

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		27/09/2016
Verified by:	Function:	Signature:	Date
Yvonne de Conchy	Team Member #2		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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 2 / 58 -

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 3 / 58 -

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
 Issue: 02
 Revision: 03
 Date: 17/11/2017

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
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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 5 / 58 -

SoIO	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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 6 / 58 -

Table of Contents

1	General	9
1.1	Scope of the Document	9
1.2	Applicable Documents	9
1.3	Reference Documents.....	9
1.4	About this document	12
1.4.1	Access policy	12
1.4.2	Terminology	12
2	ROC software products overview	12
2.1	ROC Software System (RSS)	12
2.1.1	RSS product tree.....	12
2.1.2	ROC Operations And Data System (ROADS)	13
2.1.3	ROC Ground Support Equipment (GSE)	18
2.2	ROC databases.....	19
2.2.1	ROC Mission Database (MDB)	19
2.2.2	RPW Instrument Database (IDB)	20
2.2.3	Solar Orbiter Mission Information base (MiB)	20
2.2.4	ROC-SGSE Test database (TDB).....	20
2.2.5	ROC MEB GSE database	20
2.3	ROC third-party software products.....	20
2.3.1	The Plugin-Oriented Pipeline for Python (POPPy).....	21
2.3.2	RPW user software tools and services	21
2.3.3	Software in support to the operation and data processing activities	21
2.3.4	Project management software products.....	22
2.3.5	Solar Orbiter Inter-instrument software products	22
2.3.6	External collaboration software products	22
3	ROC data products overview	22
3.1	File naming convention, data format and metadata definition	22
3.2	Data product suppliers and customers	22
4	ROC software management approach.....	24
4.1	Master schedule	24
4.2	Assumptions, dependencies and constraints	24
4.2.1	Assumptions.....	24
4.2.2	Dependencies	24
4.2.3	Constraints	24
4.3	Software development related risks	25
4.4	Monitoring and controlling mechanisms.....	26
4.4.1	ROC software development “sprint” mechanism	26
4.4.2	Software development tools.....	26
4.4.3	ROC facilities access management file.....	26
4.4.1	ROC data product management approach	26
4.5	Staffing plan	27
4.5.1	ROC project organization and responsibilities	27
4.5.2	ROC software product responsibilities	28
4.5.3	RCS specific responsibilities	28
4.5.4	RPW science data production, validation and archiving responsibilities during the Solar Orbiter mission	29
4.5.5	RPW science data production, validation and archiving responsibilities during the ground calibrations at system level.....	31



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 7 / 58 -

4.5.6	ROC software logistics support responsibilities	32
4.6	Software procurement process	32
4.7	Supplier management.....	33
4.7.1	RCS specific development plan	33
5	ROC software development approach	34
5.1	Strategy to the software development	34
5.1.1	General convention and philosophy.....	34
5.1.2	Software versioning convention	36
5.1.3	Software development configuration.....	36
5.1.4	ROC engineering validation plan	37
5.1.5	ROC software product assurance plan	37
5.2	ROC software development life cycle	37
5.2.1	ROC software development life cycle	37
5.3	Software engineering standards and techniques	38
5.3.1	ROC Python-based software engineering standards and techniques	38
5.3.2	ROC JavaScript-based software engineering standards and techniques.....	38
5.3.3	LLVM specific engineering standards and techniques.....	39
5.3.4	RCS specific engineering standards and techniques.....	39
5.4	Software development, testing, validating and execution environments at LESIA	39
5.4.1	RSS operational infrastructure overview.....	39
5.4.1	RSS development, testing and validating infrastructure overview	40
5.4.2	ROC hardware equipment	40
5.4.3	ROC hardware communication interfaces at the LESIA site.....	44
5.4.4	Software environment	44
5.5	Software development, testing, validating and execution environments at ESOC.....	47
5.5.1	RSS operational infrastructure overview.....	47
5.5.2	RSS development, testing and validating infrastructure overview	49
5.5.3	ROC hardware equipment	49
5.5.4	ROC hardware communication interfaces at the ESOC site.....	49
5.6	Software documentation plan.....	50
5.6.1	General	50
5.6.2	Software documentation identification	50
5.6.3	Deliverable items.....	50
5.6.4	Software documentation standards.....	50
6	ROC software delivery plan	50
6.1	ROC software development planning	50
6.2	ROC software main deliveries schedule	50
7	List of TBC/TBD/TBWs	57
8	Distribution list.....	58



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
Issue: 02
Revision: 03
Date: 17/11/2017

- 8 / 58 -

List of figures

Figure 1. ROC Software System product tree.....	13
Figure 2. ROC Operations And Data System (ROADS) software products.....	14
Figure 3. ROC Ground Support Equipment related software products.....	18
Figure 4. ROC software development life cycle.....	37
Figure 5. RSS infrastructure overview at LESIA.....	40
Figure 6. RSS infrastructure at ESOC.....	48

List of tables

Table 1. Terminology.....	12
Table 2. RODP plugins.....	15
Table 3. RPW Calibration Software (RCS).....	16
Table 4. Categories of ROC data products.....	23
Table 5. Categories of risk related to software development.....	26
Table 6. ROC dataset listing file columns.....	27
Table 7. ROC software customers versus suppliers.....	28
Table 8. RCS responsibilities.....	29
Table 9. Science calibrated data validation leader.....	31
Table 10. Logistics support to the ROC at LESIA and Paris Observatory.....	32
Table 11. RCS validation tests schedule.....	34
Table 12. RCS documentation delivery schedule.....	34
Table 13. Expected technical documents during ROC software development life-cycle.....	38
Table 14. ROC operating servers.....	44
Table 15. Technical accounts.....	45
Table 16. RSS software programming languages.....	46
Table 17. List of databases involved in the ROC activities.....	46
Table 18. List of RSS main software libraries.....	47
Table 19. Hardware equipment for MOC RSS.....	49
Table 20. ROC software main deliveries schedule.....	56



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 9 / 58 -

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	In preparation
AD2	ROC-GEN-OTH-NTT-00036-LES/1/0	ROC Project Glossary of terms	Xavier Bonnin	13/09/2017
AD3	ROC-GEN-MGT-PLN-00026-LES/1/3	ROC Project Management Plan (PMP)	Yvonne de Conchy, Xavier Bonnin	20/12/2016
AD4	ROC-GEN-MGT-QAD-00033-LES/0/3	ROC Software Assurance /Product Assurance Plan (SPAP)	Stephane Papais	14/11/2016

1.3 Reference Documents

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
 Issue: 02
 Revision: 03
 Date: 17/11/2017

