SMOS L1 Processor Prototype User Manual

Code : SO-UM-DME-L1PP-0016
Issue : 2.18
Date : 29/12/12

	Name	Function	Signature
Prepared by	A. Gutiérrez	Project Engineer	
	R. Castro	Project Engineer	
	P. Vieira	Project Engineer	
Checked by	J. Barbosa	Quality A. Manager	
Approved by	J. Barbosa	Project Manager	

DEIMOS Engenharia
Av. D. João II, Lote 1.17, Torre Zen, 10º
1998-023 Lisboa, PORTUGAL
Tel: +351 21 893 3017
Fax: +351 21 896 9099
E-mail: mailto:deimos@deimos.com.pt

This page intentionally left blank

Document Information

Contract Data	Classification
Contract Number: 4000101241/10/I-AM	**Internal** ☐ **Public** ☐ **Industry** ☐ **Confidential** ☑
Contract Issuer: ESA	

Internal Distribution		
Name	Unit	Copies

External Distribution		
Name	Organisation	Copies
Jean-Claude Debruyn	ESA	1
Steven Delwart	ESA	1

Archiving	
Word Processor:	MS Word 2000
File Name:	SO-UM-DME-L1PP-0016-L1PP-Software-User-Manual.doc~~SO-UM-DME-L1PP-0016-L1PP-Software-User-Manual~~
Archive Code:	SO-UM-DME-L1PP-0016-L1PP

Document Status Log

Issue	Change description	Date	Approved
1.0	Table of Contents, FAT v0	2005-05-05	
1.1	Released for CDR, after internal review	2005-06-30	
1.2	Released after CDR, incorporated RIDs	2005-08-31	
1.3	Updated for Final L1PP Release	2006-06-07	
1.4	Updated after the OSAT. Reviewed and completed with Breakpoint information.	2006-07-20	
2.0	Updated for V2R release of L1PP	2006-11-17	
2.1	Updated for V3R release of L1PP	2007-04-09	
2.2	Updated for version 3.5 of L1PP	2007-07-17	
2.3	Updated for V4R release of L1PP	2007-11-26	
2.4	Updated for L1PP v1.4.1	2008-01-25	
2.5	Added description for DPGS settings in GUI Added swap partition advice Updated after comments from ESA	2008-02-11	
2.6	Updated for L1PP v1.5	2008-03-31	
2.7	Updated for L1PP v1.6	2008-07-25	
2.8	Updated for L1PP v1.6.1	2008-10-22	
2.9	Updated for L1PP v2.0.0	2008-12-12	
2.10	Updated for L1PP V3.1.0	2009-05-15	
2.11	Updated for L1PP V2.2.0	2009-07-24	
2.12	Updated for L1PP V3.2.0	2009-09-07	
2.13	Updated for L1PP V3.3.0	2010-03-26	
2.14	Updated for L1PP V3.4.0	2010-05-31	
2.15	Updated after review for the Maintenance Phase and L1PP V3.5.0	2010-10-29	
2.16	Updated for L1PP V5.0.0 - Moved L1 algorithm configuration fields to CNFL1P. L1PP configuration is now restricted to orchestration and prototype algorithms that are not L1 baseline	2011-05-20	
2.17	Updated for L1PP V5.5.0	2011-11-29	
2.18	Updated for L1PP V6.0.0	2012-11-29	

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. Purpose and Scope

This purpose of this document is to provide to the user all the information needed for installing and running the SMOS L1 Processor Prototype (L1PP) v6.0.0. This User Manual provides the following information:

- Installation Steps;
- Configuration Procedures;
- Description of the functionalities and L1PP usage;
- Limitations and known bugs;
- Description of Test Tools for data generation and results analysis.

This document was produced in the scope of the project "SMOS Level 1 Processor Prototype Development".

1.1.1. Acronyms and Abbreviations

ADF	Auxiliary Data Files
API	Application Programming Interface
APID	Application program identifier
CFI	Customer Furnished Item
COTS	Commercial Off-The-Shelf
DPM	Data Processing Model
EE	Earth Explorer
EEFH	Earth Explorer File Handling CFI (ASCII XML library)
EM	Engineering Model
FWF	Fringe Wash Function
GUI	Graphical User Interface
HKTM	HouseKeeping Telemetry
HTML	HyperText Markup Language
ISO	International Organization for Standardization

IVT	Image Validation Test
L1PP	Level 1 processor prototype
LCF	LiCeF (LIghtweight and Cost-Effective Front-end)
MIRAS	Microwave Imaging Radiometer with Aperture Synthesis
NIR	Noise Injection Radiometer
OBET	On Board Elapsed Time
PLM	PayLoad Module
PMS	Power Measurement Signal
PUS	Packet Utilization Standard
SEPS	SMOS End-to-end Performance Simulator
SMOS	Soil Moisture and Ocean Salinity
SVP	Software Validation Plan
TBW	To Be Written
UPC	*Universitat Politècnica de Catalunya (*Technical University of Catalonia*)*
XML	Extended Markup Language
XSL	eXtensible Stylesheet Language

Table 1: Table of Acronyms.

For the complete list of acronyms, please refer to the document SO-LI-CASA-PLM-0094 “Directory of Acronyms and abbreviations” [RD.5].

1.2. Applicable and Reference Documents

1.2.1. Applicable Documents

Ref.	Code	Title	Issue *Date*
AD.2	EE-MA-DMS-GS-0008-3-7-2_080731	EE XML/Binary CFI File Handling Library User Manual	*3.7.3* *07/05/10*
AD.3	SO-DS-DME-L1PP-0007	SMOS L1 Processor L0 to L1a Data Processing Model	*2.165* *29/11/124*

Ref.	Code	Title	Issue *Date*
AD.4	SO-DS-DME-L1PP-0008	SMOS L1 Processor L1a to L1b Data Processing Model	2.165 *29/11/121*
AD.5	SO-DS-DME-L1PP-0009	SMOS L1 Processor L1b to L1c Data Processing Model	2.110 *29/11/121*
AD.6	BinX	Editkt::BinX 1.2 Developer's Guide	1.2
AD.7	SO-TR-DME-L1PP-0018	SMOS L1 Processor Software Verification and Validation Plan (SVVP)	2.121 *29/11/121*
AD.10	SO-DS-DME-L1PP-0006	SMOS L1 System Concept	2.9 *29/10/10*
AD.11	SO-TDD-DME-L1PP-027262	L1PP Test Data Set Description	1.0 *29/11/121*
AD.12	SO-TN-IDR-GS-0005	SMOS Level 1 and Auxiliary Data Products Specification	5.242 *29/11/121*
AD.13	SO-SOW-ESA-GS-6647	SMOS Expert Support Laboratories for the period 2010-2014 - ESL Level 1 Calibration and Reconstruction	1.2 *07/05/10*

Table 2: Applicable Documents.

1.2.2. Reference Documents

Ref.	Code	Title	Issue *Date*
RD.1	EE-MA-DMS-GS-0001	Earth Explorer Mission CFI Software MISSION CONVENTIONS DOCUMENT	1.5 *(07/05/10*
RD.2	PE-TN-ESA-GS-0001	Earth Explorer Ground Segment File Format Standard	1.4 *13/06/03*
RD.3	EE-MA-DMS-GS-0002	Earth Explorer Mission CFI Software GENERAL SOFTWARE USER MANUAL	3.7.3 *07/05/10*
RD.4	EE-MA-DMS-GS-0008	EXPLORER FILE HANDLING Reference Manual	3.7.3

Ref.	Code	Title	Issue *Date*
RD.5	SO-LI-CASA-PLM-0094	Directory of Acronyms and abbreviations	
RD.6	SO-TN-DME-L1PP-0169	SMOS L1 Processor L1b Refactoring	1.3 *25/07/08*

Table 3: Reference Documents

2. SMOS L1 PROTOTYPE GUIDE

This chapter presents all the information needed by the user in order to understand the objective and the functioning of the L1PP v6.0.0. The chapter first introduces the L1PP application, presenting summarily its objectives and components. Then the installation, configuration usage and tuning procedures are detailed.

2.1. Objectives

The purpose of the SMOS L1PP is to convert the MIRAS instrument outputs into Brightness Temperature measurements, geolocating them and providing observation angles and additional parameters.

The prototype receives as input SMOS Level 0 Products as well as Auxiliary Data Files (ADFs), processes them in several steps and generates as output Level 1A (SMOS reformatted and calibrated Observation and Housekeeping data in engineering units) , Level 1B (output of the image reconstruction of the SMOS observation measurements) and Level 1C (swath-based maps of Brightness Temperature) products. The L1PP is a data driven application, being able to ingest and process, in addition to the L0 products, the L1A and L1B products.

L1PP v6.0.0 supports the operational DPGS V3 format, described in [AD.12].

2.2. Components

L1PP includes different modules, responsible for the execution of the following tasks:

- Control the data flow and the processing sequence (Orchestrator);
- Perform the scientific processing of the data (Processing Units);
- Perform Read/Write and cache management operations (Core Components);
- Provide a Graphical Interface in order to allow the user to easily configure, run and analyse the results of L1PP (Graphical User Interface).

A brief description of each module is presented in the following sub-sections[1].

[1] For further details the user shall refer to the following documents: "*SMOS L1 System Concept*" [AD.10], Data Processing Model L1a [AD.3], Data Processing Model L1b [AD.4] and Data Processing Model L1c [AD.5].

2.2.1. Orchestrator

Orchestrator manages the working flow of the L1 Processor prototype. It continuously polls for new input data (L0, L1a and L1b data products) and then, according to the type of the file, it invokes the corresponding processing unit.

Orchestrator is written in ISO C99 and is linked with Processing Units, Core Components and Earth Explorer CFI libraries [RD.3].

2.2.2. Processing Units

The processing units contain the modules identified in the System Concept document and are responsible for the tasks associated with Error Correction/calibration (L1a), Image Reconstruction (L1b) and Geolocation (L1c) of the MIRAS Instruments measurements.

The Error Correction module converts HKTM raw data into engineering units, determines the calibration parameters from products generated in calibration mode and calibrates the science data products.

The Image Reconstruction receives as input the L1a products, corrects the influence of different foreign sources, such as Sun/Moon effects, on the acquired images and reconstructs the images on the Antenna Reference Frame.

Finally the Geolocation module receives as input the reconstructed images on the Antenna Reference Frame and geolocates them after performing a Ionospheric correction. As of L1PP v1.5, the Geolocation module has been improved by using OpenMP thread distribution. This means that L1PP is able now to take advantage of multi-CPU machines and improve the time performances required of L1c processing.

