an overview of the MCS and DPS components.

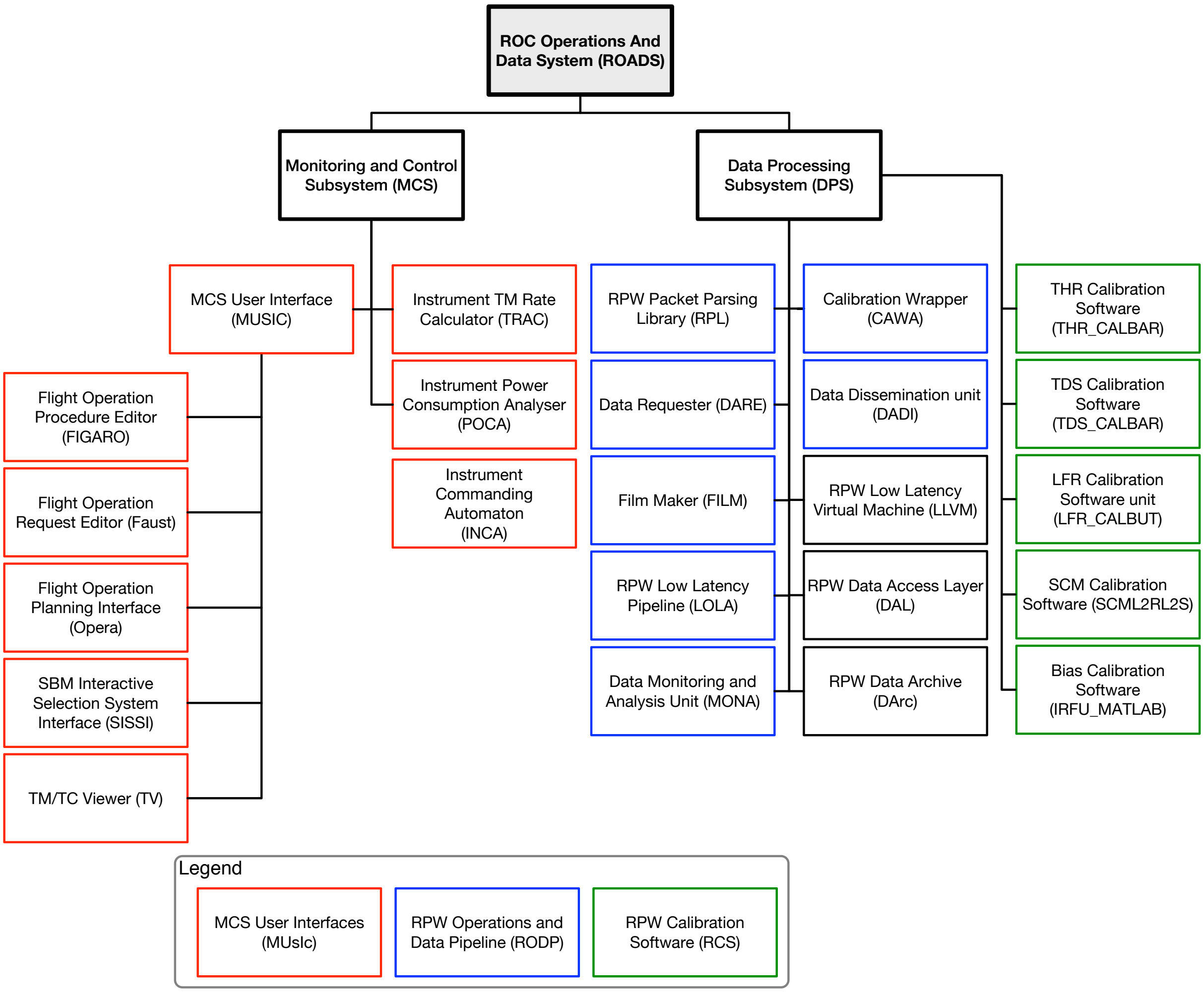

Figure 2. ROC Operations And Data System (ROADS) software products.

2.1.2.1 ROC Data Processing Subsystem (DPS) software units

2.1.2.1.1 The ROC Operations and Data Pipeline (RODP)

The “ROC Operations and Data Pipeline” (RODP) is the main RPW data processing pipeline. It performs all of the automated tasks related to the science/HK/ancillary/operations inputs data retrieval and processing.

The RODP is built with the “Plugin-Oriented Pipeline for Python” framework (POPPy) [RD31], which has a plugin-oriented architecture based on the Python package concept (see section 2.3.1 for more details). POPPy has been initially developed and applied to build the ROC-SGSE tool.

The table below gives the plugins of the RODP and their main functions. The plugins in italic belong to the POPPy core and are not represented in the figure above.

RODP plugin name	Function
DAta REquester (DARE)	Plugin in charge of requesting data delivered by the SOC/MOC interfaces, namely: TM packet/TC catalogue data from the MOC DDS interface, Solar Orbiter ancillary data and operations inputs from the SOC GFTS

	interface.
RPW Packet parsing Library (RPL)	Plugin in charge of parsing (i.e., de-commute/decompress) the RPW TM/TC packets
FILe Maker (FILM)	Plugin in charge of producing the LZ, L0, L1 and HK data files from the retrieved RPW TM packets analysis
Data INGestOr (DINGO)	Plugin in charge of ingesting data into the MDB
RPW CAlibration WrApper (CAWA)	Plugin in charge of executing the RPW Calibration Software (RCS)
RPW LOw LAtency Pipeline (LOLA)	Plugin in charge to run the Low Latency Data Pipeline (LLDP) for RPW.
Data MONitoring and Analysis Unit (MONA)	Plugin in charge of perform automated analysis of the downlinked RPW data (i.e., TC ack. analysis, TM assertion, HK/science data limits verification)
DAta DIssemination unit (DADI)	Plugin responsible of realizing the data dissemination and archiving tasks.
Pipeline UNit Keeper (PUNK)	Plugin in charge of monitoring the pipeline activities and notifying users.
PIpeliner mappER (PIPER)	Plugin in charge of initializing the pipeline and ensuring that static meta-data, required by the pipeline to run, are well defined
Pipeline Operations Planner (POP)	Plugin in charge of coordinating the operations (i.e., jobs) performed by the pipeline

Table 2. RODP plugins.

Furthermore, in order to retrieve RPW-related data, the RODP has dedicated interfaces with the MOC Data Dissemination System (DDS) and the SOC/MOC Generic File Transfer System (GFTS).

Internally, the RODP is directly connected to the ROC Mission Database (MDB) and has a dedicated interface to call the RPW Calibration Software (RCS).

The initial RODP design has inherited the ROC-SGSE architecture.

The list of RPW data produced by the RODP is given in [RD3].

2.1.2.1.2 RPW Calibration Software (RCS)

The RPW Calibration Software (RCS) are in charge of producing RPW science calibrated data files in the CDF format. In practice the RCS are external stand-alone software, which are run inside the ROC pipelines (RODP or ROC-SGSE), using a dedicated plugin named “CAlibration WrApper” (CAWA).

The RCS are developed and maintained by the teams in charge (see table below). It must be delivered to the ROC following specific procedures, described in the ROC Engineering

Guidelines for External Users (REGU) [RD17] and must include a standardized interface [RD21], in order to be called by CAWA[1].

RCS name	Function	Team in charge	Programming Languages
THR data CALiBrAtion SoftwaRe (THR_CALBAR)	Produce TNR-HFR L2/L2S calibrated science data files in the CDF format	TNR-HFR team (LESIA, Meudon)	IDL
TDS data CALiBrAtion Software (TDS_CALBA)	Produce TDS L1R/L2/L2S calibrated science data files in the CDF format. L2/L2S only concern the non-waveform (WF) data products.	TDS team (IAP, Pragues)	IDL
LFR data CALibration UniT (LFR_CALBUT)	Produce LFR L1R/L2/L2S calibrated science data files in the CDF format. L2/L2S only concern the non-WF data products.	LFR team (LPP, Palaiseau)	Python
SCMCAL	Produce TDS/LFR L2/L2S magnetic WF calibrated data files in the CDF format	SCM team (LPC2E, Orléans)	IDL
BIas CAlibration Software (BICAS)	Produce TDS/LFR L2/L2S electrical WF calibrated data files in the CDF format. The BICAS is a part of the IRFU_MATLAB package.	Bias team (IRF, Uppsala)	Matlab

Table 3. RPW Calibration Software (RCS).

2.1.2.1.3 RPW Low Latency Virtual Machine (LLVM)

The RPW Low Latency Virtual Machine (LLVM) is dedicated to process the Low Latency (LL) data for RPW. It contains a stateless version of the RODP, which is run via the dedicated LOLA plugin.

The primary instance of the RPW LLVM will have to run at the SOC site. Especially, the LLVM design, interfaces and delivery process must comply the specification defined by the SOC in the SOC Engineering Guidelines for external Users (SEGU) [RD10].

[1] Since each RCS can be written in different programming languages, the RODP calling interface must be as much as possible common between the software.

In addition, the ROC shall run its own instance of the LLVM at the LESIA. The development, test and execution environments will have to be as close as possible from the SOC one.

2.1.2.1.4 ROC Data Access Layer (DAL)

The RPW Data Access Layer (DAL) definition encompasses all of the interface layers to access RPW data at the ROC site.

2.1.2.1.5 The RPW Data Archive (DArc)

The ROC Data Archive (DArc), which regroups the ROC tools for RPW data archiving at both LESIA and external archive centres (i.e., ESA, CDPP).

2.1.2.2 Monitoring and Control Subsystem (MCS) software units

2.1.2.2.1 The MCS User Interfaces (MUSIC)

The MCS USer InterfaCes (MUSIC) is a Web interface, dedicated to the preparation of the instrument operations and to the instrument data monitoring.

The MUSIC frontend is composed of five tools:

- The RPW TM/TC Viewer (TV), used by ROC operators to promptly visualize the instrument status, TM/TC history and statistics, as well as the HK/science data.
- The RPW Flight Operation Procedure Editor (FIGARO), to create the RPW flight procedures (RFP) in the expected format [RD38].
- The RPW Flight Operation Request Editor (FAUST), to prepare and submit to the SOC/MOC the Instrument Operations Requests (IOR) in the expected format [RD8, RD44], and in accordance with the mission planning constraints.
- The RPW Operation Planning Interface (Opera), to visualize the mission and instrument planning and constraints (i.e., allocation resources) and prepare the operations timeline.
- The SBM Interactive Selection System Interface (SISSI), to manage and select the SBM1/SBM2 event data to downlink.