- 10 / 58 -

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)	Erik de Witt	03/08/2012
RD3	RPW-GEN-DAT-SPC-00006-LES/01/00	RPW Data Products	Xavier Bonnin	23/12/2016
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-ICD-0005/1/1	Solar Orbiter Low-Latency Data: Concept and Implementation	Anik de Groof	Sept., 2014
RD6	SOL-SGS-TN-0009/2/2	Metadata Definition for Solar Orbiter Science Data	Anik De Groof	15/01/2015
RD7	SOL-SGS-ICD-0002_DPICD/0/2	Data Producer to archive Interface Control Document	Luis Sanchez	05/09/2014
RD8	SOL-SGS-ICD-0003/0/2	Solar Orbiter Instrument Operation Request Interface Control Document (IOR ICD)	Christopher Watson	19/08/2014
RD9	SOL-SGS-0004/0/3	Solar Orbiter Interface Control Document for Low Latency Data CDF files	Andrew Walsh	Feb., 2015
RD10	SOL-SGS-TN-0006/1/0	SOC Engineering Guidelines for External Users	Richard Carr	04/03/2015
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-10001/1/0	Solar Orbiter FOP Preparation Plan	B. Sousa	05/09/2013
RD14	SOL-ESC-TN-12000/1/0	Solar Orbiter Mission Planning Concept	Ignacio Tanco	27/09/2013
RD15	SOL-SGS-ICD-0006/0/1	Solar Orbiter Enhanced-Flight Events Communications Skeletons Interface Control Document	Christopher Watson	05/07/2015
RD16	ROC-GEN-SYS-NTT-00008-LES/1/3	ROC Engineering Guidelines (REG)	Xavier Bonnin	09/11/2016
RD17	ROC-GEN-SYS-NTT-00019-LES/1/1	ROC Engineering Guidelines for External Users (REGU)	Xavier Bonnin	18/11/2015
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-TS-0006/0/0	Solar Orbiter Instrument Teams – SOC Test Specification	Nana Bach Christopher J. Watson	16/12/2015
RD21	ROC-PRO-PIP-ICD-00037-LES/1/1	RPW Calibration Software Interface Control Document (RCS ICD)	Manuel Duarte, Xavier Bonnin	In preparation
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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 11 / 58 -

RD25	TMC-ICD SOL-SGS-ICD-0007 v0_4	Solar Orbiter Telemetry Corridor Interface Control Document	Christopher J. Watson	25/01/2016
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 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 foundation	2016
RD34	https://docs.python.org/3/install/	Installing Python modules (Legacy Version)	Python software foundation	2016
RD35	ROC-TST-GSE-SUM-00035-LES/01/00	POPPy framework user manual	M.Duarte	24/06/2016
RD36	ROC-GEN-SYS-SPC-00026-LES/01/00	ROC Software System Specification (RSSS)	X.Bonni n	09/01/2017
RD37	ROC-GEN-SYS-SPC-00036-LES/00/10	ROC Software System Design Document (RSSDD)	X.Bonni n	09/01/2017
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.Bonni n, Sonny Lion	In preparation
RD40	RPW-SYS-MEB-GSE-SPC-00125-LES/01/01	RPW MEB GSE Description	Loic Gueguen	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-0009/0/2	Solar Orbiter Archive Plan	Pedro Osuna	01/07/2015
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	01/08/2013

			Warsaw, Nick Coghlan	
--	--	--	----------------------------	--

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].



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 13 / 58 -

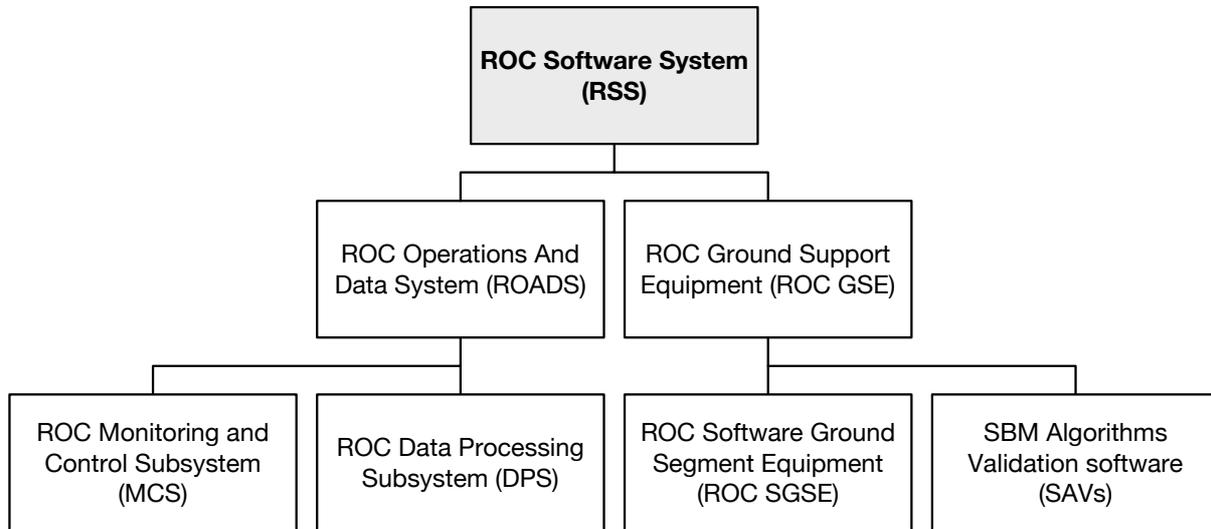


Figure 1. ROC Software System product tree.

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), which regroupes the tools and infrastructure to give an access to the ROC data.
- The RPW Data Archive (DArc), which regroupes the tools and 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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
 Issue: 02
 Revision: 03
 Date: 17/11/2017