Processing units are written in ISO C99 and are linked with Core Components and Earth Explorer CFI libraries [RD.3].

2.2.3. Core Components

The Core Components provide a set of libraries that may be used by all components of the processor prototype. These libraries include functions for:

- Reading and Writing hybrid (ASCII XML Header + Binary Data Block), DPGS V3 products and XML files;
- Load into and get from cache science and auxiliary data;
- Performing mathematical calculation over complex numbers, vectors, etc;
- Logging information regarding the execution of the processor.

Core components are written in ISO C99 and are linked with the BinaryXML File Handler CFI, the Earth Explorer CFI and Indra's XML RW API.

2.2.4. Graphical User Interface

The Graphical User Interface (GUI) allows the user to configure the prototype, start/stop the prototype, view the log and exit the application. The GUI is written in JAVA and is build independently from all the remaining components.

2.3. Installation guide

The following sections describe the steps necessary for installing L1PP: check Hardware requirements, install external libraries, execute the installation procedures and set environment variables.

2.3.1. Hardware Requirements

L1PP may be run in a Pentium IV 64 with LINUX installed. The memory and disk resources needed for executing the prototype depend on the type of algorithms being used. For full functionality, the user needs the following resources:

- 18,5 GiB of Disk Space – from these 18,5GiB, 16GiB are needed for generating the G-Matrix ADF[2]. After the generation of the ADF, the L1PP needs around 10GiB of disk space for all the ADFs, libraries and some available space for products generation;Minimum 4GiB RAM - the RAM memory available will be critical when handling the G-Matrix ADF (Level 1B). If the user wants to use the G-Matrix ADF for performing the Foreign Sources Correction, a minimum of 4GiB is advisable for dual pol (8 GiB recommended), and a minimum of 10GiB is advisable for full pol (16GiB recommended). If the user has less than 8 GiB of RAM, a swap partition of at least 8 GiB is also recommended as the total amount of memory needed in L1b processing may reach 10GiB for a full pol product half-orbit. A possibility to reduce the RAM demand in full pol processing is to disable the cross-polarisation correction in L1PP.

- From L1PP v5.5.0 onwards there is the possibility to generate and use an expanded G-Matrix. This implies that the user must have 41 GiB of disk space, since the generation of this expanded matrix requires ~38 GiB. The expanded matrix needs a minimum of 19 GiB of RAM memory.

2.3.2. Dependencies

In order to run the SMOS L1PP, it is recommended to have the following software already installed in the target platform:

[2] The ADF occupies around 8GB of disk space. However, since it is not provided with the Install Kit, it must be generated by the prototype (refer to section 2.4.4 for further details on generating the G-Matrix ADF). For generating it a minimum of 2X8GB of disk space available is needed. After the generation of the ADF, only the 8GB occupied by the ADF are needed.

- gcc 4.3 (http://gcc.gnu.org);
- glibc 2.3.4 (http://gcc.gnu.org);
- gfortran (http://gcc.gnu.org included in gcc 4.3);
- BinaryXML File Handler 3.7 (already includes BinX 1.2.6) (http://www.smos.esa.int/);
- xerces 2.8.0 (http://xml.apache.org/xerces2-j/);
- XML RW API v04.01.05 (ftp:// 131.176.251.166/smos/software/XML_RW_API/)
- xerces 2.7.0 (http://xml.apache.org/xercesc/) (needed by XML RW API)
- DOM4J 1.5.2 (http://www.dom4j.org);
- Jaxen 1.1 (http://jaxen.codehaus.org);
- Swing 1.0;
- Log4c 1.2.0 (http://log4c.sourceforge.net);
- Java J2SE SDK 1.5.0 (http://java.sun.com/j2se/1.5.0/download.html);
- Lapack 3.0 (http://www.netlib.org/lapack/);
- Blas (version included in Lapack 3.0);
- FFTW 3.1.2 (http://www.fftw.org/).

Each product referred above should be installed according to its own instructions in the target platform.

Nevertheless, the L1PP Install Kit already contains some of the above dynamic libraries in case the user has not installed them locally. The included libraries are:

- BinaryXML 3.7;
- xerces 2.8.0 (http://xml.apache.org/xerces2-j/);
- XML RW API v04.01.05 (ftp://-131.176.251.166/smos/software/XML_RW_API/)
- xerces 2.7.0 (http://xml.apache.org/xercesc/) (needed by XML RW API)
- Libxml2 2.6.16;
- Log4c 1.2.0;
- DOM4J 1.5.2;
- Jaxen 1.1;

- Swing 1.0;
- Lapack 3.0;
- Blas (version included in Lapack 3.0);
- FFTW 3.1.2.

Libraries *dom4j.jar*, *jaxen-1.1-beta-8.jar* and *swing-layout-1.0.jar* are provided in *l1pp/lib* directory (no extraction or installation required).

The rest of the libraries may need to be compiled as appropriate, so it may be possible that they need to be installed as root. Please check each installation instructions for this purpose, although for GNU packages it can be done simply by executing *./configure* and then *make*.

2.3.3. Installation Kit Description

The L1 Processor Prototype is provided as an InstallKit composed of a single tgz file. This binary distribution is delivered as L1PP-6.0.0_Installer.tgz which after decompressing contains:

- *InstallKittL1PP_6_0_0.readme*
- *InstallKittL1PP_6_0_0.tgz*: which contains the libraries and configuration files of the L1 prototype. The package contains inside a pre-configured structure of the L1PP working directory with the Linux 64 bits version and Mac OS X of the L1PP. All required configuration files, product schemas and header templates are also included;
- *InstallKittL1PP.sh*
- *InstallKittL1PP_DebianSystem.sh*

In order to run the prototype the user will also need the Auxiliary Data Files and the test data scenarios, which are provided separately:

- *ADFPackage_6_0_0.tgz*: contains all the Auxiliary Data Files needed by the prototype as well as the corresponding xml schemas and documentation;
- *TestDataPackage_6_0_0*: contains several test data scenarios (L0 products), each packed independently, corresponding configuration files in order to be used directly by the prototype. In addition, documents with the test scenarios description and with the product format specification are also provided in the same webpage[3].

2.3.4. L1PP Installation Steps

In order to install the L1PP, the user shall extract the *tgz* file and execute the install script in the command line:

[3] http://www.smos.com.pt/project_data_products.html

$ tar xzvf L1PP-Installer.tgz

$ sh InstallKitL1PP.sh $INSTALL_PATH

Where $INSTALL_PATH is the directory where the user wants to install the prototype. For example, if we want to create it in */home/smos/tools/*, the command to execute shall be:

$ sh InstallKitL1PP.sh /home/smos/tools/

and the script will extract and create a new directory inside *tools* called *l1pp-<version>* with all the L1 Prototype contents (Libraries, configuration files, xml schemas, product headers, etc.). The script will also perform an md5 checksum for all the files contained in the distribution. For a complete list of these files and their checksum, please refer to "InstallKit.md5", provided in the home directory of the distribution.

The installation script shall also update the configuration files being extracted so that the paths referred inside are correctly referenced to the installation directory. This is done automatically by using the script named *scripts/update-config-path.sh*. If at any time there is the need to reuse it with a default configuration file, it can be invoked on its own.

When installing a new test scenario from the provided Test Data Set (found on the L1PP SMOS webpage), the script *scripts/update-config-path.sh* will have to be re-executed for the *configurationFile.xml* and *log4crc* files inside the scenario directory.

After the installation is complete, the directories structure shown in Section 5 has been created by the installation script. Nevertheless, the user must later populate manually the ADF and L0/L1 products that he intends to process into the appropriate directories.

After installing the prototype, the user will need the Auxiliary Data Files contained in *ADFPackage_5_5.tgz*. After extracting the package in the $L1PP_ROOT directory, the ADFs will be automatically located in the proper location, i.e., $L1PP_ROOT/data/adf-dpgs.

2.3.5. Environment Variables

For execution of the prototype, a new environment variable named L1PP_ROOT must be created by the user, pointing to the *l1pp* directory. For instance, if the *$INSTALL_PATH* is */home/smos/tools/* , the L1PP will be installed in */home/smos/tools/l1pp-6.0.0*. Therefore L1PP_ROOT variable must be set by the user as */home/smos/tools/l1pp-6.0.0/* :

$ export L1PP_ROOT=/home/smos/tools/l1pp-6.0.0/

Remark: It is important to highlight that the environment variable $L1PP_ROOT must always terminated with a slash ("/") character. For instance if the L1PP is installed in */home/smos/tools/l1pp-v6.0.0* the
L1PP_ROOT shall be */home/smos/tools/l1pp-v6.0.0/*.

If the user wishes to launch the prototype from the command line, it will also be necessary to update the LD_LIBRARY_PATH environment variable with the location of the shared libraries, as follows:

$ export LD_LIBRARY_PATH=$L1PP_ROOT/lib64/LINUX:$L1PP_ROOT/external_libs/lib64/LINUX:$LD_LIBRARY_PATH

LD_LIBRARY_PATH shall include the directories where the different libraries are installed. Some of the external libraries are provided with the Installation Kit in the directory *L1PP_ROOT/external_libs/lib64*, while others must be installed by the user (see Section 2.3.2). The path for the L1PP internal libraries (*L1PP_ROOT/lib64* – for the 64-bit version) shall also be included in LD_LIBRARY_PATH variable.

Note: If the user changes the location of the libraries, the LD_LIBRARY_PATH shall be updated accordingly. However, the files *dom4j.jar*, *jaxen-1.1-beta-8.jar* and *swing-layout-1.0.jar*, provided in *$L1PP_ROOT/lib* shall always be kept in this directory.

These two variables are updated if the prototype is executed with the scripts provided in the installation package. If the user wants to use a different execution method, these variables must be defined. In order to avoid overriding the user's local libraries, the L1PP will only use the provided additional libraries in case it cannot find them in the user-defined path.

2.4. Usage

The prototype GUI may be started in 64bits by running the script *run-l1pp64.sh*, as follows:

$ sh ./run-l1pp64.sh

This script is provided as part of the installation package and is configured also during installation time. If the user moves the L1PP directory to another location, it shall be required to verify the correctness of the script.

The following output should be displayed upon execution:

```
Set L1PP_ROOT= /home/smostest/l1pp/
Set LD_LIBRARY_PATH=/home/smostest/l1pp//lib:/usr/local/lib/:/home/smostest/l1pp//external_libs/lib
####################
Starting the L1PP GUI
####################
SMOS Level 1 Processor Prototype v6.0.0
Developed by Deimos Engenharia & Critical Software S.A.
Under contract of EADS CASA Espacio and ESA
```

The output above confirms that the L1PP is correctly installed and the binary is running.