The MUSIC backend is composed of the following components:

- MUSIC common backend; the main backend of the MUSIC Web tool, which relies on the Django framework architecture [RD32].
- MUSIC_IDB; a module providing a database model to the other MUSIC backend components, in order to access the RPW instrument Database (IDB) in a standard way. The IDB used by the ROADS is stored in the ROC MDB. The database model is the same than for MUSIC (i.e., Django database model).
- Instrument TM RAte Calculator (TRAC); a module dedicated to the TM data rate computation for a given instrument state. Especially, this module serves to compare the instrument states against the Telemetry Corridors (TMC) provided by the SOC.
- Instrument POwer Consumption Analyser (POCA); a module to check the instrument power consumption.
- INstrument Commanding Automaton (INCA); a module in charge of managing the instrument state model (ISM) of MUSIC

The architecture of the MUSIC backend also has an interface with the MDB to retrieve/store related data and meta-data.

2.1.3 ROC Ground Support Equipment (GSE)

Figure 3 shows the product tree concerning the ROC Ground Support Equipment (GSE).

The ROC GSE application has initially concerned instrument tests performed on-ground prior to the launch, namely: the RPW EM2/PFM system calibrations and SBM1/SBM2 detection algorithm validation campaigns. Nevertheless, these GSE tools must be also used in order to pre-validate the RFP before submission, and during the in-flight operations in order to optimize the SBM1/SBM2 detection rates and to support investigations in case of anomalies.

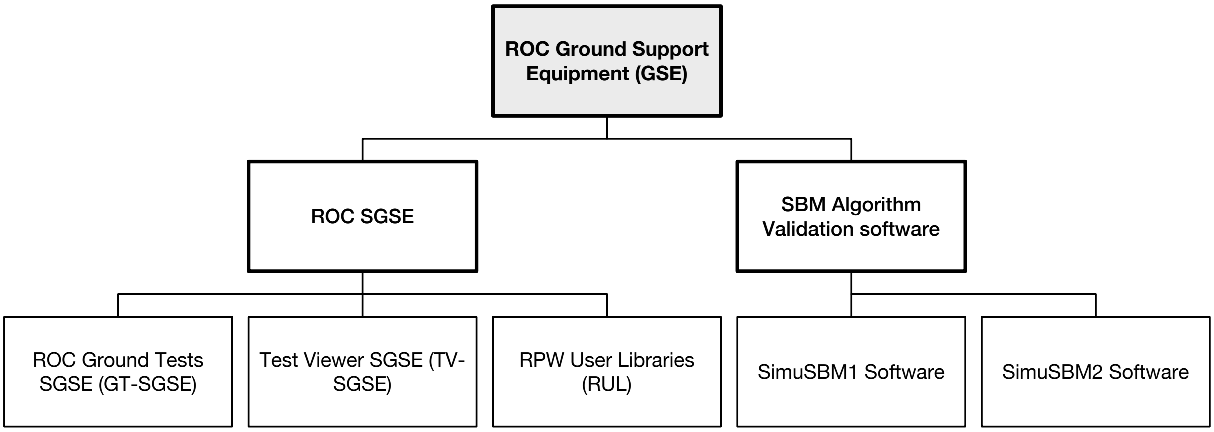

Figure 3. ROC Ground Support Equipment related software products.

2.1.3.1 ROC Software GSE (ROC-SGSE)

The ROC Software Ground Support Equipment (ROC SGSE) was initially developed in support to the RPW AIT/AIV ground calibration activities. Its main function is to retrieve from a MEB GSE database [RD40], the RPW TM/TC packet data generated during a test log session. Then, from the analysis of these TM/TC data, to produce and view L1 science and HK CDF files.

The ROC SGSE is composed of:

- The ROC Ground Tests SGSE (GT-SGSE), a data pipeline to process the RPW data generated during a test and stored into a MEB GSE database. This pipeline is also able to process the stimuli data files generated by the RPW E-GSE system.
- The Test Viewer SGSE (TV-SGSE), an integrated Graphical User Interface (GUI) to load and visualize RPW data retrieved and processed by the GT-SGSE.
- The ROC-SGSE Test Database (TDB), which is used to store information and data related to the tests, as well as pipeline meta-data.
- The RPW User Libraries (RUL), IDL and Python software libraries for people who want to retrieve and analyse ROC SGSE data.

Instances of the ROC-SGSE have firstly been deployed and run during the RPW EM2/PFM system calibration campaigns performed at CNES (Toulouse) and LESIA (Meudon) sites respectively.

Two additional instance of the ROC-SGSE must be also available in support the GSE activities prior and during the mission:

- A first instance will be required to analyse data generated during RFP pre-validation, which will consist of executing RFP - on a RPW "spare" model or S/C simulator - with the MEB GSE tools. Especially, this task will require to install and connect the instance to a dedicated MEB GSE database.
- A second instance will have to be connected to the MEB GSE, which exchanges data with the RPW "spare" model installed at the LESIA site. This instance will serve to analyse RPW data, in the case where anomalies investigations are required on-ground.

The design of the ROC-SGSE is given in the "ROC-SGSE Software Design Document" [RD22].

2.1.3.2 RPW SBM1/SBM2 Algorithm Validation software (SAVs)

The SBM algorithm validation software (SAVs) for RPW is a set of two programs "*SimuSBM1*" and "*SimuSBM2* ", in charge of asserting and validating respectively the SBM1 and SBM2 detection algorithms of the DPU Application Software (DAS).

Both programs can currently perform the following tasks:

- Read synthetized input data and produce CDF format files that comply the specification defined in the REGU.
- Perform SBM event detection and report results into XML format files that comply the specification also defined in REGU.

The SAVs have been initially developed to support the RPW flight software team in the RPW DAS SBM1/SBM2 algorithms validation.

However, these two programs shall be able to read real RPW data generated on-board S/C, in order to analyse and optimize the SBM1/SBM2 event detection rates. In particular, the ROC shall be able to refine on-board algorithms detection parameters, using the results given by the SAVs.

The description of the "SimuSBM1" and "SimuSBM2" programs are given in the "SBM1 Detection Algorithm Simulator" and "SBM2 Detection Algorithm Simulator" documents respectively.

2.2 ROC databases

2.2.1 ROC Mission Database (MDB)

The ROC Mission Database (MDB) is the central base to run and manage the ROADS software. Especially, it permits to:

- Save information and metadata related to the data processing, event reports, instrument operation planning and requests
- Keep a track of the ROADS software units activity (i.e., logs of the jobs status and exceptions)

- Host versions of the RPW Instrument database (IDB), in both the LESIA PALISADE and MOC Mission information base (MiB) formats.

The MDB will be hosted on a dedicated Relational Database Management System (RDBMS) server, maintained by the LESIA computer service (GIGL). It inherits the developments made on the ROC-SGSE Test Database (TDB).

The MDB is described in the "ROC Mission Database Description Document" (MDBDD) [RD39].

2.2.2 RPW Instrument Database (IDB)

The RPW Instrument Database (IDB) contains all of the information to analyse TM/TC packets for RPW, as formatted in the Solar Orbiter Mission Information Database (MiB).

The IDB is developed and maintained by the RPW flight software team.

The ROC Instrument Database is a copy of the IDB hosted in a specific "idb" schema of the MDB.

The ROC IDB can load both the PALISADE XML format version of the IDB, as internally used by the RPW flight software and MEB GSE teams, and the MOC MiB ASCII format version. This duplicity allows the ROC to be able to analyse, at first a step, the TM/TC packets generated from different versions of the IDB, and loaded on both RPW ground as flight models.

2.2.3 Solar Orbiter Mission Information base (MiB)

The MOC will distribute a "reference" Mission Information base (MiB) to the IT. This MiB is made up of the latest Spacecraft Reference Database (SRDB), including payload IDBs, used by the Flight Control Team (FCT) for mission operations.

The ROC shall plan to host a copy of this database, and ensure to use during the mission an IDB from the MOC MiB operational version.

2.2.4 ROC-SGSE Test database (TDB)

The Test database (TDB) is used by the ROC-SGSE to store its configuration meta-data, control its activities and manage the processing of RPW data, retrieved from the MEB GSE database.

2.2.5 ROC MEB GSE database

The ROC MEB GSE database is the MEB GSE database used by the ROC, in order to store its data generated during tests performed on the RPW "spare" model on-ground. This database can be reached:

- By the ROC-SGSE, to generate L1/HK CDF data files
- Via the MEB GSE tools, mainly in order to edit and run C-SGSE scripts or visualize data with the MA-SGSE (see [RD40] for more details about the MEB GSE tools).

2.3 ROC third-party software products

The third-party software products concern all of the software and interfaces developed by the ROC team in parallel to or outside the scope of the RSS.

2.3.1 The Plugin-Oriented Pipeline for Python (POPPy)

The architecture of all of the ROC data processing pipelines (i.e., ROC-SGSE, RODP, RLLP) is based on the "Plugin-Oriented Pipeline for Python" framework (POPPy).

This framework was initially written during the ROC-SGSE development, in order to offer a standard and easy way to build a data processing pipeline with the Python 3 programming language. The initial framework was then used and extended in order to develop the RODP.

A pipeline built with POPPy is composed of a network of inter-connected modules, also called "plugins", to work. Each plugin is typically assigned to one or more well-defined functions (e.g., retrieving data from MOC, producing RPW L1 data files, etc.). To execute a given function, the plugin relies on one or more "tasks", which can be called invidually or as a workflow.

The POPPy architecture design relies on the Python 3 package mechanism, which is very powerful in terms of software installation, configuration and execution.

A full description of this framework can be found in the document [RD35].

2.3.2 RPW user software tools and services

The ROC team should make available helpful routines that can be used by the RPW consortium and other instrument teams, in order to retrieve and handle the data produced by the ROC-SGSE and RODP.

These routines will then constitute the baseline of software libraries for the RPW data user community.

Two libraries are currently developed jointly with the MASER project team [RD41]:

- The "maser4py" Python module (https://github.com/maserlib/maser4py)
- The "maser4idl" IDL package (https://github.com/maserlib/maser4idl)

Both libraries should be freely accessible through the RPW Web portal during the whole mission.

2.3.3 Software in support to the operation and data processing activities

In addition to the ROADS and ROC GSE software products, the ROC shall ensure that tools are developed and run in order to perform the following specific tasks:

- On-board RPW TM and power budget analysis and optimization. This software will be developed by the ROC.
- On-board BIAS current setting from sweeping data. This software should be under the responsibility of the Bias team.
- Effective electrical antenna length and direction determination in-flight. The TNR-HFR team.

A first version of these tools shall be up-and-running at the beginning of the Cruise Phase (CP).

2.3.4 Project management software products

The ROC does not plan to develop any specific software relative to the project management. The centre will only rely on existing commercial software to supervise its activities (see PMP for details).