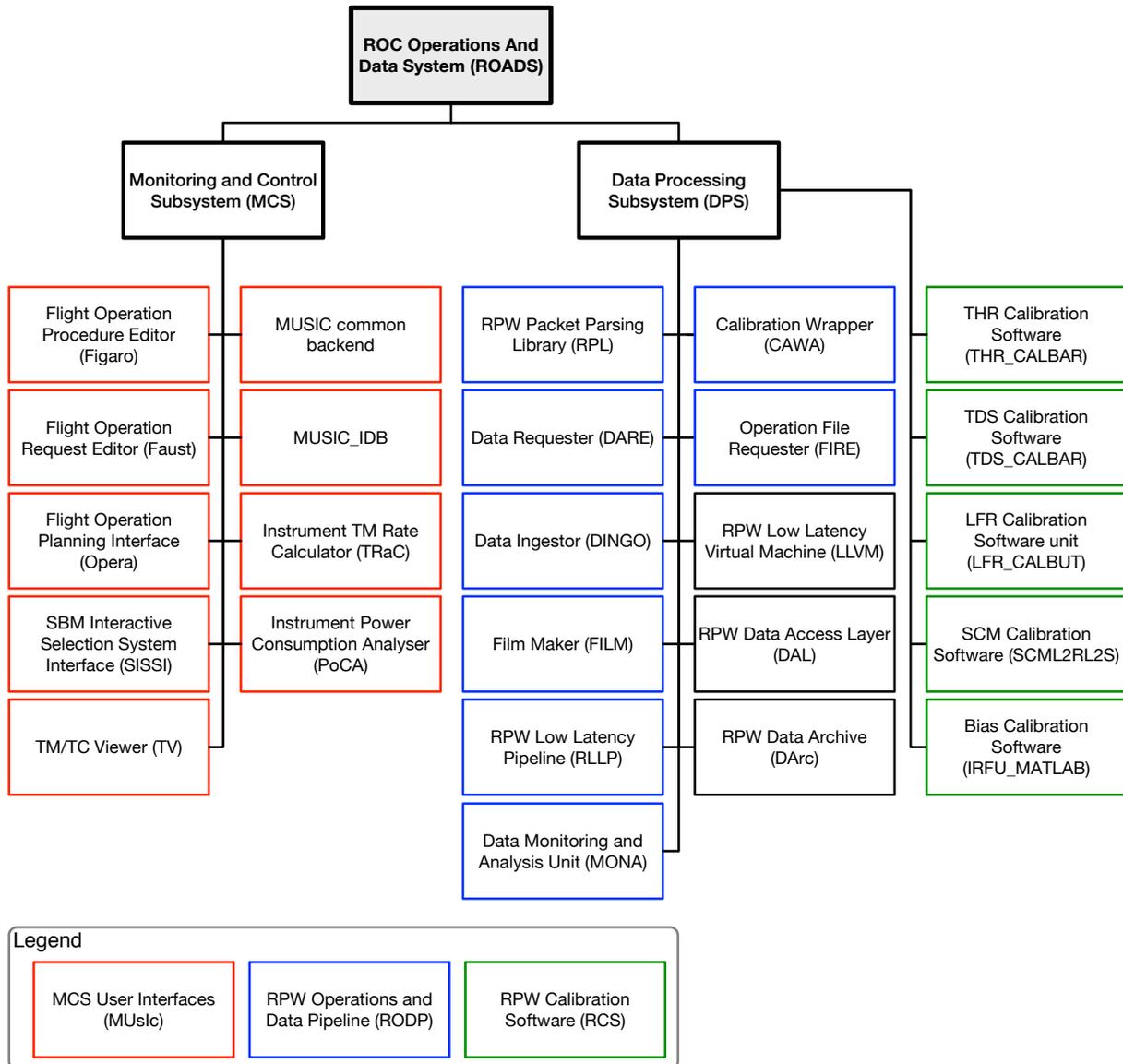


Figure 2. ROC Operations And Data System (ROADS) software products.

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 main automated tasks in terms of science/HK/ancillary data retrieval and processing, but also for operation inputs retrieval and

The RODP is built with the “Plugin-Oriented Pipeline for Python” framework (POPPy) [RD35], 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.

RODP plugin name	Function
<i>Data REquester (DARE)</i>	Plugin in charge of requesting data delivered



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 15 / 58 -

	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.
<i>RPW Packet parsing Library (RPL)</i>	Plugin in charge of parsing (i.e., de-commute/decompress) the RPW TM/TC packets
<i>FILE Maker (FILM)</i>	Plugin in charge of producing the LZ, L0, L1 and HK data files from the retrieved RPW TM packets analysis
<i>Data INGestOr (DINGO)</i>	Plugin in charge of ingesting data into the MDB
<i>RPW CALibration WrApper (CaWa)</i>	Plugin in charge of executing the RPW Calibration Software (RCS)
<i>RPW LOw LATency Pipeline (LOLA)</i>	Plugin in charge to run the Low Latency Data Pipeline (LLDP) for RPW.
<i>Data MONitoring and Analysis Unit (MONA)</i>	Plugin in charge of perform automated analysis of the downlinked RPW data (i.e., TC ack. analysis, TM assertion, HK/science data limits verification)
<i>Pipeline UNit Keeper (PUNK)</i>	Plugin in charge of monitoring the pipeline activities and notifying users.
<i>Pipeliner mappER (PIPER)</i>	Plugin in charge of initializing the pipeline and ensuring that static meta-data, required by the pipeline to run, are well defined
<i>Pipeline Operations Planner (POP)</i>	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).



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 16 / 58 -

The RCS are developed and maintained by the teams in charge (see table below). It must be delivered to the ROC following specific procedures [RD17] and must include a standardized interface [RD21], in order to be called by CAWA¹.

RCS name	Function	Team in charge	Programming Languages
<i>THR data CALiBrAtion SoftwaRe (THR_CALBAR)</i>	Produce TNR-HFR L2/L2S calibrated science data files in the CDF format	TNR-HFR team (LESIA, Meudon)	IDL
<i>TDS data CALiBrAtion SoftwaRe (TDS_CALBA)</i>	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
<i>LFR data CALibration UniT (LFR_CALBUT)</i>	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
<i>SCMCAL</i>	Produce TDS/LFR L2/L2S magnetic WF calibrated data files in the CDF format	SCM team (LPC2E, Orléans)	IDL
<i>Bias CALibration SoftwaRe (BICAS)</i>	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 [RD10].

¹ Since each RCS can be written in different programming languages, the RODP calling interface must be as



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 17 / 58 -

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 regroupes 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 POver 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.

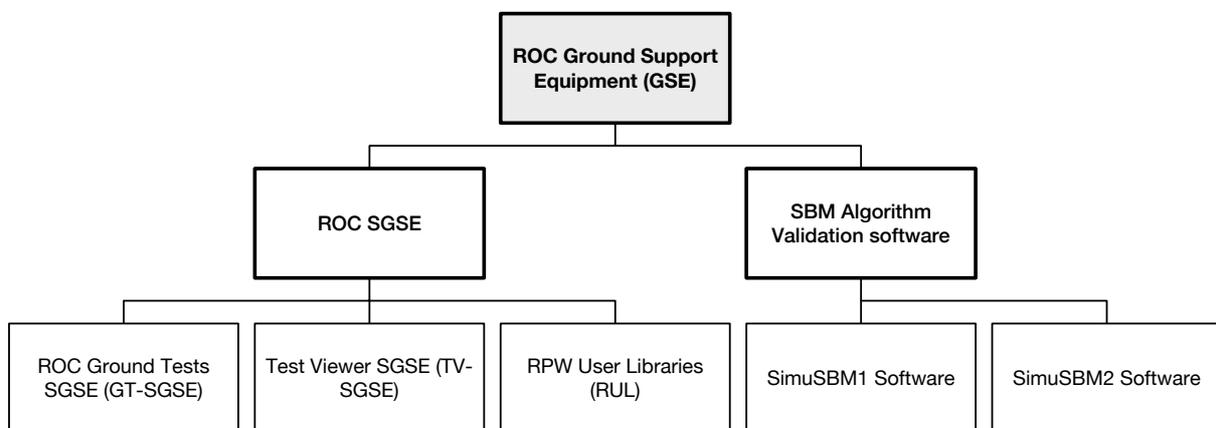


Figure 3. ROC Ground Support Equipment related software products.