Additionally the following window shall be displayed:

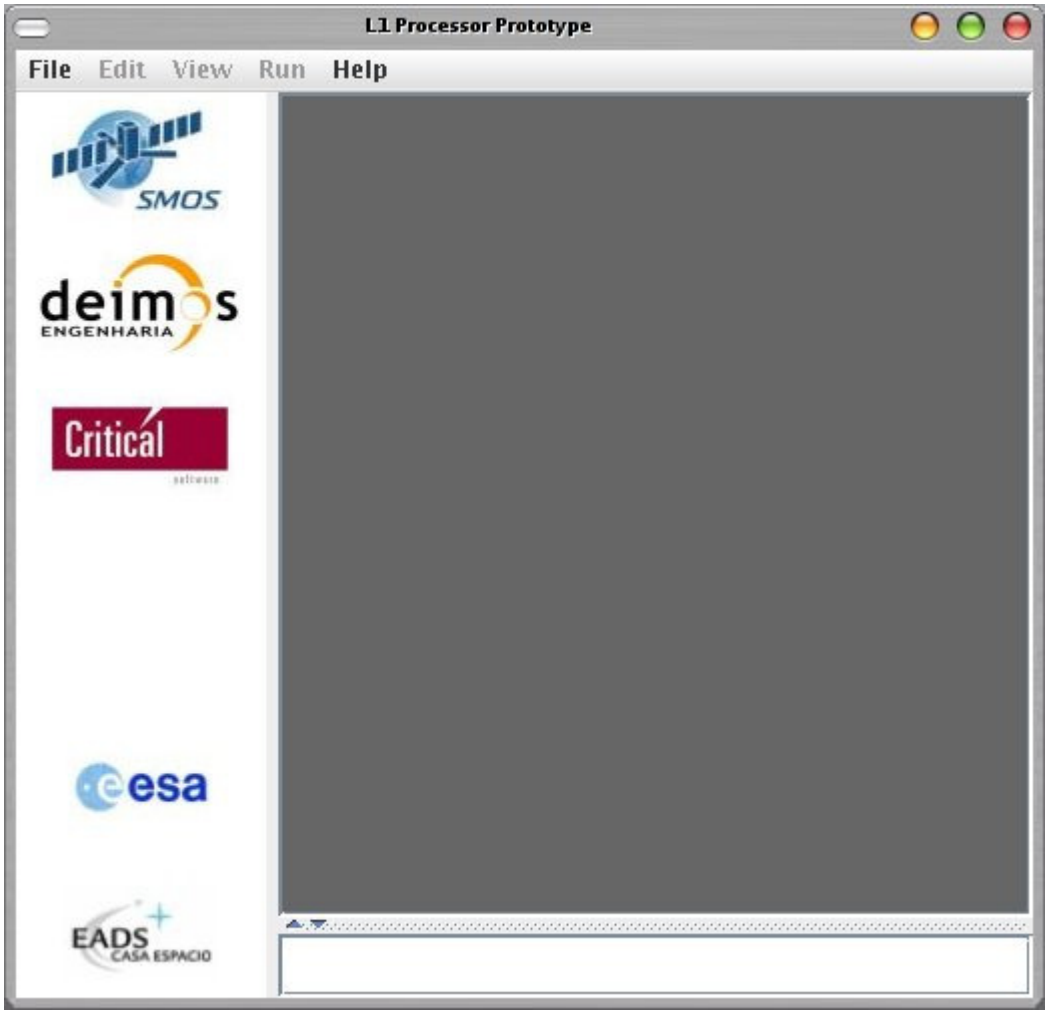


Figure 1: Prototype main window

As mentioned before, the Install Kit contains additional dynamic libraries in case the user does not have them installed, and the execution script will add their path so that the L1PP always finds them. If some of these libraries are installed, but the prototype is not taking them into account during execution, there are two steps to perform:

- Verify the values of variables LD_LIBRARY_PATH as follows:

$ echo $LD_LIBRARY_PATH

The variable LD_LIBRARY_PATH must contain the path to the directories where each of the libraries listed in Section *2.3.2 - Dependencies* are located. If the LD_LIBRARY_PATH does not contain the path to any of the referred libraries then add it to the variable using the following command:

$ export LD_LIBRARY_PATH=[Missing-Library-path]:$LD_LIBRARY_PATH

- Verify the values of variable L1PP_ROOT as follows:

$ echo $L1PP_ROOT

If the variable does not contain the correct value then modify it by typing the following command:

$ export L1PP_ROOT = [Package-main-dir]

Note in order to avoid configuring LD_LIBRARY_PATH and L1PP_ROOT every time a new session is started, the user may add the libraries path to the *.bashrc* file contained in the home directory. For this the user should edit the referred file, adding the missing libraries path.

An example of a *.bashrc* file is provided hereafter:

```
# .bashrc

# User specific aliases and functions

# Source global definitions

if [ -f /etc/bashrc ]; then

        . /etc/bashrc

fi

export LD_LIBRARY_PATH=/usr/local/lib/:/opt/xerces-c/lib:/opt/cfi/aux_tools/libxml/LINUX/lib:/opt/binxml-
fh/lib:/home/jreis/workspace/l1pp_cpp/lib

export L1PP_ROOT=/home/smos/l1pp/
```

2.4.1. Graphical User Interface

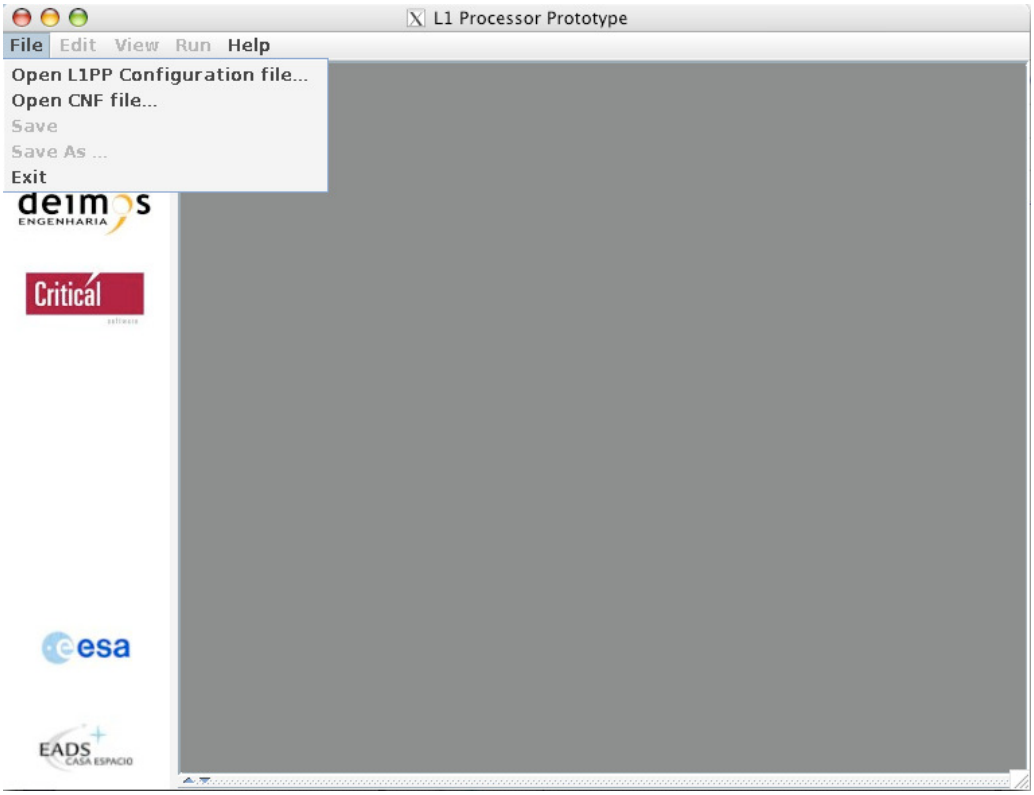
The GUI is a front-end to the core system (Level 1 Processor Prototype - L1PP), which runs as a standalone application.

The three main goals of this interface are to setup the configuration of the L1PP, to manage the L1PP execution and to analyse the execution status.

The menus displayed on the main frame allow the user to open/save the configuration files, change the configurations, start and stop the prototype, view the log file and the help contexts. In this section all screenshots are provided as examples.

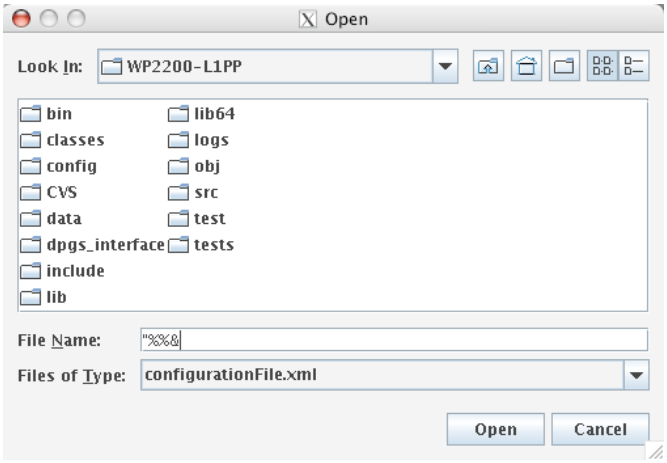
2.4.1.1. The SMOS L1 Prototype Menu

To open the prototype the user has to access the menu 'File'.



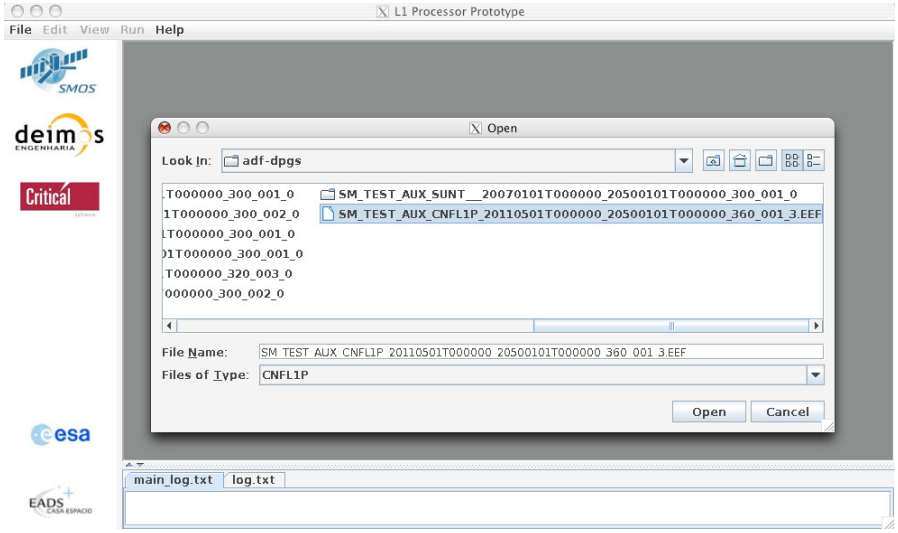
There the user has four basic options:

1. *Open L1PP Configuration File*: This allows the user to load a configuration file specific for L1PP and either edit it or use it in the L1PP execution.