2.3.5 Solar Orbiter Inter-instrument software products

At this stage of the project, the ROC team does not plan to develop specific software related to inter-instrument collaborations inside Solar Orbiter project.

2.3.6 External collaboration software products

At this stage of the project, the ROC team does not plan to develop specific software related to inter-mission science collaborations (e.g., with FIELDS instrument on Parker Solar Probe).

3 ROC DATA PRODUCTS OVERVIEW

The ROC is in charge of the data processing, dissemination and archiving tasks for RPW. The exhaustive list of instrument data to be produced during the Solar Orbiter mission is reported in the “RPW Data Products” (RDP) document [RD3].

This document must at least supply a short description of each data product, including the format and the context of creation.

3.1 File naming convention, data format and metadata definition

The file naming data format, metadata and processing level definition for the RPW science data must comply the conventions defined at Solar Orbiter level [RD12].

Conventions for the data produced during on-ground calibration tests must be defined separately in [RD4].

Other rules and procedures concerning the data production at the ROC must be listed into the REG document [RD16].

3.2 Data product suppliers and customers

The ROC will have to retrieve, process, generate, validate and deliver several categories of data products during the project. The following table gives a summary of these data, with the format, the suppliers and the main customers.

Data product category	Data format	Data supplier	Data customer(s)
MEB SGSE test log	MEB SGSE test log XML format	RPW MEB GSE team, via the ROC-SGSE MEB GSE database interface	ROC
ROC-SGSE data (HK and uncalibrated science L1 data)	CDF	ROC (generated by the ROC-SGSE from the MEB SGSE test log files)	RPW Consortium, AIT-AIV CNES/LESIA teams
RPW TM packet data	EDDS XML format	MOC (via the EDDS)	ROC
RPW TC catalogue data	EDDS XML format	MOC (via the EDDS)	ROC
RPW TC packet sequences for instrument nominal operation requests	IOR XML format	ROC (via the SOC GFTS)	SOC

RPW TC packet sequences for instrument non-routine operation requests (e.g., contingency, patches)	PDOR/MDOR XML format	ROC (via the MOC GFTS)	MOC
RPW Flight Operation Procedures	MOIS importer format (Microsoft Excel©)	ROC (directly by email to the point of contact at the MOC)	MOC
MEB SGSE GEDEON import script	MEB SGSE GEDEON XML format	ROC (exported by the FAUST tool)	ROC
RPW LZ data	XML	ROC (generated by the RODP from the RPW TM packet data)	ROC
RPW L0 data	HDF5	ROC (generated by the RODP from the LZ data)	ROC, RPW Consortium
RPW L1/L1R/L2 science data	CDF	ROC (generated by the RODP from the L0 data)	RPW Consortium, ESA/CDPP data archives, Science Community
RPW HK-digest parameters	CDF	ROC (generated by the RODP from the L0 data)	RPW Consortium
RPW L3 science data	CDF (TBC)	RPW Lead CoI teams (TBC)	RPW Consortium, ESA/CDPP data archives, Science Community
Solar Orbiter MOC Orbit/attitude/time/frame data	SPICE kernels	SOC (via GFTS)	ROC
Pre-Processed Orbit/attitude/time/frame-digest data for RPW	CDF	SOC	RPW Consortium, ESA/CDPP data archives, Science Community
Quick-looks	PNG	ROC (generated by the RODP)	RPW Consortium, ESA/CDPP data archives, Science Community
E-FECS file	XML	SOC (via the GFTS)	ROC
TMC file	XML	SOC (via the GFTS)	ROC

Table 4. Categories of ROC data products.

4 ROC SOFTWARE MANAGEMENT APPROACH

4.1 Master schedule

The general project master schedule and main milestones are presented in the PMP [AD3].

4.2 Assumptions, dependencies and constraints

4.2.1 Assumptions

At the stage of the project, the ROC assumes that:

- No formal review of the instrument ground segments design will be organised by ESA. Only RFP and interfaces with the MOC and SOC will be validated during the dedicated test campaigns.
- Only the RPW LLVM will have to be delivered to the MOC/SOC.
- With the exception of the LLVM and as long as the expected MOC/SOC interfaces are compliant, the instrument ground segments are free to design its software[2].

4.2.2 Dependencies

The RSS development and execution are dependent of:

- The RPW IDB availability, which is under the responsibility of the RPW flight software manager at the LESIA
- The RCS, which are maintained by the RPW analyser/sensor teams
- The MOC and SOC interfaces availability, hosted and maintained by the ESOC and ESAC respectively.
- The MEB GSE availability, under the responsibility of the RPW MEB GSE manager at LESIA
- The possibility of using RPW engineering model (or S/C simulator) to validate the flight procedures.
- Since most part of the RSS is run on LESIA servers, the availability of these servers and the Internet/Intranet networks.

4.2.3 Constraints

As software supplier the ROC developer team is limited by the following schedule constraints:

- The Solar Orbiter mission schedule and constraints (see PMP)
- The instrument ground segment key points and reviews (see PMP)
- The SOC/MOC interface test schedule [RD20]
- The SOV/SVT campaign planning [RD29]
- The SOC Low Latency (LL) virtual appliance testing and delivery schedule [RD30]
- The RPW instrument calibration plan [RD28]

[2] Nevertheless, the software design still has to comply the high-level and related-specification requirements.

Additionally, the ROC shall take account of the technical constraints:

- The RSS will be hosted at the LESIA site, using the hardware and software resources of the laboratory. Especially, the ROC systems shall comply the LESIA policy in terms of software security and applicability.
- Instances of the ROADS will have to be deployed and run at the MOC site during the commissioning phage. Although the instrument teams might have the possibility to bring their own computer equipment, they will have to ensure that their systems are compliant with the MOC software policies and can be run correctly.

4.3 Software development related risks

The following table attempts to identify the categories of risk that could potentially become points of potential failures for the ROC software development activities. The main consequence is often a delay in the software delivery.

Category of risk	Cause(s)	Consequence(s)	Severity	Probability	Solution(s) to mitigate the risk
Hardware/Operating System (OS) failure at LESIA	Obsolescence, overvoltage	Delay in development, which can cause a delay in the delivery	High	Medium	Plan backup systems to be rapidly deployed at LESIA site (but also during NECP operations at MOC). To mitigate the risk at OS level, the ROC uses virtual machines as primary servers for the RSS. (Hardware/OS recovery is the responsibility of the GIGL)
Obsolete software/hardware	Software updates are not retro-compatible, unavailable hardware devices	Delay in development, which can cause a delay in the delivery	Low	Low	Use as much as possible stable, portable and time-honoured software

					technology
Unexpected personnel reduction in the developer team	Transfer, voluntary redundancy, contract end, disease, pregnancy, ...	Under-sized team, loss of expertise, delay in development	High	Medium	Prompt Non permanent post hiring or internal replacement by the LESIA
Lack of experience	Developer not familiar with a software technology or with quality requirements	Software quality loss, required specification not reached, software delivery delay	Medium	Medium	The ROC shall offer the possibility to its developer team to get professional training and to be followed by a software quality assurance manager

Table 5. Categories of risk related to software development.

4.4 Monitoring and controlling mechanisms

4.4.1 ROC software development "sprint" mechanism

The concept and implementation of the ROC software development "sprints", based on the Agile Scrum method, are described in the PMP.

4.4.2 Software development tools

In addition to the team collaboration tools listed in the PMP, the ROC will rely on the following tools to ensure the software development:

- **Jenkins continuous integration tool**, to control the S/W design and functionalities over the development.
- **Gitlab server**, to store the source codes of the RSS tools (i.e., ROC-SGSE, RODP, MUSIC, LLVM, etc.) and to report related issues, especially the following repositories shall be at least found:
    - **AdminTools**, to store programs and data to administrate the RSS
    - **DataPool**, to store data templates (i.e., CDF skeletons, XSD, etc.)
    - **OpsLib**, to archive operation files (i.e., IOR/MDOR/PDOR, flight procedures and ROC C-SGSE XML scripts)
    - **OpsLib-test**, same than OpsLib but for testing.
    - **RocDocs**, to store ROC documentation source files and templates (.docx, latex)
    - **SupportTools**, used to store additional programs in support to ROC activities
    - **TestTools**, to store software tools in support to the RSS test and validation.

In addition, the Gitlab server will be also used to deliver to the ROC and archive at LESIA the RCS files.

4.4.3 ROC facilities access management file

Information about the user access privileges to the ROC facilities, servers, services and disks shall be reported into a dedicated Excel© file.

This document shall be under the responsibility of the RPW GSPM Manager and must be always up-to-date.

4.4.1 ROC data product management approach

4.4.1.1 ROC data set listing files

The ROC team shall write and maintain an up-to-date list of data to be produced by the RODP at the LESIA during the whole mission. Each line of this list shall correspond to an uniquely identified data set. The list of fields to be provided for each data set is given in the section 4.4.1.2. Besides, each data set shall be associated with a given template, e.g., skeleton tables for the CDF files, XSD for the XML files.

Both the ROC-SGSE and RODP will have its own data sets. The related convention for the RODP and ROC-SGSE must be respectively defined in [RD3] and [RD4].

4.4.1.2 ROC data set listing file description

The following table gives the list of columns to be found in the ROC data set files.

Column name	Description	Priority
Dataset ID	Identifier of the dataset in the ROC system. It shall be unique.	Mandatory
Dataset name	Full name of the dataset	Mandatory
Dataset description	Description of the dataset	Mandatory
Data processing level	Processing level of the dataset	Mandatory
Data file format	File format of the dataset	Mandatory
Data file periodicity	Periodicity of production of the data file for the current dataset (e.g., daily, monthly, per event, etc.)	Mandatory
Data validation leader	Entity (person, institute or team) in charge of the dataset validation	Mandatory
Data validation contributor(s)	Entities (person, institute or team) that contribute to the data validation.	Optional
Data definition schema file	Name of file containing the dataset definition schema (e.g., CDF skeleton, XSD, etc.)	Optional
Parent dataset IDs	ID(s) of the parent dataset(s) required to produce the current dataset.	Optional
Calibration table file(s)	Name of the file(s) containing the table to calibrate the data of the current dataset.	Optional
Software name	Name of the software used to produce the current dataset	Mandatory
Software development/maintenance leader	Entity (person, institute or team) in charge of developing and maintaining the software.	Mandatory

Software development/maintenance contributor(s)	Entities (person, institute or team) that contribute to the software development and maintenance.	Optional
Software application leader	Entity (person, institute or team) which is effectively run the software.	Mandatory

Table 6. ROC dataset listing file columns.