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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 19 / 58 -

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 synthesized input data and produce CDF format files that comply the specification defined in [RD17].
- Perform SBM event detection and report results into XML format files that comply the specification defined in [RD17].

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)



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 20 / 58 -

- 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 an IDB compatible with this MiB 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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 21 / 58 -

2.3.1 The Plugin-Oriented Pipeline for Python (POPPy)

The architecture of both the ROC-SGSE and RODP software 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 plugins.

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, producing L1 science data files, etc.). To execute a given function, the plugin relies on a sequence of “tasks”, which must be written using the framework convention and can be called from a standardized command line interface (CLI).

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 will be developed by the Bias team.
- Effective electrical antenna length and direction determination in-flight. This software will be developed by the TNR-HFR team.

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 22 / 58 -

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” 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)
<i>MEB SGSE test log</i>	MEB SGSE test log XML format	RPW MEB GSE team, via the ROC-SGSE MEB GSE database interface	ROC
<i>ROC-SGSE data (HK and uncalibrated science L1 data)</i>	CDF	ROC (generated by the ROC-SGSE from the MEB SGSE test log files)	RPW Consortium, AIT-AIV CNES/LESIA teams
<i>RPW TM packet data</i>	EDDS XML format	MOC (via the EDDS)	ROC
<i>RPW TC catalogue data</i>	EDDS XML format	MOC (via the	ROC



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 23 / 58 -

		EDDS)	
<i>RPW TC packet sequences for instrument nominal operation requests</i>	IOR XML format	ROC (via the SOC GFTS)	SOC
<i>RPW TC packet sequences for instrument non-routine operation requests (e.g., contingency, patches)</i>	PDOR/MDOR XML format	ROC (via the MOC GFTS)	MOC
<i>RPW Flight Operation Procedures</i>	MOIS importer format (Microsoft Excel©)	ROC (directly by email to the point of contact at the MOC)	MOC
<i>MEB SGSE GEDEON import script</i>	MEB SGSE GEDEON XML format	ROC (exported by the FAUST tool)	ROC
<i>RPW LZ data</i>	XML	ROC (generated by the RODP from the RPW TM packet data)	ROC
<i>RPW L0 data</i>	HDF5	ROC (generated by the RODP from the LZ data)	ROC, RPW Consortium
<i>RPW L1/L1R/L2 science data</i>	CDF	ROC (generated by the RODP from the L0 data)	RPW Consortium, ESA/CDPP data archives, Science Community
<i>RPW HK parameters</i>	CDF	ROC (generated by the RODP from the L0 data)	RPW Consortium
<i>Solar Orbiter MOC Orbit/attitude/time/frame data</i>	SPICE kernels	SOC (via GFTS)	ROC
<i>Pre-Processed Orbit/attitude/time/frame data for RPW</i>	CDF	SOC	RPW Consortium, ESA/CDPP data archives, Science Community
<i>Quick-looks</i>	PNG	ROC (generated by the RODP)	RPW Consortium, ESA/CDPP data archives, Science Community
<i>E-FECS file</i>	XML	SOC (via the GFTS)	ROC
<i>TMC file</i>	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 software².

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 “spare” model (or S/C simulator) to validate its RFP.
- 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]

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

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
 Issue: 02
 Revision: 03
 Date: 17/11/2017

- 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 phase. 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
<i>Hardware/Operating System (OS) failure at LESIA</i>	Obsolescence, overvoltage	Delay in development, which can cause a delay in the delivery	High	Medium	Plan backup systems to be rapidly deployed. 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)
<i>Obsolete software/hardware</i>	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
<i>Unexpected personnel reduction in the developer team</i>	Transfer, voluntary redundancy, contract end, disease, pregnancy, ...	Under-sized team, loss of expertise, delay in development	High	Low	Prompt Non permanent post hiring or internal replacement by the LESIA
<i>Lack of experience</i>	Developer not familiar with	Software quality loss,	Medium	Medium	The ROC shall offer the



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 26 / 58 -

	a software technology	required specification not reached, software delivery delay			possibility to its developer team to get professional training.
--	-----------------------	---	--	--	---

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



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 27 / 58 -

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

Table 6. ROC dataset listing file columns.

4.5 Staffing plan

This section presents the responsibilities and organization of key personnel and institutes involved in the ROC software development.

4.5.1 ROC project organization and responsibilities

The organization, personnel and responsibilities at the project level are detailed in the PMP.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 28 / 58 -

4.5.2 ROC software product responsibilities

4.5.2.1 Customer versus supplier roles

The following table resumes the customer versus supplier roles concerning the systems, software and database units of the RSS.

Software System	Customer	Supplier
ROADS	ROC	ROC
ROC GSE	ROC	ROC
LLVM	SOC (primary instance run at ESAC) / ROC (secondary instance run at LESIA)	ROC
RCS	ROC	RPW analyser/sensor teams (BIAS team for BICAS, SCM team for SCMCAL, THR team for THR_CALBAR, TDS team for TDS_CALBA and LFR team for LFR_CALBUT)
MEB GSE tools and databases	ROC	MEB GSE team
ROC MDB	ROC	ROC
RPW IDB	ROC	Flight software team (for PALISADE version) / MOC (for MiB version)
Solo Mission Information Database	ROC	MOC

Table 7. ROC software customers versus suppliers.

4.5.2.2 Key personnel responsibilities

According to the CIRD, the RPW Ground Segment Project Manager, **Xavier Bonnin**, must ensure that the design of the ROC software, interfaces and data products meet the requirements addressed at both the RPW and Solar Orbiter levels. Moreover, she/he must approve the ROC engineering documentation.

The RPW Ground Segment Deputy Project Manager, **Sonny Lion**, must supervise the ROC software and data design definition, development, testing and validation. After the launch, he must ensure that the ROC software system is always up and running.

The RPW sensor/analyser lead CoIs: **Jan Soucek** (TDS), **Thomas Chust** (LFR), **Milan Maksimovic** (THR), **Andris Vaivads** (BIAS) and **Volodya Krasnosselskikh** (SCM) must ensure that their RCS software, data and documentation they have to deliver to the ROC are compliant, operational and up-to-date during all of the phases of the mission. Especially, this implies that RCS data products are well calibrated and up-to-date.

4.5.3 RCS specific responsibilities

The RCS will have to be delivered to the ROC by the teams in charge. The ROC is in charge of the S/W execution only, however it is the responsibility of the teams to ensure the maintenance of the RCS during the mission.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 29 / 58 -

The following table summarizes the list of RCS to be developed by which team and for which data products.