Note: This step is mandatory if no configuration file was passed as argument when the application was launched.

2. *Open CNF File*: This allows the user to load a generic L1 Configuration File (AUX_CNFL1P) that contains the entire standard L1 configuration values.



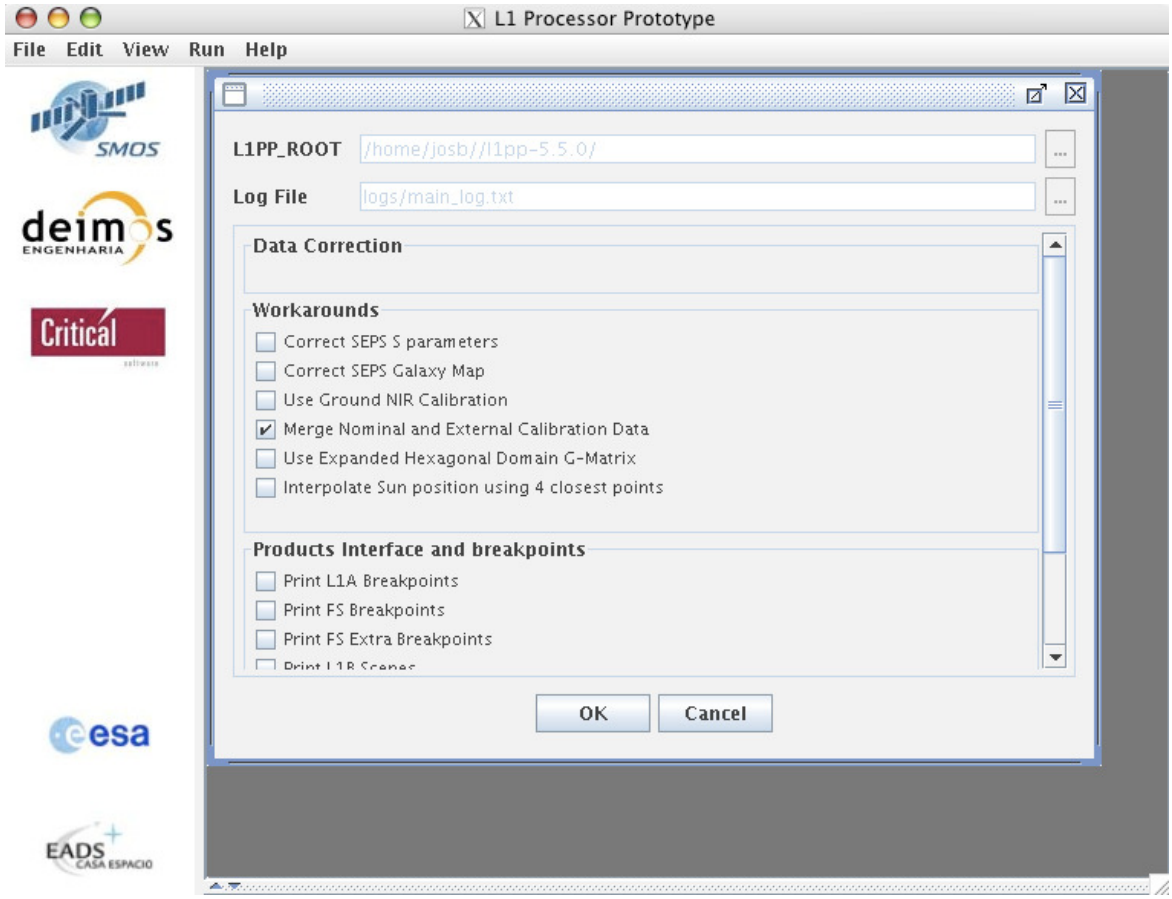
Note: This step is also mandatory if no AUX_CNFL1P file was passed as argument when the application was launched.

3. *Save/Save as*: This gives the user the possibility to save the current specific L1PP configurations to the already opened file or to a different location. The AUX_CNFL1P, however, is not editable within the L1PP application, as it is considered an external auxiliary file.
4. *Exit:* To terminate the application.

Note: The Run menu button is only enabled if the user has successfully loaded both the corresponding L1PP configuration file and the AUX_CNFL1P file. Otherwise it will remain greyed out to indicate that the L1PP still does not have enough input information to run.

2.4.1.2. Preferences Window

After loading the L1PP configuration file, there are a couple of flags that can be toggled to activate/deactivate some functionality on the prototype. This can be done in the *Preferences* panel (found under the *Edit* menu):



Note: This window can only be opened after a configurationFile.xml was loaded.

The preferences on the previous window shall be modified carefully. For instance the "Data Correction" items were added in order to correct L0 Data incorrectly generated. The user shall only modify these tags if new L0 Data is available, with these corrections already performed. The workarounds were added to cope with some missing functionality (case of the "Use NIR Ground Calibration" item~~) or to help on the validation against the SMOS End to End Performance Simulator (case of the "Correct SEPS" items)~~.

The Breakpoints items allow the user to decide whether the breakpoint ASCII Files shall be produced or not. It is important to highlight that for Data Scenarios with a considerable number of Scenes (for instance half-orbit scenarios), a large amount of breakpoint data is generated. Section 4 - Annex: Breakpoints Format describes in detail the breakpoints formats.

The complete set of flags defined in the L1PP specific configuration file are described in Section 2.5. For details on the format and contents of the AUX_CNFL1P, please refer to [AD.12]

2.4.1.3. Configure Window

This Configuration Window can be opened using the menu options "Run -> Configure" and allows the user to configure several parameters, namely:

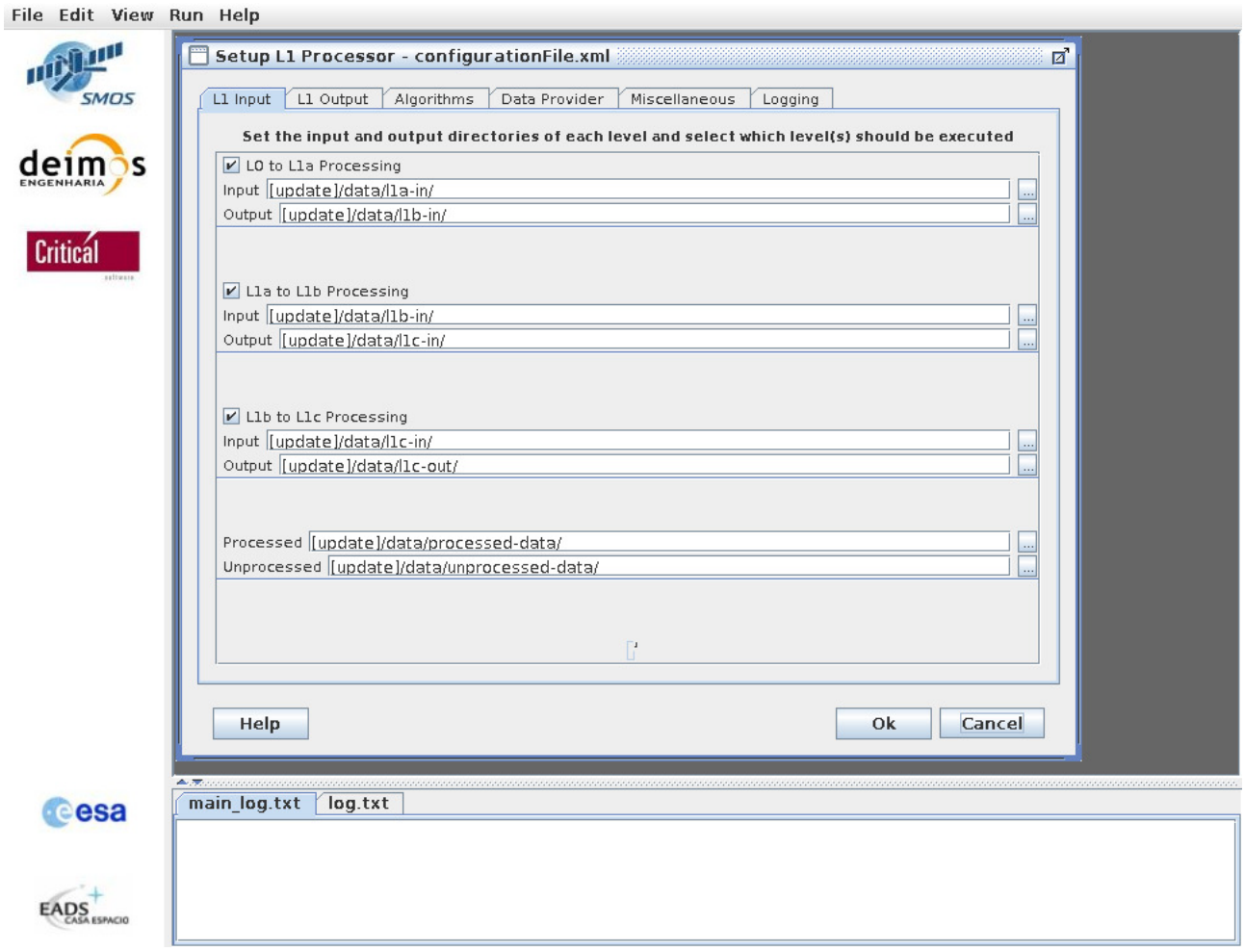
- Set the path names of the directories where data is read and written into;
- Toggle on/off specific L1PP algorithms to be applied in the processing;
- Toggle on/off the products to be generated;
- Configure Data Provider processing strategy;
- Configure Miscellaneous parameters, such as number of CPUs to be used, processing mode (test/operational),etc;
- Select the logging level.

Each of the enumerated items is configured in a different tab. For each tab there is a context help button which provides further details. For the full list of configuration items and possible values, please refer to Section 2.5).

Several configuration parameters available in the configuration file are not editable from this panel.