4.5 Staffing plan

The ROC staff is presented in the PMP.

4.6 Software procurement process

ROC software development, test and application environments shall essentially rely on free software projects (e.g., Python, Linux, etc.) to work.

For commercial software such as IDLⒸ and MatlabⒸ programming languages, it has been decided with the GIGL agreement that the ROC developer teams can use the licences supplied by the LESIA.

Software licences shall be reported into the ROC Software Reuse File (SRF).

Additionally, Personal Computers (PC) - or equivalent Virtual Machine (VM) - operating with Windows 7Ⓒ is needed in order to run the MEB GSE tools.

4.7 Supplier management

4.7.1 RCS specific development plan

4.7.1.1 Convention and procedures

In the case of the RCS, the teams shall follow the specific convention and procedures defined in the REGU.

4.7.1.2 Monitoring and controlling mechanisms

Additionally to the validation tests, regular engineering teleconferences shall be organized by the GSPM to follow the RCS development, validation and delivery, including the software data products (i.e., CDF skeletons) and the interface with the RODP.

During these teleconferences, it shall be discussed:

- The progress of the RCS development, including the L1R/L2 data production and the compliance with the ROC interface. The calibrations must be discussed in other dedicated teleconference cycle.
- The list of ended/to be done tasks prior to the next teleconference
- The delivery planning (S/W source codes, related-files and documentation)

The GSPM shall write and send to the people involved the minutes of meeting of the teleconference, and take up-to-date the list of action-items related to the RCS development activities.

4.7.1.3 RCS implementation philosophy

Since the RCS are developed with different programming languages, the ROC must plan to develop a specific interface to call the RCS from the RODP. This interface must be specified in a dedicated “RPW Calibration Software Interface Control Document” (RCS ICD) [RD21]. The way the RCS must be delivered must be described in the REGU.

4.7.1.4 RCS validation tests schedule

The table below gives an overview of the schedule of the RCS implementation tests. The test specification, expected results and schedule shall be detailed in the ROC Software System Validation Plan (SVP). The procedures related to the RCS tests shall be given in the REGU.

Test name	Description and main objectives	Schedule constraints
RCS “interface compliance” test	Test the RCS interface is compliant with the RCS ICD. This test is performed using the dedicated RCS interface validation tool.	To be completed prior to the ROC software validation campaign.
RCS “E2E” integration test	End-to-end (E2E) test to validate the RCS execution by the RODP. It mainly consists of comparing L1R/L2 data, generated by the RODP during the test, with expected outputs provided by the teams in charge.	To be performed during the ROC software validation campaign.

Table 7. RCS validation tests schedule.

4.7.1.5 RCS software delivery schedule

The main delivery schedule about the RCS is listed in the section 6.2.

4.7.1.6 RCS documentation delivery schedule

Concerning the RCS documentation, the RCS teams shall produce:

- A Software Requirement Specification (SRS) document
- A Software User Manuel (SUM).

First versions of these documents shall be delivered to the ROC according to the following schedule. After, up-to-date versions of the documents will have to be supplied to the ROC when required. Especially, any new software release shall yield to a new SUM.

Document(s)	Version	Schedule constraints
SRS	Draft version	To be delivered with the RCS “preliminary” datapackage
SRS / SUM	First issue / Draft version	To be delivered with the RCS “ready-for-flight” datapackage
SUM	First issue	To be delivered with the RCS “fully operational” datapackage

Table 8. RCS documentation delivery schedule.

Templates of SRS and SUM, indicating the expected content, will have to be provided by the ROC.

5 ROC SOFTWARE DEVELOPMENT APPROACH

5.1 Strategy to the software development

5.1.1 General convention and philosophy

5.1.1.1 ROC software development, test and validation philosophy

5.1.1.1.1 RSS development at LESIA

Except the RCS, all of the software relative the RSS will be developed by the ROC on a dedicated *development* environment, including: a development server, specific software environment (e.g., dedicated virtual environments for the software running under Python, specific branches of the Git repository), specific databases and file systems to store input/output files for development purpose only. The *developmennt* environment shall be as much as possible similar, but not necessary the same, than the *production* environment used in operations.

The rules and procedures to be applied when developing ROC software must be detailed in the REG.

In addition, the ROC developer team will rely on a continuous integration tool (see section 4.4.2) in order to avoid regressions in the codes during the development. This tool will verify - after every modification by developers - that the software works correctly, by launching all of the unit tests found inside the code (see the SVP for more details).

5.1.1.1.2 RSS test and validation at LESIA

The RSS software will be tested and validated by the ROC on a dedicated *preproduction* environment, including: a preproduction server (in practice the preprod. server will be the same than the prod. server), specific software environment (e.g., dedicated virtual environments for the software running under Python, specific branches of the Git repository), specific databases and file systems to store input/output files for testing purpose only. The *preproduction* environment shall be as much as possible similar to the *production* environment used in operations.

The rules and procedures to be applied when testing and validating ROC software must be detailed in the REG.

In particular, the *preproduction* environment will be that a software validation campaign will be also planned, prior to the ROADS “ready-for-flight” version delivery. It will allow the ROC to check that the required functionalities and performances of the RSS are reached before the launch. The specification, organization and the expected results of this campaign must be reported into the SVP.

5.1.1.1.3 RCS development, test and validation

Concerning the RCS, the teams in charge must develop and test the S/W for their own sub-systems. Prior to the delivery, the teams must ensure that the software stand-alone execution works correctly on the ROC software environment, and that the interface is compliant with the RCS ICD. Especially, the ROC team will not perform the S/W compilation and configuration on both its development and production servers.

5.1.1.1.4 RPW LLVM development, test and validation

As described in the section 2.1.2.1.3,the RPW LLVM is a VM where a stateless version of the RODP is installed.

The LLVM validation is hence performed in two steps:

- The verification of the execution of the RODP, via the LOLA plugin, is done in the framework of the RODP validation test
- The VM it-self includes specific validation processes, which have to be compliant with the SOC expectations [RD10].

Note that concerning the RPW LLVM, the final validation is always performed by the SOC.

5.1.1.1.5 RSS development, test and validation at MOC site

Instances of the RSS will have to be deployed and run at the MOC site, to analyses RPW data during NECP.

Details about how these instances will be tested and validated are not known at this stage of the project.

5.1.1.2 ROC software execution and maintenance philosophy

All of the software relative to the RSS, including the RCS, shall be hosted and run at the LESIA only. It required the RCS to be delivered to the ROC by teams in charge, following the procedures defined in the REGU. Nevertheless, the primary instance of the LLVM for RPW will be run at the SOC. And the ROC team shall plan to deploy secondary instance(s) of the ROADS at the MOC site during the RPW related NECP operations.

The RSS deployed at LESIA will have to run on dedicated production servers: one for the backend (i.e., ROC pipelines, RODP and ROC-SGSE) and one for the user interfaces (i.e., MUSIC Web tool). There must be one environment per instance (e.g., one virtual environment, one database instance, etc.).

The monitoring of the RSS execution must be done using dedicated tools. In addition, software, including the RCS, will have to generate log files, giving the history of the processes.

Any upgrade of software will have to be done via the ROC software version control system (VCS), as explained in the REG.

Note that the RCS teams can produce data from their local software instance, to support calibration validation and software upgrading for instance, but these data shall be not released publicly without the permission of the RPW PI.

5.1.1.3 ROC software development conventions, procedures and rules

The RPW GSPM shall define general conventions, procedures and rules concerning the ROC software development at LESIA. These conventions shall be listed into the REG.

The REGU document shall address to the sub-system teams the additional rules specific to the RCS development.

In all cases, the RPW GSPM shall ensure the homogeneity and the re-usability of the software development and application environment. Moreover it shall ensure the technologies used are as much as possible sustainable over all of the phases of the project.

5.1.2 Software versioning convention

The software versioning convention for RSS must be defined in the REGU, the convention proper to the LLVM is defined in [RD10]; otherwise the convention to apply for RSS software must be defined in the REG.

5.1.3 Software development configuration

The ROC software developers shall rely on:

- The ROC Gitlab server to regularly save (i.e., commit and push) a copy of their works on the dedicated remote repositories
- The ROC Gitlab server to regularly check progress made by the other developers on a given remote repository
- The ROC Gitlab servers to follow development activities and tasks priorities via the dedicated issues “sprint” dashboards
- The ROC Confluence Wiki page to follow the ROC “sprint” meetings activities and reports.
- The ROC Jenkins interface to check the continuous integration activities, and more particularly the unit tests results.
- The ROC mailing lists to follow the discussions related to the ROC project.

According to this, the ROC GSPM will have to control that the rules and procedures defined in the REG, are well respected by the developer team.

5.1.4 ROC engineering validation plan

The way the whole ROC engineering infrastructure will be validated must be described in the SVP document. It must include the list of tests to be achieved, their purpose, the procedure to follow to perform and validate them, the person in charge and the logistic required for the tests.

A timeline presenting the deadlines of the validation test campaigns, in agreement with the ESA schedule [RD20], must also be provided.

5.1.5 ROC software product assurance plan

The software product assurance plan of the ROC is described in the SPAP.

5.2 ROC software development life cycle

5.2.1 ROC software development life cycle

The RSS software development life cycle is illustrated on the figure below.

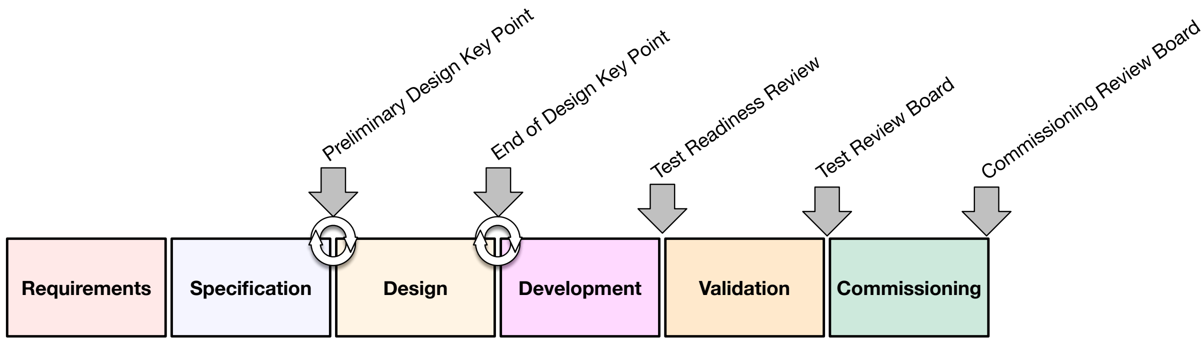

Figure 4. ROC software development life cycle.