Software name	Entity in charge of the software development, test, validation, maintenance and delivery	Entity in charge of the software execution	Data products generated by the RCS	Software programming language
BICAS	BIAS team (IRF-U)	ROC team (LESIA)	- TDS (LFM) / LFR L2/L2S electrical WF data	Matlab
LFR_CALBUT	LFR team (LPP)	ROC team (LESIA)	- LFR L2/L2S LFR data products, except WF data - LFR L1R electrical/magnetic WF data	Python
SCMCAL	SCM team (LPC2E)	ROC team (LESIA)	- TDS/LFR L2/L2S magnetic WF data	IDL
TDS_CALBA	TDS team (IAP)	ROC team (LESIA)	- TDS L2/L2S data products, except WF data - TDS L1R electrical/magnetic WF data	IDL
THR_CALBAR	THR team (LESIA)	ROC team (LESIA)	- THR L2 data products	IDL

Table 8. RCS responsibilities.

The detailed list of people involved is given in the PMP.

4.5.4 RPW science data production, validation and archiving responsibilities during the Solar Orbiter mission

4.5.4.1 Science data production

The ROC will be in charge to produce the RPW science data at the LZ, L0, L1R, L1 and L2 processing levels during the mission. This hence includes also the calibrated science data, using the RCS delivered by the sub-system teams.

4.5.4.2 Science data quality validation responsibilities

Several steps of validation will have to be realized between the RPW TM data retrieval at the LESIA and the delivery of definitive science calibrated data to the ESA archive centre.

The ROC science team will be in charge of the final validation of the science data. Nevertheless, the expertise of the sub-systems teams will be required to refine calibration process, at least of the beginning of the mission.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 30 / 58 -

Table 9 presents the person in charge of the calibrated science data validation (i.e. data validation leader) at the LESIA, given by science data products category. The contributors from each sub-system team are also indicated; they are supposed to support the ROC for producing good quality science data. Technically speaking, it should mainly consist of delivering to the ROC, refined version of the RCS calibration tables.

Science data products category	Data validation leader at LESIA	Data validation contributor(s)
LFR spectral products – electric component	M.Maksimovic (TBC)	T.Chust (LPP)
LFR WF products – electric component	F.Pantellini (TBC)	A.Vaivads (IRF-U), T.Chust (LPP)
LFR spectral products – magnetic component	O.Alexandrova	M.Kretzschmar (LPC2E), T.Chust (LPP)
LFR WF products – magnetic component	O.Alexandrova	M.Kretzschmar (LPC2E), T.Chust (LPP)
LFR BP1/BP2 products	TBD	T.Chust (LPP)
THR products - electric component	L.Matteini	M.Maksimovic
THR products - magnetic component	L.Matteini	M.Kretzschmar (LPC2E), M.Maksimovic
TDS histogram, stats, MAMP and SM-PDS products	M.Maksimovic (TBC)	J.Soucek (IAP)
TDS WF products – electric component	M.Maksimovic (TBC)	J.Soucek (IAP)
TDS WF products – magnetic	M.Maksimovic (TBC)	M.Kretzschmar (LPC2E), J.Soucek (IAP)



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 31 / 58 -

component		
-----------	--	--

Table 9. Science calibrated data validation leader.

4.5.4.3 Science data distribution and archiving responsibilities

All of the processed science data files will be stored at LESIA. Preliminary LZ/L1/HK/ANC data will have to be distributed by the ROC within a short time - less than 24 hours after the downloading through the MOC DDS - to the RPW consortium and other instrument teams.

The definitive data will have to be delivered to the SOC within 3-months, in order to be archived at the ESAC data archive centre. The way the science data will be archived at the SOC is described in [RD43].

RPW science data archiving at the CDP (Toulouse) is also foreseen, but the details are not known at this stage of the project.

4.5.5 RPW science data production, validation and archiving responsibilities during the ground calibrations at system level

4.5.5.1 Science data production and distribution responsibilities

During the EM blank calibrations in CNES (Toulouse), the AIT/AIV CNES team will be in charge to ensure the production of L1 uncalibrated science data files, using the local instance of the ROC-SGSE. Resulting CDF format files will have to be then sent to the LESIA using the dedicated exchange file interface (SEF).

During the thermal calibration on PFM in LESIA (Meudon), the ROC team will produce itself the L1 uncalibrated science data from its ROC-SGSE instance. This instance will be deployed on its production server at LESIA. Moreover it shall have a direct access to the CNES MEB GSE database in order to be able to retrieve test log data produced during calibrations.

The L2R/L2S science calibrated data production will be ensured by the sub-system teams themselves, from the L1 data files provided by the ROC. Data will have to be available within a short time - less than 72h - to the RPW consortium for analysis.

At the end all of the processed science data files will be archived at the LESIA.

4.5.5.2 Science data quality validation responsibilities

The ROC team does not plan to validate the science data quality during the ground calibrations. The primary validation of these data will have to be performed directly by the sub-system teams involved.

4.5.5.3 Science data archiving responsibilities

All of the relevant data produced during the ground calibration campaigns, including stand-alone, must be archived at the ROC. It includes the following data products:

- LZ, L0, L1 and HK data files produced by the ROC-SGSE
- L1R and L2S data files produced by the RCS
- CDF skeleton tables and calibration tables used to produce the data, as well as the associated documentation



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 32 / 58 -

4.5.6 ROC software logistics support responsibilities

4.5.6.1 At LESIA and Paris Observatory

It has been decided that the computer services of the LESIA (GIGL) and of the Observatoire de Paris (DIO), will have in charge to supply facilities to the ROC for the following needs:

- Hosting, maintaining and ensuring the system administration of the ROC hardware equipment, i.e., servers and disks. In practice, ROC servers and disks will be integrated into the LESIA computer infrastructure.
- Ensuring the availability of the network and accesses, which permit data exchange between the ROC equipment and with the external servers (i.e., intranet/internet accesses, NFS-like mounting systems, SFTP/HTTPS servers).
- Maintaining up-and-running collaboration tools (Web page, Git/SVN servers, Wiki page, mailing lists, etc.)
- Supporting ROC team in the use of logistics (e.g., telecon/visiocon system usage).
- Providing a helpdesk to the ROC team in case of assistance or problems. Especially, the GIGL and DIO shall be able to promptly recover these facilities in case of failure.

The following table summarizes the list of equipment and the computer service in charge of the maintenance/assistance. Details about the concerned ROC device can be found in the section. Obsolete equipment is written in grey.

ROC facilities	Service in charge of the maintenance
Servers	GIGL
Data disks	GIGL
Communications (intranet/internet)	GIGL (at LESIA level) and DIO (at the Observatory level)
RPW Web portal	GIGL
ROC Confluence/JIRA site	GIGL
ROC SVN repository	GIGL
ROC Gitlab server	DIO
ROC mailing lists	DIO
Telecon system	GIGL
Visiocon	GIGL

Table 10. Logistics support to the ROC at LESIA and Paris Observatory.

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).



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 33 / 58 -

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 “ROC Engineering Guidelines for external User” (REGU) document [RD17].

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 [RD17].

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 11. 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	First issue	To be delivered with the RCS “E2E test” 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 12. 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 *development* 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 [RD16].

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).



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 35 / 58 -

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 [RD16].