The configuration window is composed by 6 tabs. Each of the tabs is described hereafter:

2.4.1.3.1. L1 input



This tab allows the user to:

1. Toggle on/off the levels to be processed.

Three checkboxes are presented, each one is associated to a level of processing: L0-L1a, L1a-L1b, L1b-L1c. If the user wants to disable a certain level of processing for instance L0-L1a then the corresponding checkbox should be unchecked. In terms of processing, this means that the input data will be fetched passed to the corresponding processing units and the output products will be generated for that specific level.

2. Setup the input and output directories of each level of processing.

For each level the user either writes the pathname directly in the checkbox or presses the button on the right hand side of the *text field* and uses the file browser window to select the pathname

3. Select the *processed data* directory.

The user specifies the path to where the products (e.g L0 Ancillary or L1a HKTM) are moved after being successfully processed.

4. Select the *unprocessed data* directory.

Used to specify the path to where the products (e.g L0 Correlated or L1B Science) are moved in the case some error occurs and the file can not be processed.

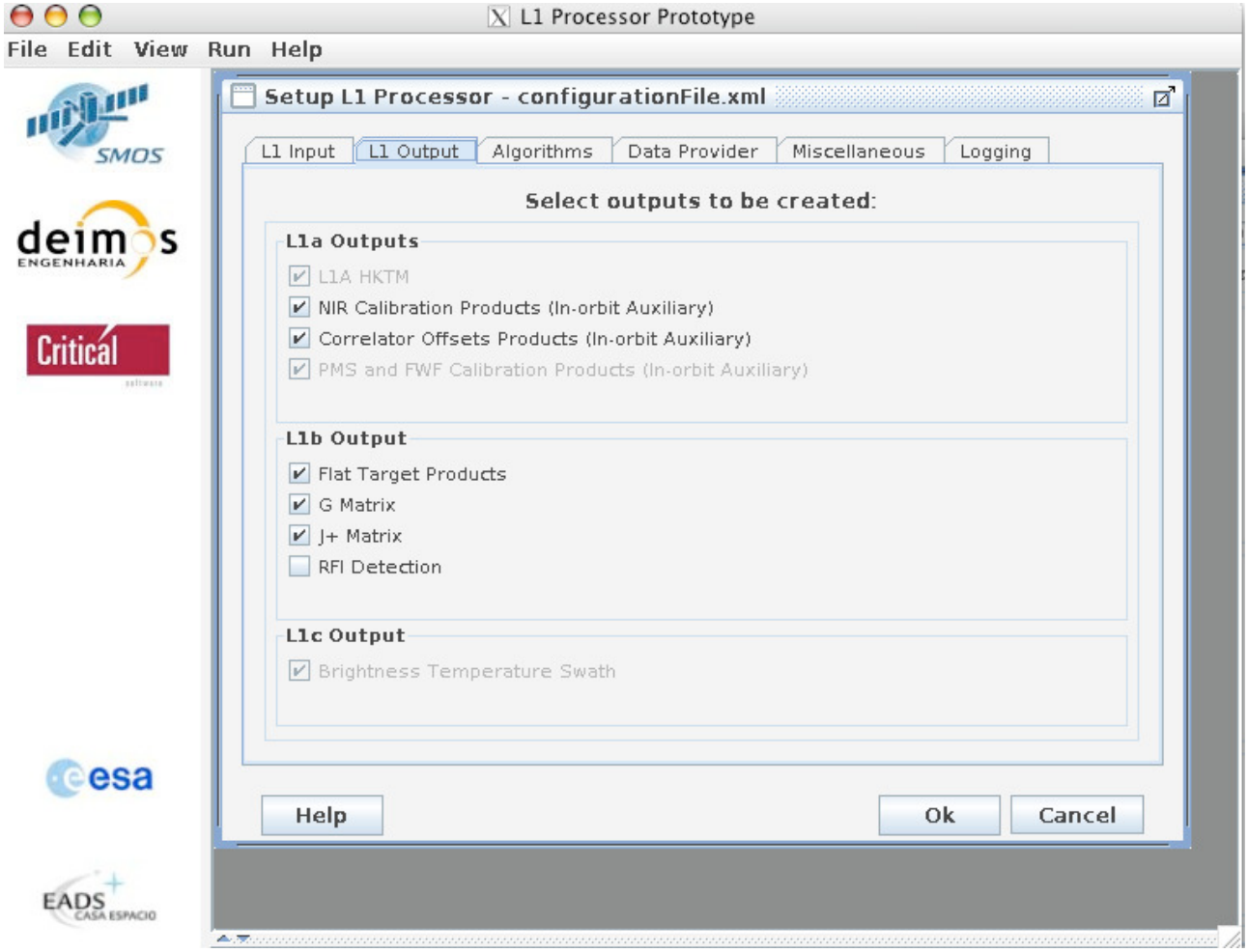
5. Select the *auxiliary data* locations.

Used to specify the path to where the auxiliary data files are located. Since the user might want to use different baselines for processing the data, the configuration file supports an independent directory for each ADF[4].

NOTE: Without having filled the *text fields* correctly the L1PP will not be able to process any data.

[4] ~~Some of the ADFs supported by the configurationFile for backward compatibility (e.g. AUX_PATT99, FLATT, JMAT, GMAT, VTEC, APOD) are not needed for nominal processing. The absence of these files will not stop the processing or have any impact on the quality of the data produced.~~

2.4.1.3.2. L1 Output



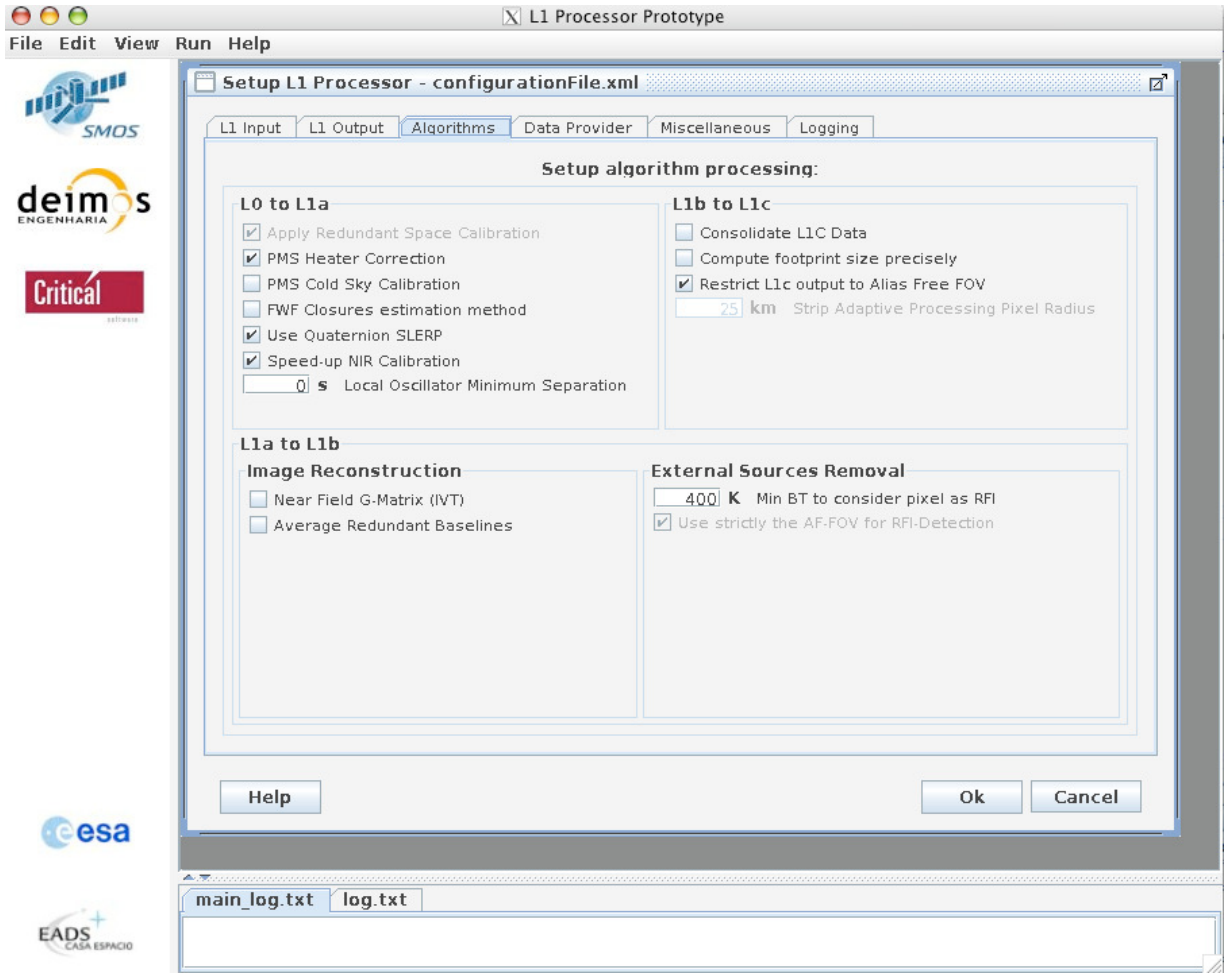
This tab allows the user to select the products to be generated.

Several checkboxes are presented, each one is associated to a specific product, for example NIR Calibration Products. If the user does not want the system to generate a specific product then he should uncheck the associated checkbox.

Note: In-Orbit auxiliary files are calibration products produced in orbit.