The *requirements* and *specification* steps mainly consist of writing the CIRD and the RSSS, according to the requirements and specification at higher level.

The *design* step is dedicated to define the RSS design and to list the associated unit tests to be implemented.

The *development* step will have to be driven with two weeks-spaced "sprint" meetings (see section 4.4.1).

There is no specific testing phase, since the code source is written and continuously integrated - with non-regression verification capabilities offered by the Jenkins tool - during the *development* step.

The *validation* and *commissioning* steps are dedicated to validate the RSS expected functionalities, respectively prior to the launch and after the in-flight instrument-commissioning phase.

The table below gives the list of expected events – reviews, key points, and validation campaigns - for each step of the software development cycle. The list of documents to be delivered before/after each review/key point is given in the SPAP.

Software development step	Expected review/key points/events
Requirements	N/A
Specification	N/A
Design	Preliminary design key point
Development	End of design key point (to take account of changes in the specification and planning)
Validation	ROC software validation campaign kick-off meeting at the beginning of the step / ROC software validation campaign review board at the end.
Commissioning	ROC commissioning review board

Table 9. Expected technical documents during ROC software development life-cycle.

The RSS life-cycle schema is closed to a typical waterfall life cycle. However, since higher-level specification is susceptible to change during the development (mainly due to the SOC/MOC interfaces design and data exchange formats), it shall be flexible enough to introduce possible iterations between *specification*, *design* and *development* steps (white curved arrows on the figure).

The possible software modifications to be introduced in software shall be discussed in the "sprint" meetings every two weeks.

5.3 Software engineering standards and techniques

ROC software is mainly written using Python programming language, for the backend parts, and JavaScript, for the Web frontend parts.

5.3.1 ROC Python-based software engineering standards and techniques

The ROC software written using the Python programming language shall:

- Use Python 3.4 version or higher
- Be developed, tested and run using virtual environments [RD33]. There must be one virtual environment per instance
- Use as much as possible the Python package mechanisms [RD34]
- Follow the PEP8 syntax standard [RD45]

It shall concern at least the following software:

- ROC-SGSE
- RODP
- MUSIC backend, built with Django framework [RD32]

5.3.2 ROC JavaScript-based software engineering standards and techniques

The ROC software written using the JavaScript (JS) programming language shall:

- Use JS ES6 version or higher
- Since there is no well-accepted JS coding standard, follow the syntax convention defined in the REG.

It shall concern at least the following software:

- MUSIC frontend

5.3.3 LLVM specific engineering standards and techniques

The engineering standards and techniques specific to the LLVM can be found in [RD10].

5.3.4 RCS specific engineering standards and techniques

For RCS teams using Python, it is strongly recommended that the same standards and techniques than the ROC are applied to develop their software.

5.4 Software development, testing, validating and execution environments at LESIA

The software environments related to the RSS, which shall be deployed and run at the LESIA site during the mission, are presented in the next sections.

5.4.1 RSS operational infrastructure overview

Figure 5 shows the infrastructure – servers, client and data exchange protocols - to be implemented at the LESIA site, in order to execute the RSS in operation. The infrastructure related to the MEB GSE tools is not presented on the figure.

There are five main components:

- The ROC "production" server, used to deploy and run the ROC pipelines (RODP, ROC-SGSE and RCS). Besides, the pipelines installed on the production server will have to be able to communicate with the DDS server on the MOC site.
- The ROC "Web" server, which stores the backend programs of the MUSIC Web tool
- The ROC "database" server, where all of the instances of the ROC databases are deployed in production
- The ROC "file system" server, where all of the data produced internally by the RSS, or retrieved from outside, are saved. Especially, SFTP and HTTPS servers are also mounted on a dedicated area of this server, in order to exchange data with external collaborators sites (e.g., SOC/MOC GFTS and DDS, RPW consortium).
- MUSIC clients, who are not strictly speaking a server, but gathers all of the MUSIC users computers connecting to the MUSIC Web tools (e.g., through Web browsers).

All of these components will be only accessible from the LESIA intranet network. Additionally, the MOC DDS and MOC/SOC GFTS external interfaces will have to be available too.

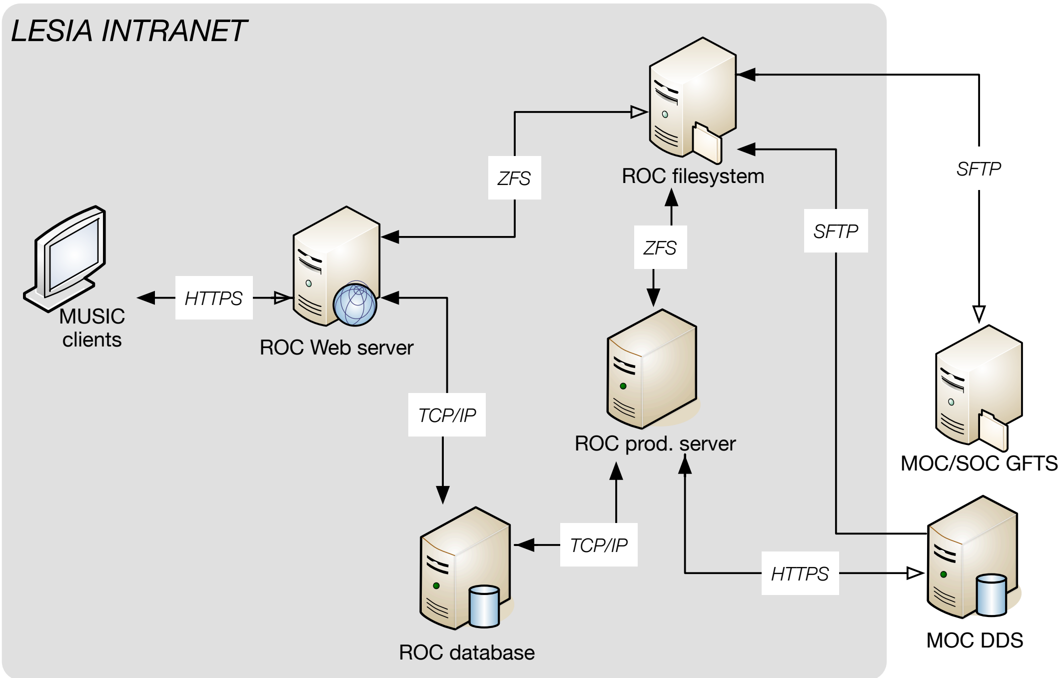

Figure 5. RSS infrastructure overview at LESIA.

The detailed list of each component is given in the sections below.

5.4.1 RSS development, testing and validating infrastructure overview

The ROC developers are free to use their own PCs or the roc development server for writing source codes.

However, since the development and testing environments are highly coupled, it is strongly recommended to use an infrastructure as close as possible from the operational one, and to be able to regularly – at least once a day – pull the newest work in the dedicated remote repository on the ROC Gitlab server. Besides, it will allow Jenkins to run verification tests after a request mergings.

The ROC shall use the operational infrastructure, but with dedicated *testing* and *validating* environments respectively, to test and validate the RSS. In another word, it requires to plan dedicated instances of databases, virtual environments, software input/output data files and corresponding folders, ... The description of the validation facilities must be given in the ROC SVP.

It must be highlighted that the MOC DDS and MOC/SOC GFTS interfaces might be tested and validated independently, and may hence be not components of the testing and validating infrastructure. Nevertheless, the ROC shall plan to simulate data exchange via these interfaces, for specific software tests and during the ROC software validation campaign.

5.4.2 ROC hardware equipment

According to the responsibilities defined in the CIRD, all of the ROC hardware: servers, data disks and interfaces, are maintained and administrated by the GIGL.

5.4.2.1 ROC data storage volume capability

The nominal RPW TM raw data rate on-board is 5.5 kbps, which corresponds to a volume of 59.4 MB per day. Nevertheless, the actual rate might be increased up to three times this value during best data downlink windows.

According to this, an estimation of the full RPW data volume after processing (i.e., HK, L1 and L2 data files production) for a day can be ~2 GB. Additionally, the ROC will have to store L3 and ancillary data (i.e., SPICE kernels), summary plots and operations-related files (i.e., E-FECS, TMC, IOR, procedures etc.), which should occupy around 1 GB per day.

In consequence, the total amount of data is expected to be ~1 TB per year during the mission, so 10 TB over 10 years including the extended phase.

Moreover, the data processing and dissemination steps might require redundancy of data files (e.g., predictive/definitive or private/public data), hence 20 TB over 10 years shall be planned to be confortable enough. Finally, the data from ground tests, i.e., mainly GSE data, as well as personal data stored by the ROC users should require ~10 TB.

The ROC data product files, mainly generated from the ground calibration campaigns with GSE, are currently stored using two 8 TB data disks. However, the total data volume capacity will be extended to 32 TB before the launch.

Additionally, the ROC plans to rely on the DIO data storage facilities at the Paris Observatory, in order to save its long-term data archive. Especially, data archive will have to be regularly - at least every week – saved on magnetic tapes. This facility will have to be up-and-running at the beginning of the cruise phase.

5.4.2.2 ROC database storage volume capability

The ROC databases - MDB, TDB and ROC instances of the MEB GSE database - will be hosted on the LESIA database server. The total ROC databases size estimation is 3 TB over 10 years.

5.4.2.3 ROC servers

Table below provides the name and function of the ROC servers hosted at LESIA.