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 document [RD17]. Nevertheless, the primary instance of the LLVM for RPW will be run at the SOC. And the ROC team shall plan to deploy



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 36 / 58 -

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 [RD16].

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 dedicated REG document [RD16].

An extension document “ROC Engineering Guidelines for External Users” (REGU) document [RD17] 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 [RD17], the convention proper to the LLVM is defined in [RD10]; otherwise the convention to apply for RSS software must be defined in [RD16].

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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 37 / 58 -

- 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 document [RD16], 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 “ROC Software Product Assurance Plan” document (SPAP) [AD4].

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.

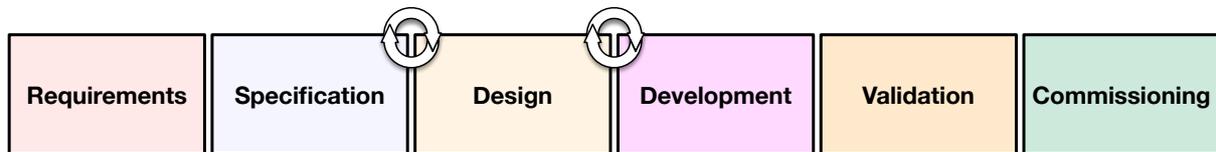


Figure 4. ROC software development life cycle.

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 [AD4].

Software development step	Expected review/key points/events
<i>Requirements</i>	N/A
<i>Specification</i>	N/A



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 38 / 58 -

<i>Design</i>	Preliminary design key point
<i>Development</i>	End of design key point (to take account of changes in the specification and planning)
<i>Validation</i>	ROC software validation campaign kick-off meeting at the beginning of the step / ROC software validation campaign review board at the end.
<i>Commissioning</i>	ROC commissioning review board

Table 13. 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 programming language shall:

- Use JavaScript ES6 version or higher
- Since there is no Follow the syntax standard defined in [RD16].

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.

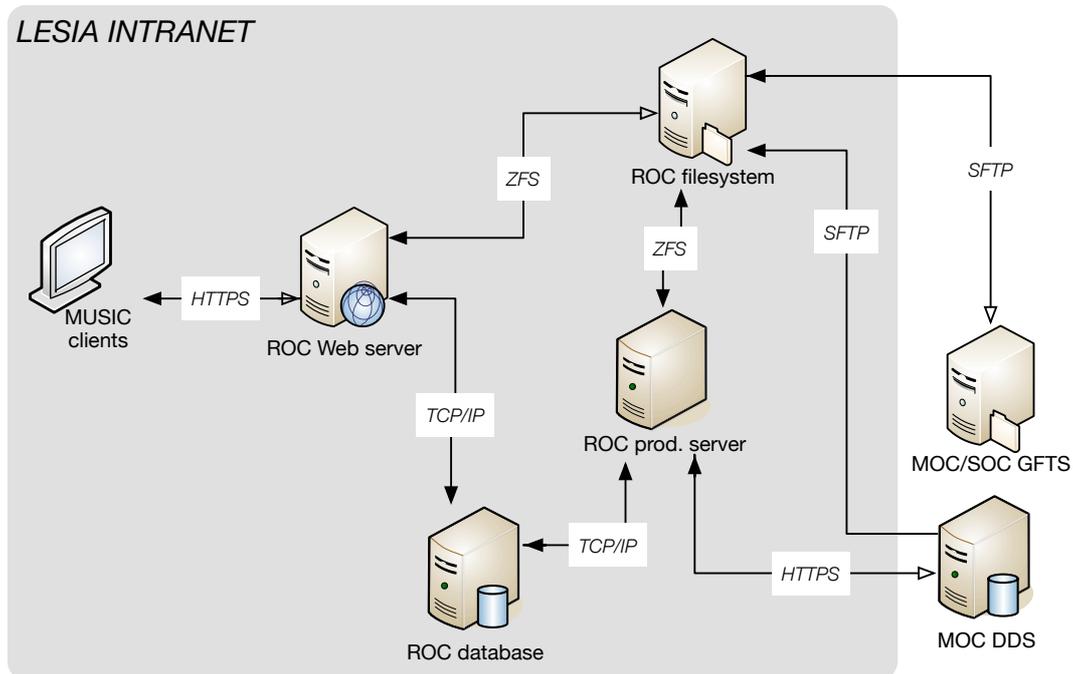


Figure 5. RSS infrastructure overview at LESIA.

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 section 4.5.6, 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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 41 / 58 -

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 comfortable 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 servers

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

Server name	Function	Configuration	Access privileges
<i>roc.obspm.fr</i>	The “production server” is used to run the primary instance of the RODP in prod.	<ul style="list-style-type: none"> • 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.
<i>roc-dev.obspm.fr</i>	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	<ul style="list-style-type: none"> • 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.
<i>roc-web.obspm.fr</i>	The “web server” is used to host the MUSIC tool	<ul style="list-style-type: none"> • A x86 Intel architecture VM • Inter(R) Xeon(R) 2.4 	The administration of this server is done by the GIGL.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 42 / 58 -

		GHz <ul style="list-style-type: none"> Linux Debian Jessie OS. 871 GBytes (shared memory)	Only accessible to the ROC developer team through SSH.
<i>damnancay.obspm.fr</i>	The damnancay server is used to host, test and run the RPW LLVM instances.	<ul style="list-style-type: none"> Same than the EUI LLVM 	The administration of this server is done by the GIGL. Only accessible to the ROC LLVM developers through SSH.
<i>roc-nfsvm</i>	VM used by the LLVM to test the NFS mounting. This VM is hosted on damnancay.	<ul style="list-style-type: none"> Same than the EUI NFS testing VM 	
<i>rocpc1.obspm.fr</i>	This is the server used by the ROC to host and run the MEB GSE tools	<ul style="list-style-type: none"> Windows 7 	The administration of this server is done by the ROC developer team.
<i>arietis.obspm.fr</i>	The <i>arietis.obspm.fr</i> server hosts the RPW Web page at the LESIA.		The administration of this server is done by the GIGL, with specific accesses for the ROC team in order to edit the RPW Web page through the SPIP ³ framework.
<i>lesia11.obspm.fr</i>	The “data storage” server is used to reach the LESIA data disks, which store the ROC data. 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	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.