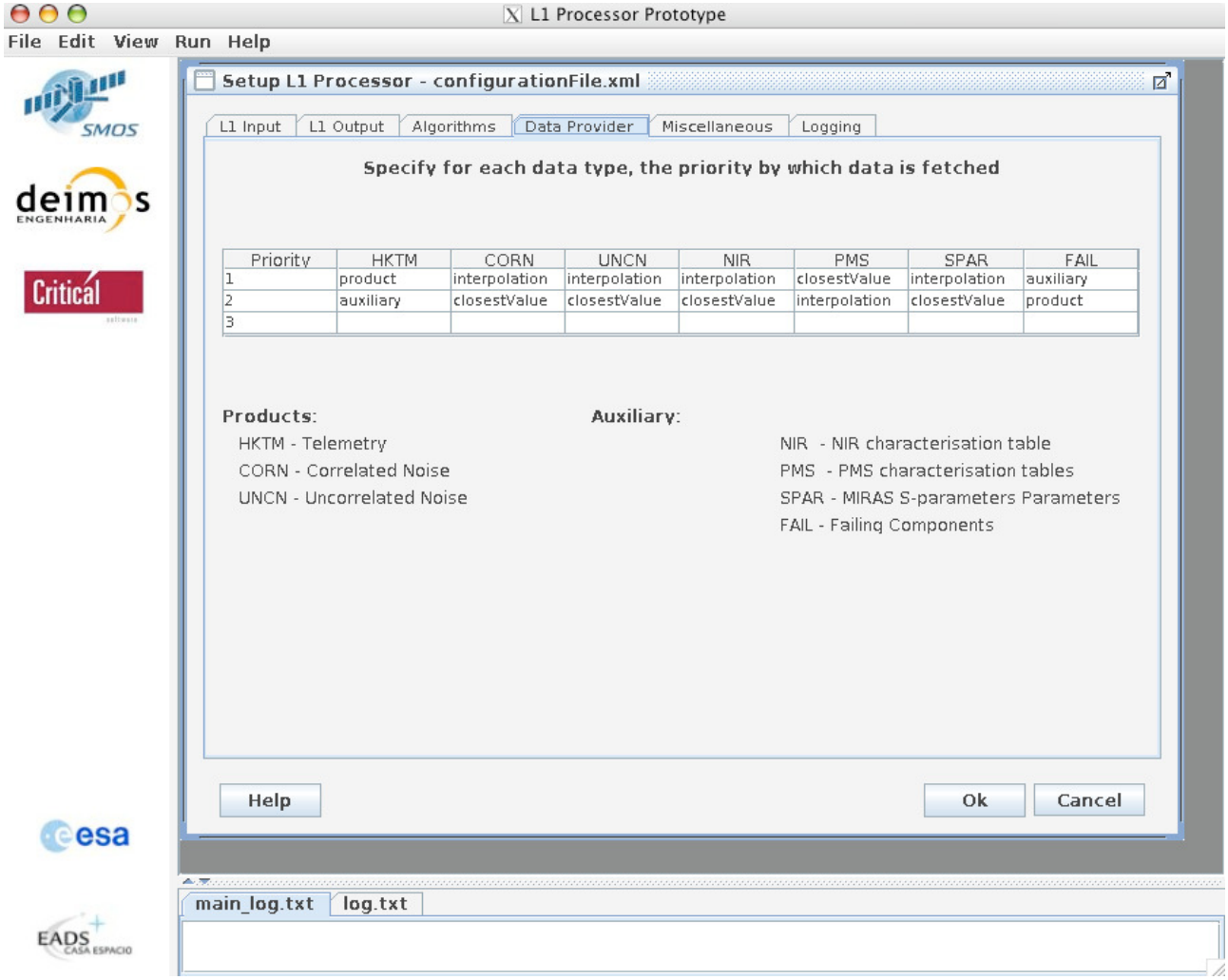
2.4.1.3.3. Algorithms

This tab allows the user to select the L1 prototype algorithms to be applied in the processing. The main L1 algorithm configuration is driven by the AUX CNFL1P, and is not editable from this tab. Only L1PP specific fields can be edited from here:



Several checkbox are presented, each one is associated to a specific algorithm, e.g. Near Field G Matrix (used during Image Validation Test campaign at ESTEC). The user may enable/disable the processing of certain L1PP specific tasks, using a specific algorithm.

2.4.1.3.4. Data Provider Strategy



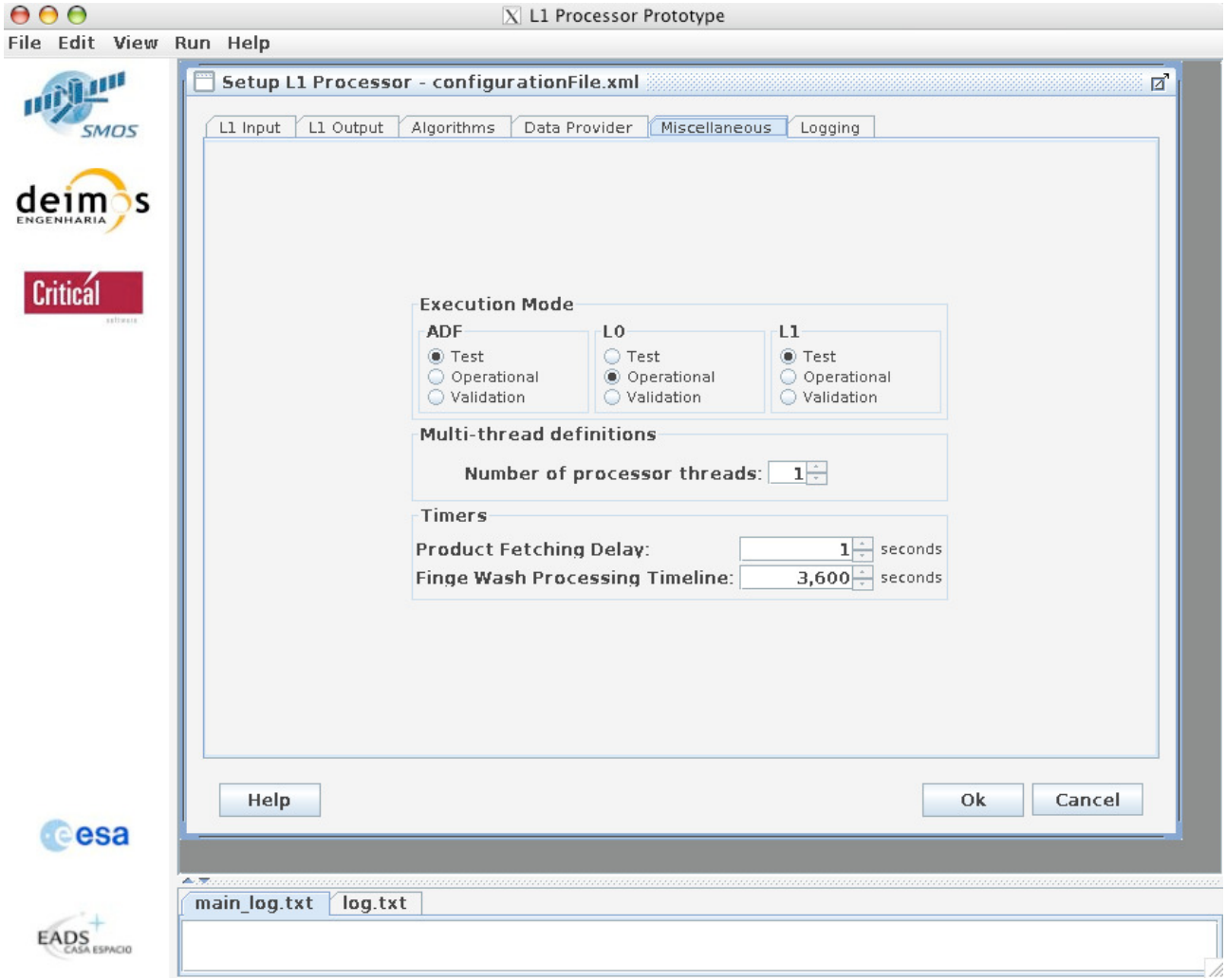
This tab allows the user to configure the processing strategy for each data quantum fetched by Data Provider (Cache Management module included in the *Core Components* library). For instance in case of HKTM, if the user wants the system to fetch in first place HKTM data from the product file and only if the product file is not available fetch data from the auxiliary file, the user should set the HKTM column with:

a. priority 1 dropdown box to *product.*

b. priority 2 dropdown box to *auxiliary*.

Each type of product has two or three priority levels and for each product there is a type of action, which can be performed.

2.4.1.3.5. Miscellaneous



This tab allows the user to:

1. Select tThe mode of processing (test or operational) for ADF files, input L0 products and output L1 products.

   In case the user specifies 'Test' for ADF or L0 products, only files with prefix SM_TEST will be accepted as input by the L1PP. In case it is specified for L1 products, L1PP will generate output products with the SM_TEST prefix

   In case the user specifies 'Operational' for ADF or L0 products, only files with prefix SM_OPER will be accepted as input by the L1PP. In case it is specified for L1 products, L1PP will generate output products with the SM_OPER prefix.

2. Setup the number of threads (any number) to be used by the L1PP (this takes advantage of any multi-CPU platform in which L1PP is run);

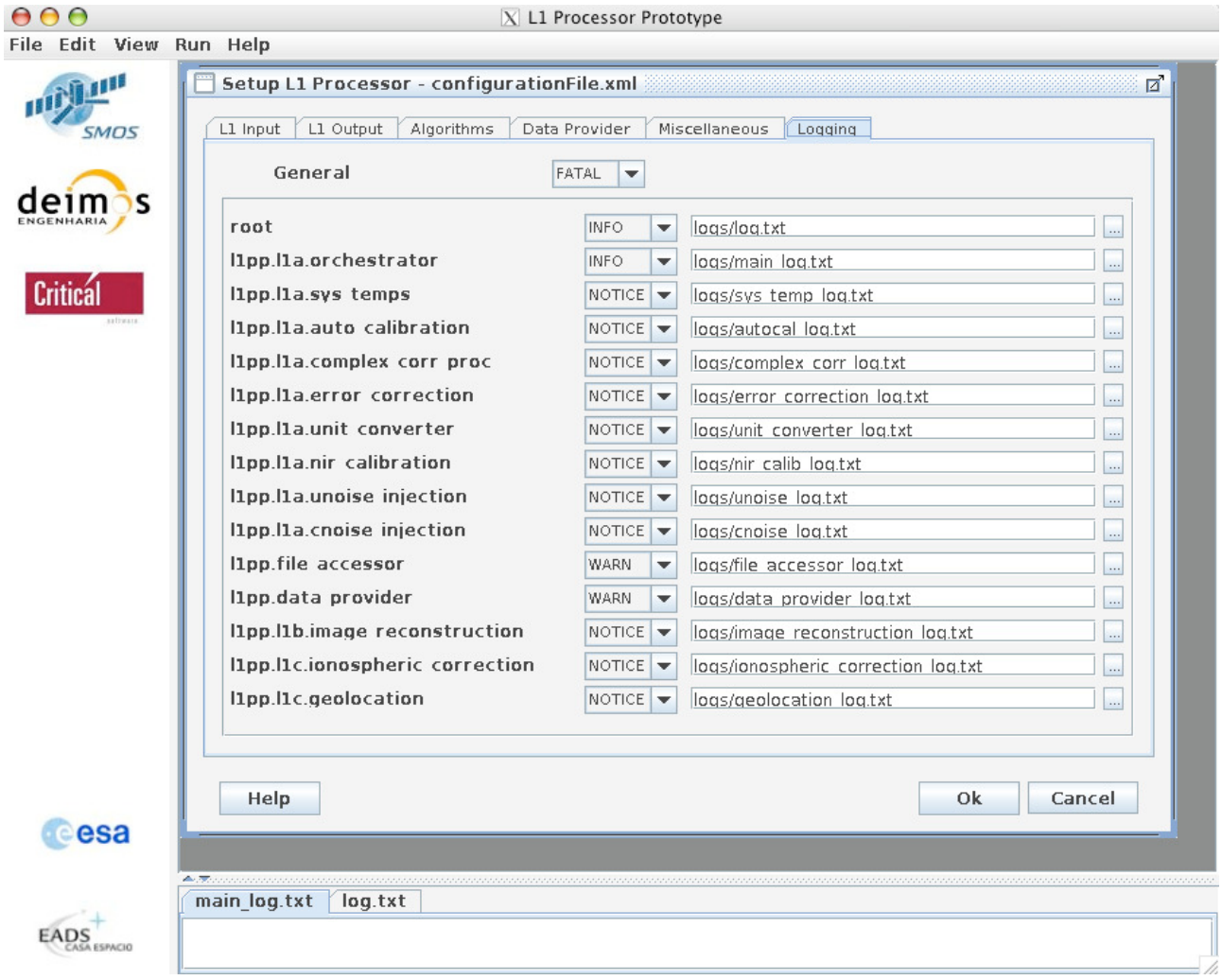
3. The product fetching delay;

   This configuration item is used to specify the delay existing between the arrival of the first product and the beginning of the processing. The user has to press the up and down buttons to specify the delay number of seconds.