Server name	Function	Configuration	Access privileges
roc.obspm.fr	The "production server" is used to run the primary instance of the RODP in prod.	• A x86 Intel architecture VM • Inter(R) Xeon(R) 2.4 GHz • Linux Debian Jessie OS. • 871 GBytes (shared memory)	The administration of this server is done by the GIGL. Only accessible to the authorized ROC developers through SSH.
roc-dev.obspm.fr	The "development server" is dedicated to develop, test and validate the RODP, before implementation on the "production" server. Teams to test and validate their RCS, before delivery to the ROC, can also use it. The dev. server also hosted a test instance of MUSIC	• A x86 Intel architecture VM • Inter(R) Xeon(R) 2.4 GHz • Linux Debian Jessie OS. • 871 GBytes (shared memory)	The administration of this server is done by the GIGL. Accessible to the ROC team and RPW consortium people involved in the ground segment activities.
roc-web.obspm.fr	The "web server" is used to host the MUSIC tool	• A x86 Intel architecture VM • Inter(R) Xeon(R) 2.4 GHz • Linux Debian Jessie OS. 871 GBytes (shared memory)	The administration of this server is done by the GIGL. Only accessible to the ROC developer team through SSH.
dam-nancay.obspm.fr	The dam-nancay server is used to host, test and run the RPW LLVM instance at LESIA.	• Same than the EUI LLVM	The administration of this server is done by the GIGL. Only accessible to the ROC LLVM developers through SSH.
roc-nfsvm	VM used by the LLVM to test the NFS mounting. This VM is hosted on damnancay.	• Same than the EUI NFS testing VM	
rocpc1.obspm.fr	This is the server used by the ROC to host and run the MEB GSE tools	• Windows 7	The administration of this server is done by the ROC developer team.
arietis.obspm.fr	The *arietis.obspm.fr* server hosts the RPW		The administration of this server is done

	Web page at the LESIA.		by the GIGL, with specific accesses for the ROC team in order to edit the RPW Web page through the SPIP[3] framework.
lesia11.obspm.fr	The "data storage" server is used to reach the LESIA data disks, which store the ROC data files. It currently provides 16 Terabytes of space. This server is only accessible in read/write privilege from the roc-dev.obspm.fr and roc.obspm.fr via a ZFS mounting system. HTTPS and SFTP servers also runs, giving access of private/public directories.	16 Terabytes (to be extended to 32 before the launch)	The administration of this server is done by the GIGL, with specific accesses for the ROC and sub-system teams.
TBD	The ROC plans to have a dedicated server for its long-term data archive. This server will have to be reachable from the roc.obspm.fr server using NFS mechanism. It is expected that regular backup of data is performed. This server will have to be ready before the launch.	TBD (at least 32 TB over 10 years)	The administration of this server is done by the DIO.
bdd-lesia.obspm.fr	The "LESIA database server" will be used to host the ROC databases.		The administration of this server is done by the GIGL, with specific accesses for the ROC team to its databases.
TBD	One PC connected to at least 2 22'' size screen	• Minimal configuration to display the MUSIC	The maintenance and administration

[3] http://www.spip.net/

	monitors, in order to view the RPW activity during the mission. This equipment shall be available 24h, in the RPW operations room at LESIA.	frontend from a client (i.e., Web browser)	of this equipment will be under the GIGL responsibility. The ROC system administrators shall ensure the MUSIC client is still up-and-running
gitlab.obspm.fr	Used to host the ROC gitlab server		The administration of this server is done by the DIO, with specific accesses for the ROC team to manage its Git repositories. Some repo. must also be accessible to the RPW teams.
confluence-lesia.obspm.fr	Hosts the ROC confluence site.		The administration of the server is done by the GIGL.
jira-lesia.obspm.fr	Hosts the ROC JIRA site.		The administration of the server is done by the GIGL.

Table 10. ROC operating servers.

Note that the development and production servers are not sized to store a large amount of data volume, and shall be used for program testing and execution only.

5.4.3 ROC hardware communication interfaces at the LESIA site

At the LESIA site, it shall include:

- Internal communication interfaces between the ROC servers
- Internal communication interfaces between the ROC servers and data disks
- Internal communication interfaces between the ROC servers and ROC team computers.
- Communication with the Internet network through the dedicated interfaces.

The availability of these facilities is under the GIGL and DIO responsibilities.

5.4.4 Software environment

5.4.4.1 ROC server Operating Systems (OS)

The ROADS servers will run on the same **SMP Debian Jessie x86_64** Operating System (OS).

The default command processor of the system will be the **Bourne-Again SHell** (BASH) V4. In consequence, any S/W executed on the ROC servers shall work within this OS environment.

An additional Windows 7 server is also available to use the MEB GSE tools and database instances of the ROC.

5.4.4.2 ROC LLVM OS

The RPW LLVM is a tailored copy of the LLVM developed by the Solar Orbiter EUI team using the CentOs OS.

5.4.4.3 User account management

The ROC shall people directly involved in the ROC activities to have a dedicated user account on the *roc-dev* development server, as well as a dedicated space on the ROC data disk to store its personal data related to the project.

Convention concerning the server usage, e.g., user access and data storage quota, etc., must be reported by the ROC GSPM into the REGU.

The complete list of user accounts and access privileges shall be reported on the “ROC device access document”.

The access to the ROC development and production servers shall be only possible via the SSH protocol and using LESIA LDAP account mechanism. Especially, the production server shall only be accessible from people labelled as “ROC system administrators”.

5.4.4.4 Technical account management

In addition to the personnel user accounts, the following so-called “technical” accounts shall be created to manage and run the ROC pipelines.

Technical account	Description	Development server	Execution server	Comment
roc_rdop	Technical account for the RPW Data and Operations Pipeline execution	roc-dev.obspm.fr	roc.obspm.fr	Access to this account is restricted to the ROC developer team only
roc_sgse	Technical account for the ROC SGSE execution	roc-dev.obspm.fr	roc.obspm.fr	Access to this account is restricted to the ROC developer team only
roc_rllp	Technical account for the RPW Low Latency Pipeline execution	roc-low.obspm.fr	roc-low.obspm.fr	Access to this account is restricted to the ROC developer team only
rocuser	Technical account used to have a generic read-only access to the Git repositories of the ROC			
roc_git	Technical account		gitlab.obspm.fr and	

	used to synchronized ROC gitlab and JIRA issues		jra.obspm.fr	

Table 11. Technical accounts.

5.4.4.5 Root account management

The OS administration and maintenance of the servers and disks are ensured by the GIGL. However people needing to perform specific operations at the OS level can ask for a root access. These people shall be clearly identified and belong to the ROC system administrator team only. Any request for a root access shall be sent to the ROC GSPM.

5.4.4.6 ROC data access policy

All of the data produced by the ROC will be stored in dedicated data disks (see section 5.4.2.1). Access to the data disks will be possible from the ROC servers using the ZFS mounting system. Specific access privileges must be defined for each user. Some directories in the data disks will be also visible from Internet to allow people to retrieve data.

User access privileges must be reported into the "ROC device access management file".

5.4.4.7 RSS software programming languages

Most of the ROC software is based on the Python 3 programming language, which becomes both a major programming language, and a more and more used language in the astronomy community. The frontend part of the MUSIC Web tool will be developed using Javascript.

The following table gives the list of main ROC software, the programming language name and the main modules to be used for the development.

Software	Programming Languages	Main external modules/libraries
ROC software		
RODP	Python 3 + Cython	numpy, SQLAlchemy
LLVM	Linux BASH, Python 3 + Cython	numpy, SQLAlchemy
MUSIC	Python 3 with Django framework (backend) + Javascript (Frontend)	numpy, matplotlib, react
ROC-SGSE	Python 3 + Cython	numpy, matplolib, SQLAlchemy, PyQt5 (for the TV-SGSE)
SimuSBM1 Software	IDL	
SimuSBM2 Software	IDL	
RCS		
THR_CALBAR	IDL	
TDS_CALBA	IDL	
SCMCAL	IDL	
LFR_CALBUT	Python 3	
BICAS	Matlab 5	

Table 12. RSS software programming languages.

5.4.4.8 Database management systems

The following table presents the databases involved into the ROC activities and the associated management system.

Database	Functions	Management system
ROC-SGSE Test Database (TDB)	Database of the ROC-SGSE	PostgreSQL
ROC Instrument Database (IDB)	ROC instance of the RPW IDB	PostgreSQL
ROC Mission Database (MDB)	Central database of the ROADS	PostgreSQL
MEB SGSE Database	MEB GSE database	MySQL
RPW Instrument Database (IDB)	RPW IDB	XML (LESIA PALISADE format) / PostgreSQL (MOC MiB format)
Mission information Base (MiB)	Database used by the MOC to perform the in-flight operations. It contains a copy of the RPW IDB.	PostgreSQL

Table 13. List of databases involved in the ROC activities.

Note: in practice the ROC IDB will be included as an "idb" schema in the ROC MDB Postgres database.

5.4.4.9 Main software libraries

The following table presents the list of main software libraries that must be used to developed and run the RSS. The full list of software dependencies will have to be reported into the ROC Software Reuse File (SRF), as explained in the SPAP.

Library name	Short description
CDF	NASA CDF library
Cython	C for Python module
matplotlib	Mathematical plotting library for Python
numpy	Scientific Python library
SPICE	NAIF SPICE Toolkit
SQLAlchemy	Database ORM module for Python
django	Python Web framework
react	Javascript library

Table 14. List of RSS main software libraries.

5.4.4.10 Software development and testing support tools

The ROC developer team shall use the virtual environment and the package mechanisms to develop, test and install the software written in Python 3, namely: RODP, ROC-SGSE and MUSIC (backend part) software. These mechanisms are flexible, robust and widely used in the developer community.

Unit tests must also be implemented into the RODP, ROC-SGSE and MUSIC software, in order to validate critical functions. The list of unit tests must be reported into the SVP.

The verification of the modifications in the source codes must rely on the Gitlab and Jenkins tool capabilities. Especially, each merging request on Gitlab shall trigger a run in Jenkins, in order to check that the code still works correctly (i.e., non-regression tests). Especially, software under Python shall use the *pytests*, *tox* and *hypothesis* modules to design and run the verification tests with Jenkins.

The tools in support to the validation must be reported in the SVP. Any specific procedure and/or convention related to the development and testing support tools must be reported by the ROC GSPM into the REG.

5.5 Software development, testing, validating and execution environments at ESOC

This section presents more specifically the software environments, which are related to the RSS instances to be deployed and run at the ESOC site during the NECP.

5.5.1 RSS operational infrastructure overview

During RPW-related NECP operations, the ROC must be able to run specific "secondary" instances of the RSS at the ESOC site.

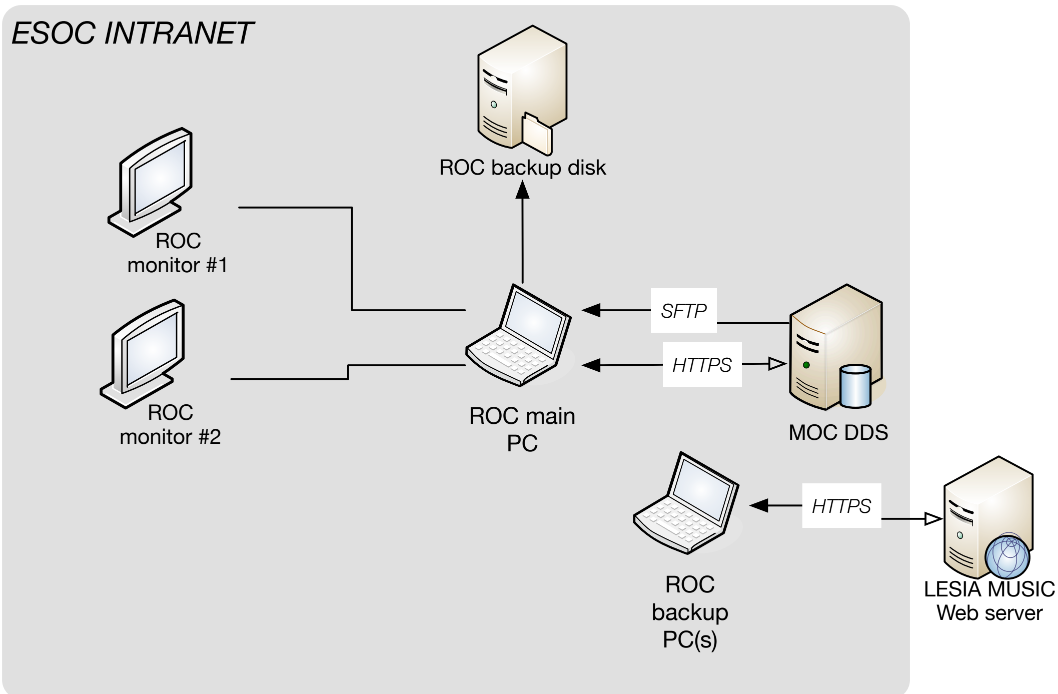

Figure 6. RSS infrastructure at ESOC.