³ <http://www.spip.net/>



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 43 / 58 -

	private/public directories.		
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	TBD (at least 32 TB over 10 years)	The administration of this server is done by the DIO.
<i>bdd-lesia.obspm.fr</i>	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 monitors, in order to view the RPW activity during the mission. This equipment shall be available 24h, in the RPW operations room at LESIA.	<ul style="list-style-type: none"> Minimal configuration to display the MUSIC frontend from a client (i.e., Web browser) 	The maintenance and administration of this equipment will be under the GIGL responsibility. The ROC system administrators shall ensure the MUSIC client is still up-and-running
<i>gitlab.obspm.fr</i>	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.
<i>confluence-lesia.obspm.fr</i>	Hosts the ROC confluence site.		The administration of the server is done by the GIGL.
<i>jira-lesia.obspm.fr</i>	Hosts the ROC JIRA site.		The administration of the server is done by the GIGL.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 44 / 58 -

Table 14. 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	Description	Development	Execution server	Comment
-----------	-------------	-------------	------------------	---------



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 45 / 58 -

account		server		
<i>roc_rdop</i>	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
<i>roc_sgse</i>	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
<i>roc_rllp</i>	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
<i>rocuser</i>	Technical account used to have a generic read-only access to the Git repositories of the ROC			

Table 15. 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
<i>RODP</i>	Python 3 + Cython	numpy, SQLAlchemy
<i>LLVM</i>	Linux BASH, Python 3 + Cython	numpy, SQLAlchemy



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES
 Issue: 02
 Revision: 03
 Date: 17/11/2017

<i>MUSIC</i>	Python 3 + Django framework (backend) + Javascript (Frontend)	numpy, matplotlib, react
<i>ROC-SGSE</i>	Python 3 + Cython + PyQt5 (for the TV-SGSE)	numpy, matplotlib, SQLAlchemy
<i>SimuSBM1 Software</i>	IDL	
<i>SimuSBM2 Software</i>	IDL	
<i>THR CALBAR</i>	IDL	
<i>TDS CALBA</i>	IDL	
<i>SCMCAL</i>	IDL	
<i>LFR CALBUT</i>	Python 3	
<i>BICAS</i>	Matlab 5	

Table 16. 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
<i>ROC-SGSE Test Database (TDB)</i>	Database of the ROC-SGSE	PostgreSQL
<i>ROC Instrument Database (IDB)</i>	ROC instance of the RPW IDB	PostgreSQL
<i>ROC Mission Database (MDB)</i>	Central database of the ROADS	PostgreSQL
<i>MEB SGSE Database</i>	MEB GSE database	MySQL
<i>RPW Instrument Database (IDB)</i>	RPW IDB	XML (LESIA PALISADE format) / PostgreSQL (MOC MiB format)
<i>Mission information Base (MiB)</i>	Database used by the MOC to perform the in-flight operations. It contains a copy of the RPW IDB.	PostgreSQL

Table 17. 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 [AD4].



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 47 / 58 -

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

Table 18. 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 *pytest*, *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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 48 / 58 -

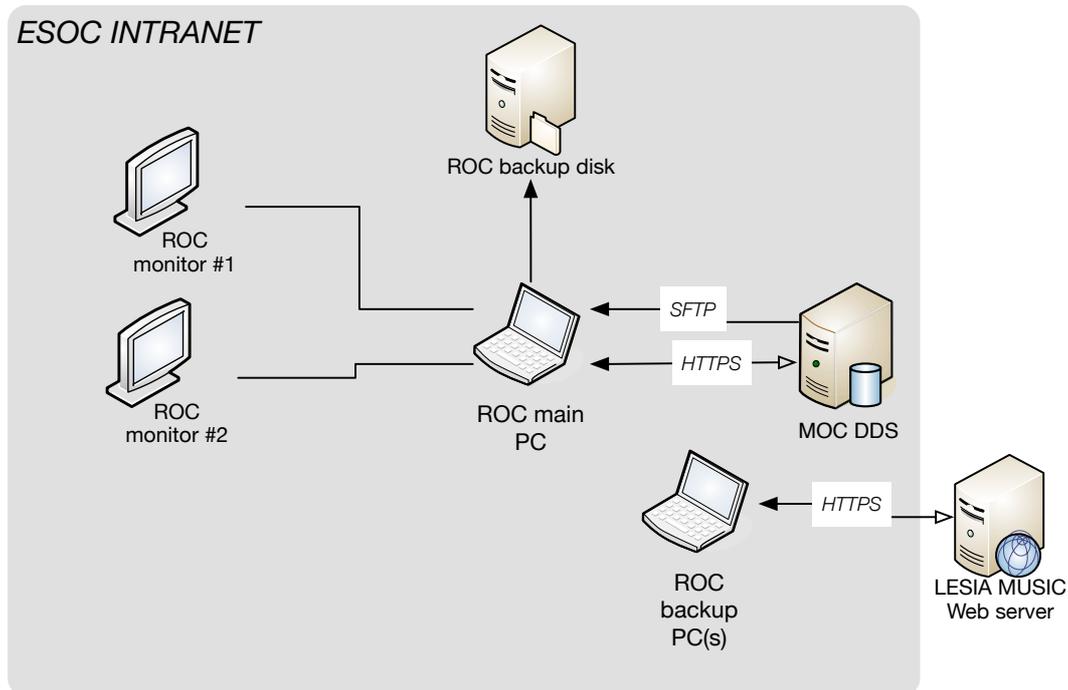


Figure 6. RSS infrastructure at ESOC.

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 comfortable 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 RSS 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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 49 / 58 -

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: <ul style="list-style-type: none"> • Processor: 2.3 GHz • RAM: 16 Go • Volume: 1To 	
TBD	First backup computer, in case of main computer failure	<ul style="list-style-type: none"> • At least same config. than the main computer 	
TBD	Second backup computer, in case of first backup computer failure	<ul style="list-style-type: none"> • At least same config. than the main computer 	
TBD	External data disk to backup data generated by the MOC RSS	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • At least 22'' size screen 	
TBD	Second external monitor to be connected to the main computer	<ul style="list-style-type: none"> • At least 22'' size screen 	

Table 19. 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.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 50 / 58 -

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.

The next sections give only the main delivery schedule.

6.2 ROC software main deliveries schedule

Table below gives the main deliveries of the ROC software products. The delivery schedule dependencies must be consistent with the ROC dev. planning file content.

Software deliverable	Main available capabilities	Software validation leader	Developer(Developer(s)) in charge	Delivery schedule dependencies
ROC GSE				
SimuSBM1 software V1	Version for the SBM1 detection RPW DPU software validation on-ground campaign	Xavier Bonnin (ROC)	- Xavier Bonnin - Oksana Kruparova	Prior to the RPW DPU software SBM1 algorithm



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 51 / 58 -

				validation campaign on-ground
SimuSBM1 software V2	Second version to be used for SBM1 detection algorithm validation and optimization in-flight.	Xavier Bonnin (ROC)	- Antonio Vecchio	Prior to the RPW DPU software SBM1 algorithm validation campaign in-flight
SimuSBM2 software V1.0.0	Version for the SBM2 detection RPW DPU software validation on-ground campaign	Xavier Bonnin (ROC)	- Xavier Bonnin - Milan Maksimovic - Olga Alexandrova	Prior to the RPW DPU software SBM2 algorithm validation campaign on-ground
SimuSBM2 software V2.0.0	Second version to be used for SBM2 detection algorithm validation and optimization in-flight.	Xavier Bonnin (ROC)	- Antonio Vecchio	Prior to the RPW DPU software SBM2 algorithm validation campaign in-flight
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	Prior to the ROC-SGSE V1 development
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	Prior to the EM2 blank tests calibration campaign at CNES
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)	Prior to the PFM thermal calibration campaign at LESIA
ROADS				
RSS "E2E test" version	Preliminary version, ready for the MOC/SOC SOV-0 E2E	Xavier Bonnin	- Quynh Nhu Nguyen	Prior to the ESA SOV-0