4. FWF timeline.

   This configuration item specifies the timeline to be considered when fetching L1a Fringe Wash products. This means that depending on the timeline configuration, the number of products to be processed by the G Matrix Generator will vary. If the timeline is quite long then more FWF products will be processed otherwise less products will be processed by the G Matrix generator. Note that this field is obsolete from v 1.3.0, as the FWF consolidation process takes care of grouping the available files into a unique one.

2.4.1.3.6. Logging



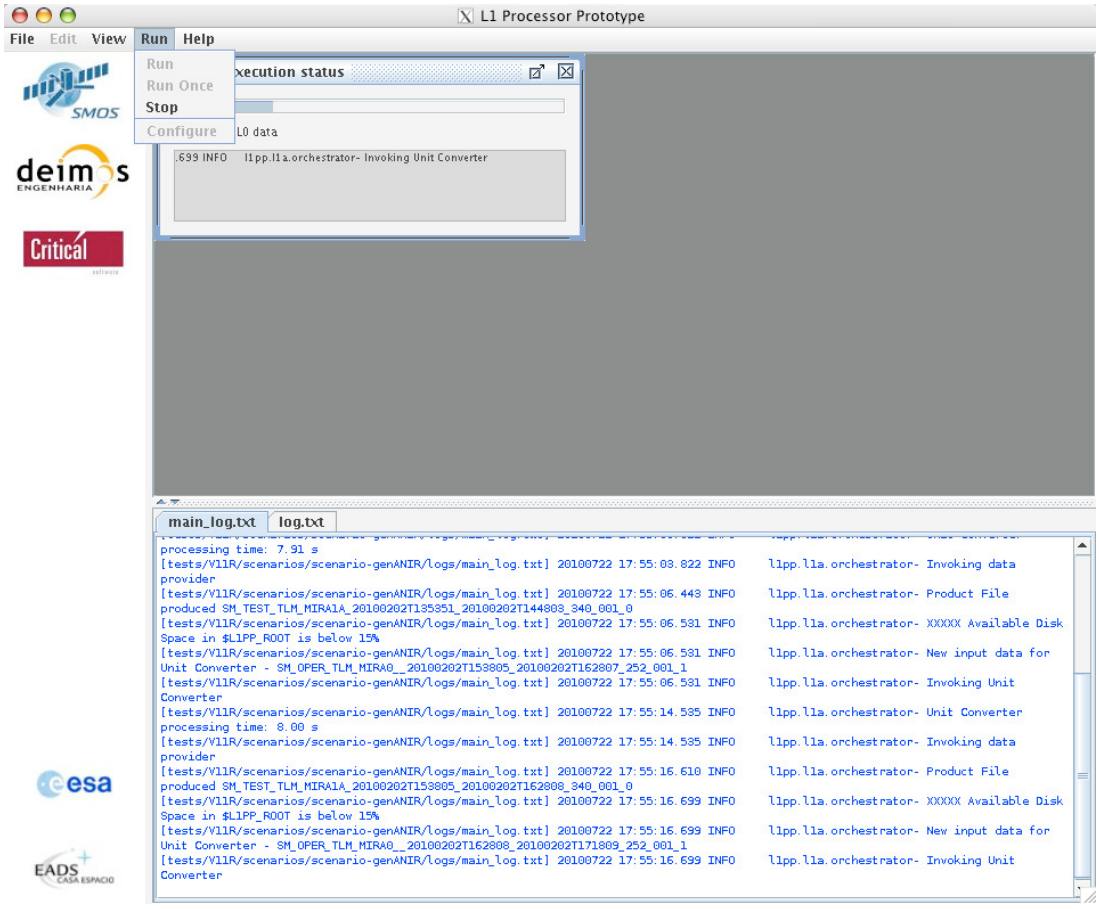
This tab allows the user to select the logging level for each module. The available modes are:

a. Debug;
b. Info;
c. Notice;
d. Error;
e. Fatal.

The top level outputs debug messages in the log file plus any of the messages of the lower levels of log (Info, Notice, Error, Fatal). The second one (Info) outputs in the log file info messages plus any of the message of the other lower levels (Notice, Error, Fatal). For the other levels the same reasoning applies.

2.4.1.4. System execution

After loading and tuning the configuration files, the prototype can be run using the "Run" Menu:



Start processing

To start the L1PP execution the user should select the “Run” option in the “Run” menu. As a result of this action a separate process is launched, which is dedicated to the L1PP execution.

The user can monitor the status of the process with an external application such as TOP. The user should search in process list for name 'l1po'[5].

Note: The user must save the configurations, if any change was performed, before start the processing. The configuration preferences are not save automatically. For saving the modifications, the user must explicitly choose the “File/Save” or “File/Saves” options in the main window.

Monitoring processing status

The user has to ways of monitoring the processing status:

1. Check the progress bar:

After the user selects “Run” a progress bar is launched showing the percentage of work done and which step of the processing is being executed (e.g. error correction, image reconstruction).

2. View the log file (refer to section Logging monitoring).

Stop processing

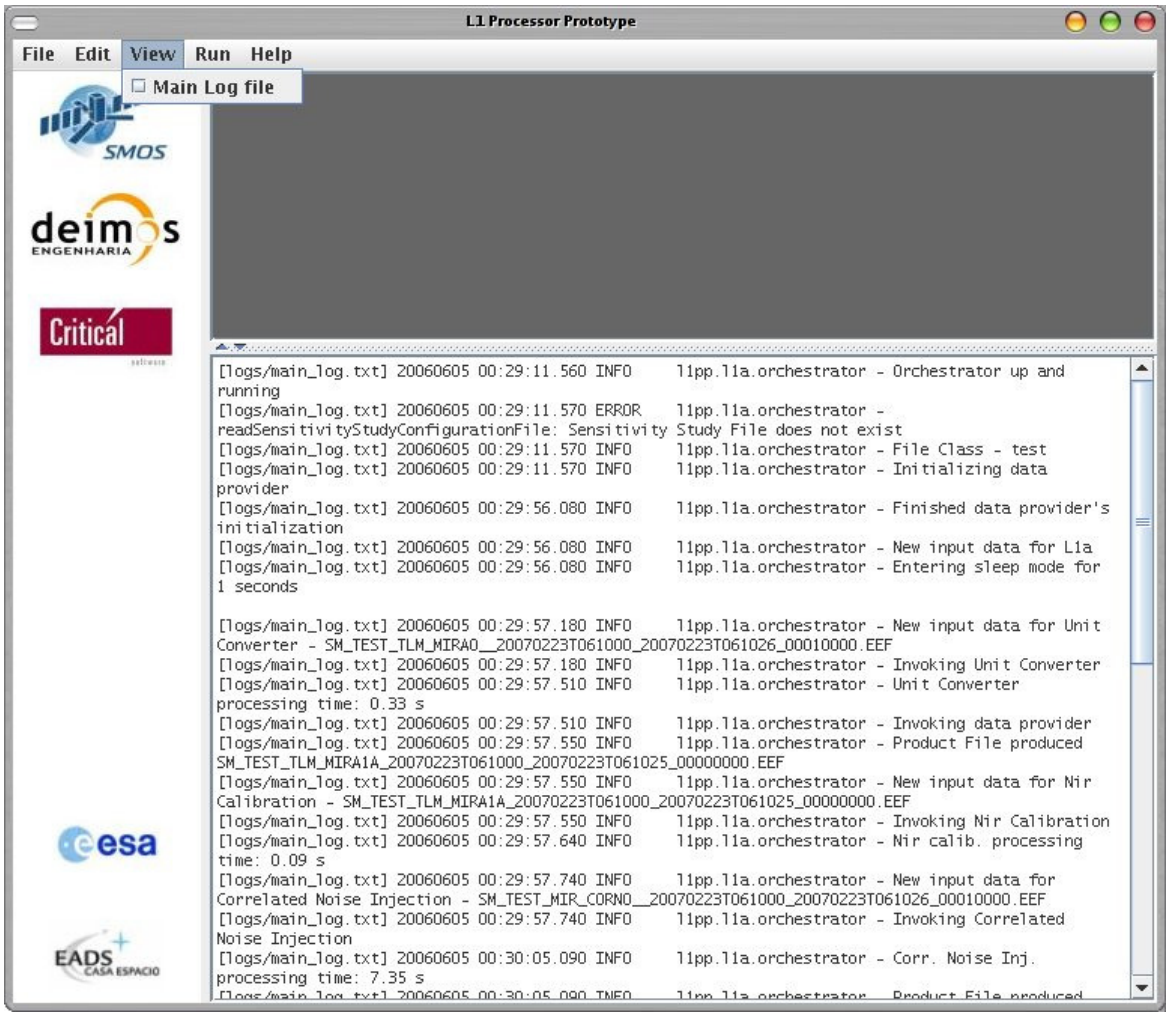
At any time the user may stop the process by selecting 'Stop' in the “Run” menu. As a result of this the process where the L1PP is being executed will be signalled and its execution will end.

NOTE: some temporary files might be left on the temporary directory ($L1PP_ROOT/data) if some processing was broken using this menu option.