At the stage of the project, the detailed organization of the LEOP/NECP operations at MOC is not known; nevertheless, as shown on the figure above, we can reasonably assume the following scheme:

- One main PC is connected to the MOC DDS in order to retrieve, process and view RPW data flow in quasi-real time using the local RSS instance. In addition, it shall offer the possibility to easily distribute data files locally generated (e.g., L1/HK CDF data files), to other RPW people present on the site. Nevertheless, this action should not be done when the RSS is working (i.e., during critical phases of the operations).
- Extra monitors will be connected to the main PC to visualize data. Two screens in addition to the PC, may be confortable enough.
- The main PC will have to be connected to an external data disk, which shall perform regular backup of the PC content and data.
- Extra PCs, at least two, shall be planned in backup. Especially, the RSS will have to be installed and ready to be used in case of main PC failures. If an Internet connexion is possible, these PCs may also rely on the MUSIC Web tool at LESIA to view the data.

This equipment will have to be brought from the LESIA to the ESOC by the ROC team, and be installed by the latter with the support of the ESOC staff. To ensure as much as possible easy and quick transportation and deployment, the RSS instances will have to be installed and tested in advance (i.e., at LESIA) on laptops, with the typical system performances expected to run the RSS without latency (see RSSS for more details). Especially, they will have the capability to retrieve data from the MOC DDS interface, and to store data generated during the operations on the local hard disk.

In practice, only ROADS sub-system functionalities are required, and thus, the ROC does not expect to use ROC GSE in this case. In parallel, the primary instance might also start to retrieve and process data at LESIA.

5.5.2 RSS development, testing and validating infrastructure overview

The infrastructure for the development of the MOC RSS should be the same than for the LESIA instance. Nevertheless, the ROC GSPM will have to ensure that the technical specification allows users to install and run the RSS on PC laptop with the required system configurations.

At this stage of the project, the MOC team has not communicated information about how the RSS infrastructures can be tested and validated at the ESOC site. Nevertheless, MOC RSS instances will be tested on the dedicated laptops at LESIA, during the validation campaign, as explained in the SVP.

5.5.3 ROC hardware equipment

Table below gives the list of hardware equipment that the ROC team expects to use at ESOC site for LEOP/NECP RPW-related operations, including the main function and the minimum system configuration.

Device name	Function	Configuration	Comment
TBD	Main PC to run the RSS at MOC	Minimal configuration should be: • Processor: 2.3 GHz	

		• RAM: 16 Go • Volume: 1To	
TBD	First backup computer, in case of main computer failure	• At least same config. than the main computer	
TBD	Second backup computer, in case of first backup computer failure	• At least same config. than the main computer	
TBD	External data disk to backup data generated by the MOC RSS	• 2To	The external data disk will be connected to the main PC (or backup in case of failure)
TBD	First external monitor to be connected to the main computer	• At least 22'' size screen	
TBD	Second external monitor to be connected to the main computer	• At least 22'' size screen	

Table 15. Hardware equipment for MOC RSS.

5.5.4 ROC hardware communication interfaces at the ESOC site

Since the RSS will be deployed and run on PCs at ESOC, the latter shall ensure that the ROC equipment can promptly retrieve from the MOC DDS and process the RPW data, at least for the main PC.

5.6 Software documentation plan

5.6.1 General

The software documentation plan presents the management of the ROC engineering documentation.

5.6.2 Software documentation identification

The software documentation identification shall follow the convention defined in the PMP.

5.6.3 Deliverable items

5.6.3.1 ROC deliverable documentation

The documentation concerning the ROC software is given in the PMP.

5.6.3.2 RCS deliverable documentation

Consult the section 4.7.1.5 for more details.

5.6.4 Software documentation standards

The software documentation shall follow the standards specified in the PMP.

Any additional procedures and rules concerning the software documentation shall be listed into the REG.

6 ROC SOFTWARE DELIVERY PLAN

This section presents all of the S/W products to be delivered to the ROC and by whom in preparation of the RPW ground segment activities. The following schedule shall always be consistent with the planning defined at project level in the PMP.

6.1 ROC software development planning

The ROC software development planning must be detailed in the *ROC planning file*, as defined in the PMP. Especially, the planning file must report the milestones relevant to the software development schedule, the tasks inter-dependencies and associated resources.

6.2 ROC software deliveries schedule

This section presents the ROC software deliveries schedule, in agreement with the project planning and constraints defined in the PMP.

6.2.1 RSS data package main releases schedule

The ROC plans to release six main versions of the RSS over the ROC planning. Major milestones in the activities at the project level motivate these releases. Especially, the ROC shall ensure the main functionalities are available to ensure the required tasks. The RSS functionalities can be read in the RSSS.

Note the RSS validation campaign should take place at least one month before the "ready-for-flight" version release (RSS4), in order to let the time to perform the Test Review Board (TRB), as defined in the SPAP.

The ground segment project manager shall ensure that the RSS validation campaign schedule, provided in the SVP, is consistent with the current schedule.

RSS release ID	RSS data package release name	Main required software units and related functionalities	Person in charge of the release	Release related deadline
RSS0	RSS "preliminary" version data package	- SBM1/SBM2 Algorithm Validation Software V1, ready for the ground DPU SBM detection algorithms validation campaign - RPW Packet parsing library (RPL) preliminary version (compatible with the RPW IDB 2.2.3)	X.Bonnin (ROC)	Beginning of the DPU SBM detection algorithms ground validation campaign
RSS1	RSS "EM2 calibrations" version data package	- Preliminary ROC-SGSE with functionalities ready to process and visualize the RPW for the EM2 blank calibration tests at CNES (Toulouse) - "Hello world" version of the RPW LLVM	X.Bonnin (ROC)	Beginning of the RPW EM2 blank calibration tests

RSS2	RSS "PFM calibrations" version data package	- ROC-SGSE with functionalities ready to process and visualize the RPW and E-GSE stimuli data for the PFM thermal calibrations at LESIA (Meudon)	X.Bonnin (ROC)	Beginning of the RPW PFM thermal calibration
RSS3-0	RSS "0th E2E test" version data package		X.Bonnin (ROC)	Beginning of the SOV "0th E2E" test campaign
RSS3	RSS "CP E2E test" version data package	Preliminary version, ready for the MOC/SOC SOV E2E test, including: - Partial MUSIC Web tool capabilities (Procedures list viewing, importing and model validating/IOR creation and model validating, E-FECS and TMC data plotting, TM/TC visualization) - Full RPW LLVM expected capabilities (i.e., "processing+test" version) - SOC GFTS interface up and running - ROC-SGSE V2 - RODP "E2E" version, with partial data processing The data package shall include the following documents: - First draft of the ROC User Manual and Reference Guide (system requirements, installation process, tutorial for the FIGARO/FAUST/TV tool and for the RODP basic execution)	X.Bonnin (ROC)	Beginning of the SOV "CP E2E" test campaign
RSS4	RSS "ready for flight" version data package	Full validated version, ready for the commissioning and cruise phase operations, including: - Full processing of the RPW data products (including dissemination and archiving) - Full MUSIC Web tool capabilities, except for SISSI. - SOC/MOC DDS/GFTS interfaces up and running - RPW LLVM SOC and LESIA instances up and	X.Bonnin (ROC)	Launch

		running - ROC-SGSE mission instance ready to be used for operations - Advanced draft of the ROC User Manual and Reference Guide (all functionalities described, without SISSI tool)		
RSS5	RSS “fully operational” version data package	Fully operational RSS, including: - Full MUSIC tool expected capabilities - Full RODP tool expected capabilities - Full LLVM expected capabilities To be delivered at the end of the commissioning phase. - First issue of the ROC User Manual and Reference Guide (all functionalities covered)	X.Bonnin (ROC)	Beginning of the CP

Table 16. RSS data package release main releases schedule.

6.2.2 ROC software main deliveries schedule

Table below gives the main deliveries of all ROC software products.

The RSS release dependencies must be consistent with both the project and dev. planning. Especially, the LLVM delivery shall be anticipated w.r.t. the SOC LLVM delivery schedule (see PMP). In the same way, the effective integration of the RCS into the RSS shall be realized and validated prior to the RSS release. In another word, the RCS team shall plan to deliver their software to the ROC enough time before the release.

Detailed schedule will have to be reported and kept up-to-date in the ROC planning file.