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 52 / 58 -

datapackage	<p>test, including:</p> <ul style="list-style-type: none"> - Partial MUSIC Web tool capabilities (Procedures/IOR/MDOR/PDO R edition, assertion and submission ready, E-FECS, TMC data plotting, TM/TC/HK/L1 visualization) - Full RPW LLVM expected capabilities - SOC/MOC GFTS interface up and running - ROC-SGSE V2 <p>The datapackage shall include the following documents:</p> <ul style="list-style-type: none"> - 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) 	(ROC)	<ul style="list-style-type: none"> - Sonny Lion - Aichatou Boubacar 	“E2E2 test campaign
RSS “ready for flight” version datapackage	<p>Full validated version, ready for the commissioning and cruise phase operations, including:</p> <ul style="list-style-type: none"> - Full processing of the RPW data products - Full MUSIC Web tool capabilities, except for SISSI. - SOC/MOC DDS/GFTS interfaces up and running - Advanced draft of the ROC User Manual and Reference Guide (all functionalities described, without SISSI tool) 	Xavier Bonnin (ROC)	<ul style="list-style-type: none"> - Quynh Nhu Nguyen - Sonny Lion - Aichatou Boubacar 	After the ROC software validation campaign and prior to the RPW ground segment acceptance review by the CNES
RSS “fully operational” version datapackage	<p>Fully operational ROADS, including:</p> <ul style="list-style-type: none"> - Full MUSIC tool expected capabilities - Full RODP tool expected capabilities - Full LLVM expected capabilities <p>To be delivered at the end of the commissioning phase.</p> <ul style="list-style-type: none"> - First issue of the ROC User Manual and Reference Guide (all functionalities covered) 	Xavier Bonnin (ROC)	<ul style="list-style-type: none"> - Quynh Nhu Nguyen - Sonny Lion - Aichatou Boubacar 	After the validation of the ROADS during the in-flight commissioning



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 53 / 58 -

RPW Low Latency Virtual Machine (LLVM) V0.1.0	First “Hello World” VM configuration and test version.	Richard Carr (SOC)	- Xavier Bonnin	First version, according to the LLVM delivery schedule
RPW Low Latency Virtual Machine (LLVM) V0.X.0	Second “processing data” version. Delivered with an up-to-date LLDDD and “Testcard” scenario data LL01 CDF.	Richard Carr (SOC)	- Sonny Lion	After the “Hello World” version, and prior to the testcard scenario test
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	After the “processing” version
RCS – THR CALBAR				
THR_CALBAR preliminary version	Software version being able of producing ROC-SGSE THR preliminary L2S CDF files. Delivered with the THR L2S master CDF files.	Milan Maksimovic (LESIA)	- Antonio Vecchio - Quynh Nhu Nguyen	Prior to PFM thermal calibrations
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	After the preliminary version and prior to the ROADS “E2E test” version delivery
THR_CALBAR “ready-to-flight” version	Software version being able of producing SoIO 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	After the “E2E test” version and prior to the ROC software validation campaign
THR_CALBAR “fully operational” version	Fully validated version of the software after commissioning.	Milan Maksimovic (LESIA)	- Lorenzo Matteini - Quynh Nhu Nguyen	After the validation during the in-flight commissioning and prior to the ROADS “fully operational” version



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 54 / 58 -

				delivery
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	Prior to PFM thermal calibrations
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.			After the preliminary version and prior to the ROADS “E2E test” version delivery
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.			After the “E2E test” version and prior to the ROC software validation campaign
LFR_CALBUT “fully operational” version	Fully validated version of the software after commissioning.	Thomas Chust (LPP)	- Bruno Katra, Rodrigue Piberne	After the validation during the in-flight commissioning and prior to the ROADS “fully operational” version delivery
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	Prior to PFM thermal calibrations
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	After the preliminary version and prior to the ROADS “E2E test” version delivery



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 55 / 58 -

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.			After the “E2E test” version and prior to the ROC software validation campaign
TDS_CALBA “fully operational” version	Fully validated version of the software after commissioning.			After the validation during the in-flight commissioning and prior to the ROADS “fully operational” version delivery
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	Prior to PFM thermal calibrations
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	After the preliminary version and prior to the ROADS “E2E test” version delivery
BICAS “ready-to-flight” version	Software version being able of producing Solo RPW 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.	Andris Vaivads	- Erik Johansson	After the “E2E test” version and prior to the ROC software validation campaign
BICAS “fully operational” version	Fully validated version of the software after commissioning.	Andris Vaivads	- Erik Johansson	After the validation during the in-flight commissioning



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 56 / 58 -

RCS – SCMCAL				
				g and prior to the ROADS “fully operational” version delivery
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	Prior to PFM thermal calibrations
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	After the preliminary version and prior to the ROADS “E2E test” version delivery
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	After the “E2E test” version and prior to the ROC software validation campaign
SCMCAL “fully operational” version	Fully validated version of the software after commissioning.	Matthieu Kretzschmar	- Jean-Yves Brochot - Gamil Cassam-Chenai	After the validation during the in-flight commissioning and prior to the ROADS “fully operational” version delivery

Table 20. ROC software main deliveries schedule.



ROC Software Development Plan

Ref: ROC-GEN-SYS-PLN-00015-LES

Issue: 02

Revision: 03

Date: 17/11/2017

- 58 / 58 -

8 DISTRIBUTION LIST

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

INTERNAL

LESIA CNRS	x	M. MAKSIMOVIC
	x	Y. DE CONCHY
	x	X. BONNIN
	x	QN NGUYEN
		B. CECCONI
	x	A. VECCHIO
	x	S. LION
	x	A. ABOUBACAR

LESIA CNRS	x	N. FULLER

EXTERNAL (To modify if necessary)

CNES		C. FIACHETTI
	x	E. BELLOUARD
		R.LLORCA-CEJUDO
		E.LOURME
		M-O. MARCHE
	x	E.GUILHEM
	x	E. LE DE
	x	M. ROUZE
	x	J-M. TRAVERT
	x	D. RAULIN
IRFU		L. BYLANDER
		C.CULLY
		A.ERIKSSON
	x	E.JOHANSSON
	x	A.VAIVADS
LPC2E	x	Y. KHOTYAINTSEV
	x	J.-Y. BROCHOT
		G. JANNET
		T.DUDOK de WIT
SSL	x	M. KRETZSCHMAR
	x	G. CASSAM-CHENAI
		S.BALE

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