Software deliverable	Main available capabilities	Software validation leader	Developer(s) in charge	RSS release dependencies
ROC GSE				
SimuSBM1 software V1	Version for the SBM1 detection RPW DPU software validation on-ground campaign	Xavier Bonnin (ROC)	- Xavier Bonnin (with the support of Oksana Kruparova)	Delivered with the RSS0
SimuSBM1 software V2	Second version to be used for SBM1 detection algorithm validation and optimization in-flight.	Xavier Bonnin (ROC)	- Antonio Vecchio	Delivered with the RSS5

SimuSBM2 software V1.0.0	Version for the SBM2 detection RPW DPU software validation on-ground campaign	Xavier Bonnin (ROC)	- Xavier Bonnin (with the support of Milan Maksimovic and Olga Alexandrova)	Delivered with the RSS0
SimuSBM2 software V2.0.0	Second version to be used for SBM2 detection algorithm validation and optimization in-flight.	Xavier Bonnin (ROC)	- Antonio Vecchio	Delivered with the RSS5
RPL V0.1.0	First version of the RPW Packet Parsing Library. Compliant with the IDB 2.2.3. To be used in priority by the ROC-SGSE.	Xavier Bonnin (ROC)	- Thierry Sauzière	Delivered with the RSS0
ROC-SGSE V1	Version to be used for the EM2 blank tests calibration campaigns at CNES (Toulouse). Includes the RPL V1.0.	Xavier Bonnin (ROC)	- Manuel Duarte	Delivered with the RSS1
ROC-SGSE V2	Version to be used for the PFM thermal calibration campaigns at LESIA (Meudon).	Xavier Bonnin (ROC)	- Manuel Duarte, - Xavier Bonnin (after M.Duarte leave)	Delivered with the RSS2
RPW Low Latency Virtual Machine (LLVM) V0.1.0	First "Hello World" VM configuration and test version.	Richard Carr (SOC)	- Xavier Bonnin	Delivered with the RSS1
RPW Low Latency Virtual Machine (LLVM) V0.X.0	Second "processing data" version. Delivered with an up-to-date RPW Low Latency Data Description Document (LLDDD) and "Testcard" scenario data LL01 CDF.	Richard Carr (SOC)	- Sonny Lion	Delivered with the RSS3-0
RPW Low Latency Virtual Machine (LLVM) V1.0.0	Full version "processing+test". - Fully compliant and validated. Delivered with the first issue of the RPW LLVM Design Document and User Manual.	Richard Carr (SOC)	- Sonny Lion	Delivered with the RSS3
RCS – THR_CALBAR				
THR_CALBAR	Software version being able	Milan	- Antonio	Delivered with

preliminary version	of producing ROC-SGSE THR preliminary L2S CDF files. Delivered with the THR L2S master CDF files.	Maksimovic (LESIA)	Vecchio - Quynh Nhu Nguyen	the RSS2
THR_CALBAR "processing" version	Software version being able of producing ROC-SGSE THR L2S CDF files. Delivered with the THR L2S master CDF files, the SRS issue 1 and the draft of SUM.	Milan Maksimovic (LESIA)	- Antonio Vecchio - Quynh Nhu Nguyen	Delivered with the RSS3
THR_CALBAR "ready-to-flight" version	Software version being able of producing SolO RPW THR L2 CDF files – ROC interface compliant according to RCS ICD. Delivered with the THR L2 master CDF files, SRS issue 1 and SUM issue 1.	Milan Maksimovic (LESIA)	- Lorenzo Matteini - Quynh Nhu Nguyen	Delivered with the RSS4
THR_CALBAR "fully operational" version	Fully validated version of the software after commissioning.	Milan Maksimovic (LESIA)	- Lorenzo Matteini - Quynh Nhu Nguyen	Delivered with the RSS5
RCS – LFR_CALBUT				
LFR_CALBUT preliminary version	Software version being able of producing ROC-SGSE LFR preliminary L2R/L2S CDF files. Delivered with the LFR L2R/L2S master CDF files.	Thomas Chust (LPP)	- Bruno Katra	Delivered with the RSS2
LFR_CALBUT "processing" version	Software version being able of producing ROC-SGSE LFR L1R/L2S-non-WF CDF files. Delivered with the LFR L1R/L2S master CDF files, the SRS issue 1 and the draft of SUM.	Thomas Chust (LPP)	- Bruno Katra	Delivered with the RSS3
LFR_CALBUT "ready-to-flight" version	Software version being able of producing SolO RPW LFR L1R/L2-non-WF CDF files – ROC interface compliant according to RCS ICD. Delivered with the LFR L1R/L2 master CDF files, SRS issue 1 and SUM issue 1.	Thomas Chust (LPP)	- Bruno Katra - Rodrigue Piberne	Delivered with the RSS4
LFR_CALBUT	Fully validated version of the	Thomas	- Bruno	Delivered with

“fully operational” version	software after commissioning.	Chust (LPP)	Katra, - Rodrigue Piberne	the RSS5
RCS – TDS_CALBA				
TDS_CALBA preliminary version	Software version being able of producing ROC-SGSE TDS preliminary L2R/L2S CDF files. Delivered with the TDS L2R/L2S master CDF files.	Jan Soucek (IAP)	- David Pisa	Delivered with the RSS2
TDS_CALBA “processing” version	Software version being able of producing ROC-SGSE TDS L1R/L2S-non-WF CDF files. Delivered with the LFR L1R/L2S master CDF files, the SRS issue 1 and the draft of SUM.	Jan Soucek (IAP)	- David Pisa	Delivered with the RSS3
TDS_CALBA “ready-to-flight” version	Software version being able of producing SolO RPW LFR L1R/L2-non-WF CDF files – ROC interface compliant according to RCS ICD. Delivered with the LFR L1R/L2 master CDF files, SRS issue 1 and SUM issue 1.	Jan Soucek (IAP)	- David Pisa	Delivered with the RSS4
TDS_CALBA “fully operational” version	Fully validated version of the software after commissioning.	Jan Soucek (IAP)	- David Pisa	Delivered with the RSS5
RCS – BICAS				
BICAS preliminary version	Software version being able of producing ROC-SGSE TDS/LFR preliminary L2S-E-WF CDF files. Delivered with the TDS/LFR L2S-E-WF master CDF files.	Andris Vaivads	- Erik Johansson	Delivered with the RSS2
BICAS “processing” version	Software version being able of producing ROC-SGSE TDS/LFR L2S-E-WF CDF files. Delivered with the TDS/LFR L2S-E-WF master CDF files, the SRS issue 1 and the draft of SUM.	Andris Vaivads	- Erik Johansson	Delivered with the RSS3
BICAS “ready-to-flight”	Software version being able of producing SolO RPW	Andris Vaivads	- Erik Johansson	Delivered with the RSS4

version	TDS/LFR L2-E-WF CDF files – ROC interface compliant according to RCS ICD. Delivered with the TDS/LFR L2-E-WF master CDF files, SRS issue 1 and SUM issue 1.			
BICAS “fully operational” version	Fully validated version of the software after commissioning.	Andris Vaivads	- Erik Johansson	Delivered with the RSS5
RCS – SCMCAL				
SCMCAL preliminary version	Software version being able of producing ROC-SGSE TDS/LFR preliminary L2S-B-WF CDF files. Delivered with the TDS/LFR L2S-B-WF master CDF files.	Matthieu Kretzschmar	- Jean-Yves Brochot - Gamil Cassam-Chenai	Delivered with the RSS2
SCMCAL “processing” version	Software version being able of producing ROC-SGSE TDS/LFR L2S-B-WF CDF files. Delivered with the TDS/LFR L2S-B-WF master CDF files, the SRS issue 1 and the draft of SUM.	Matthieu Kretzschmar	- Jean-Yves Brochot - Gamil Cassam-Chenai	Delivered with the RSS3
SCMCAL “ready-to-flight” version	Software version being able of producing SolO RPW TDS/LFR L2-B-WF CDF files – ROC interface compliant according to RCS ICD. Delivered with the TDS/LFR L2-B-WF master CDF files, SRS issue 1 and SUM issue 1.	Matthieu Kretzschmar	- Jean-Yves Brochot - Gamil Cassam-Chenai	Delivered with the RSS4
SCMCAL “fully operational” version	Fully validated version of the software after commissioning.	Matthieu Kretzschmar	- Jean-Yves Brochot - Gamil Cassam-Chenai	Delivered with the RSS5
RODP				
RODP “E2E test” version	- Partial data processing capabilities (LZ, L0, L1/HK productions)	X.Bonnin	X.Bonnin S.Lion Q.N. Nguyen	Delivered with the RSS3
RODP “ready-for-flight” version	- Full data processing capabilities (TM packet, TM/TC report and ancillary	X.Bonnin	X.Bonnin S.Lion Q.N. Nguyen	Delivered with the RSS4

	data retrieval, LZ, L0, L1/HK and L2 productions, summary plots and ancillary data processing, dissemination and archiving) - Automated data monitoring and user notification			
RODP "fully operational" version	Fully validated version of the software after commissioning.	X.Bonnin	X.Bonnin S.Lion Q.N. Nguyen	Delivered with the RSS5
MUSIC (including TRAC, POCA and INCA)				
MUSIC "preliminary" version	- TRAC preliminary version to compute RPW TM rate	X.Bonnin (ROC)	T.Sauzière	Delivered with the RSS0
MUSIC "E2E test" version	- FIGARO: flight procedure visualization and importation - FAUST: MTP/STP IOR generation from sequences - OPERA: E-FECS/TMC data visualization over MTP timeline	X.Bonnin (ROC)	S.Lion Q.N Nguyen A.Aboubacar N.Fuller	Delivered with the RSS3
MUSIC "ready-for-flight" version	- FIGARO: full capabilities - FAUST: full capabilities - OPERA: full capabilities - SISSI: preliminary version (SBM1/SBM2 list visualisation)	X.Bonnin (ROC)	S.Lion Q.N Nguyen A.Aboubacar N.Fuller	Delivered with the RSS4
MUSIC "fully operational" version	Fully functional, validated version of the software after commissioning.	X.Bonnin (ROC)	S.Lion Q.N Nguyen A.Aboubacar N.Fuller	Delivered with the RSS5

Table 17. ROC software main deliveries schedule.

7 LIST OF TBC/TBD/TBWS

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

8 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	LESIA CNRS	x	N. FULLER
	x	Y. DE CONCHY			
	x	X. BONNIN			
	x	QN NGUYEN			
		B. CECCONI			
	x	A. VECCHIO			
	x	S. LION			
	x	A. ABOUBACAR			

EXTERNAL (To modify if necessary)

CNES		C. FIACHETTI	AsI/CSRC		J.BRINEK
	x	E. BELLOUARD			P.HELLINGER
		R.LLORCA-CEJUDO			D.HERCIK
		E.LOURME			P.TRAVNICEK
		M-O. MARCHE	IAP		J.BASE
	x	E.GUILHEM			J. CHUM
	x	E. LE DE			I. KOLMASOVA
	x	M. ROUZE			O.SANTOLIK
	x	J-M. TRAVERT		x	J. SOUCEK
	x	D. RAULIN		x	D. PISA
IRFU		L. BYLANDER	IWF		G.LAKY
		C.CULLY			T.OSWALD
		A.ERIKSSON			H. OTTACHER
	x	E.JOHANSSON			H. RUCKER
	x	A.VAIVADS			M.SAMPL
	x	Y. KHOTYAINTSEV			M. STELLER
LPC2E	x	J.-Y. BROCHOT	LPP	x	T.CHUST
		G. JANNET			A. JEANDET
		T.DUDOK de WIT			P.LEROY
	x	M. KRETZSCHMAR			M.MORLOT
	x	G. CASSAM-CHENAI		x	B. KATRA
SSL		S.BALE		x	R. PIBERNE