2.4.1.5. Logging monitoring

The user can monitor the log file produced by the prototype by selecting the menu *View->Main Log File*. The window displayed is simply a dump of the file *main_log.txt* located in the log directory and defined in the log configuration file *log4crc* (described in Section 2.5.2).

[5] “L1 ProOcessor”



In the window displayed the user can scroll up and down and view the steps that were executed by the prototype. The User must enable the log viewing in the "View" menu, during execution.

2.4.2. Running the prototype in text mode

Alternatively the user may start the prototype from the command line, in text mode, as follows:

$ sh run-l1pp64.sh $AUX_CNFL1P_FILE_NAME –text

Text mode will not invoke the GUI and will process the data with the configuration file available in the *config* directory.

The run-l1pp64.sh s script will update the LD_LIBRARY_PATH environment variable and will run the prototype using default configuration files: *config/configurationFile.xml* and *config/log4crc*.

If the user wants to run the prototype with different configuration files, the following command should be used:

$ bin/l1po_64bits $CONFIG_FILES_PATH $CONFIG_FILE_NAME $AUX_CNFL1P_FILE_NAME

For instance:

$ bin/l1po_64bits $L1PP_ROOT/data/scenarios/scenario-01/ $L1PP_ROOT/data/scenarios/scenario-01/configurationFile.xml adf-dpgs/SM_TEST_AUX_CNFL1P_20110501T000000_20500101T000000_360_001_3 .EEF,

where the directory *$L1PP_ROOT/data/scenarios/scenario-01/* contains both the *configurationFile.xml* and the *log4crc* configuration files described in Section 2.5.

Note: If the user starts the prototype from the command line, the variables L1PP_ROOT and LD_LIBRARY_PATH must be set according to the instructions provided above (Section 2.3.5).

Additionally, it is also possible to enable the execution of only one L1PP instance, instead of running it in an endless loop. This mode is activated by passing the option "-once" to the L1PP executable command, for example:

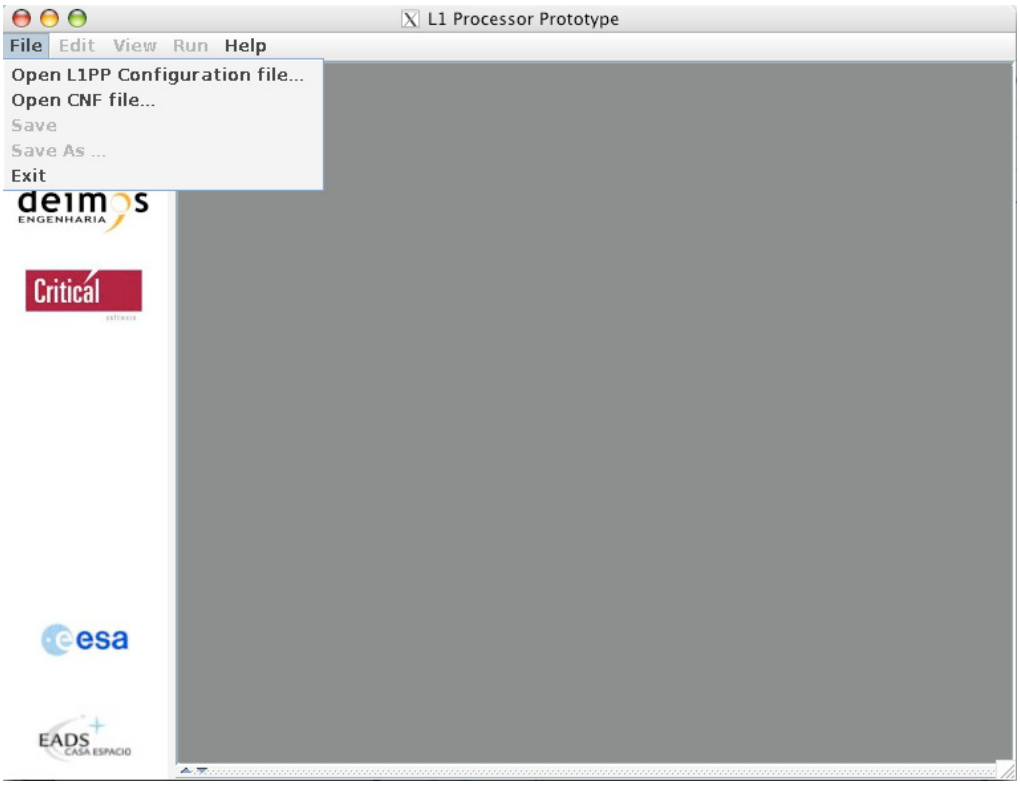
$ bin/l1po_64bits $CONFIG_FILES_PATH $CONFIG_FILE_NAME $AUX_CNFL1P_FILE_NAME -once

2.4.3. Loading Different Scenarios

The Installation Kit includes different test scenarios that can be loaded and run in the prototype. The scenarios are provided in the directory *$L1PP_ROOT/~~data~~test/scenarios/scenario-XX/* , where XX represents the description of the scenario. The scenarios are described in detail in the Test Data Description document [AD.11]. The paths defined in the configurationFiles inside the scenarios must be updated to the user environment by executing:

$ scripts/update-config-path.sh $L1PP_ROOT/~~data~~test/scenarios/scenario-XX/ configurationFile.xml

To load a scenario, the user shall use the "File" Menu in the main window and choose the corresponding *configurationFile.xml* file:



Then, for each scenario, a configuration file is provided in *$L1PP_ROOT/~~data~~tests/scenarios/scenario-XX/configurationFile.xml*, while the input data is placed, or instance in *$L1PP_ROOT/~~data~~tests/scenarios/scenario-XX/l1a-in/ directory*.

After running these scenarios, the outputs of the execution will be places in *$L1PP_ROOT/~~data~~tests/scenarios/scenario-XX/processed-data/* or *$L1PP_ROOT/~~data~~tests/scenarios/scenario-XX/l1c-out/*. Some of the scenarios were previously executed, and the results are already provided in *$L1PP_ROOT/~~data~~tests/scenarios/scenario-XX/processed-data/results/*.

2.4.4. Generation of G and J+ Matrices ADFs

As mentioned previously, due to the size of the file, the G-Matrix (GMATD) ADF is not directly provided with the prototype. However, the Test Data Package included in the release of the prototype contains a scenario with the Fringe Wash Calibration L1a products used as inputs for the G and J+ Matrices (scenario-genMatr).

Due to problems with the handling of G and J+ Matrices, it is recommended that users installing L1PP in a big endian machine (e.g. PowerPC) should generate the G and J+ Matrices locally. This will avoid problems, as the J+ matrix provided in the Test Data webpage is generated on a little endian environment.

Therefore, in order to generate the matrices, the user only needs to only load the scenario-genMatr and run the prototype. Please refer to the previous section for general instructions on how to load a test scenario.

Since the JMATD ADF is already provided in the SMOS L1PP webpage, and the GMATD can be generated in less than 20 minutes (while the JMAT may take up to 40h), the user may choose to generate only the G-Matrix, by switching off the flag "J+ Matrix" in the outputs tab of the L1 output panel (shown in section 2.4.1.3.2). In addition it is also possible to disable the generation of the G-Matrix by switching off the flag "G Matrix" in the same tab.

The nominal processing will be to generate both matrices in a single execution.

Note: Several aspects shall be taken into account when generating the matrices:

1. For generating the G-Matrix at least 16 Gigabytes of space shall be available on disk;

2. The generation of the J+ Matrix shall be executed only in 64-bit mode.

3. After the generation of the matrices, the files shall be placed in the ADFs directory (typically *$L1PP_ROOT/data/adf-dpgs*) in order to be used by the prototype. The user shall check if an ADF with the same name already exists, before moving the files to the ADFs directory. If the user wants to keep several versions of the ADFs on the same directory, the counter filed on the file name shall be used:

SM_TEST_MIR_GMATD__20100202T144523_20500101T000000_350_001_0
SM_TEST_MIR_GMATD__20100202T144523_20500101T000000_350_002_0

If these two files with the same name are placed in the same directory, Data Provider will always fetch the one with the higher counter [6], i.e.,

SM_TEST_MIR_GMATD__20100202T144523_20500101T000000_350_002_0

To generate the expanded G-matrix it is necessary to have at least 41 GiB of disk space available.

2.5. Configuration Files

The prototype requires three configuration files, contained typically in the directory *$L1PP_ROOT/config*[7]. One is used to configure the prototype (*configurationFile.xml*), the second one is used to configure the logging (*log4crc*) and the third one (*binxml-fh.conf.xml*) is used to configure the BinXML library. The prototype and the logging configuration files are described in the following subsections (2.5.1 and 2.5.2), while the binxml configuration file is out of the scope of the current

[6] The counter field is the last 3 digit number in the filename (i.e. 002).

[7] While the configurationFile.xml and log4crc may be placed in a directory other than *$L1PP_ROOT/config*, the *binxml-fh.conf.xml* shall always be placed in this directory.

document not being therefore described here [8]. Furthermore, two other optional configuration files, the *configurationFileSensitivyStudy.xml* and *ivtConfiguration.xml* files, must be present in the *$L1PP_ROOT/config/* directory whenever Sensitivity Studies are performed or the Near Field Algorithm for G-matrix generation is used, respectively. These are described in subsections (2.5.3 and 2.5.4).

Note: The configuration files from release 5.5.0 are no longer compatible with previous ones due to the introduction of the Hexagonally Expanded G-Matrix and other flags.

2.5.1. L1PP configuration file

In general terms, the configuration file (*configurationFile.xml*) contains a set of tag elements. The general tag contains configurations that are applicable to all the modules of the prototype. The moduleX tag contains configurations specific of module X.

```
<!-- This is the configuration file for the SMOS L1PP -->
<smos-l1pp>
  <general>
    general configuration valid for the whole L1PP
    comment each variable stating its purpose, possible values, usage
  </general>
  <moduleX>
    specific configuration for a given module
  </moduleX>
</smos-l1pp>
```

A detailed description of each tag is provided hereafter.

General Configuration Elements

[8] For further information about the binxml configuration file, please refer to the EE XML/Binary CFI User Manual [RD3].

Configuration item	Description	Key	Possible Values
general/root_path	Path (absolute) of the L1PP (currently not used by the prototype)	N/A	any
general/miscellaneous/system	4 letter key to be used in DPGS headers “System” field	N/A	L1PP
general/miscellaneous/processing_centre	4 letter key to be used in DPGS headers “Processing_Centre” field	N/A	DMEP
general/miscellaneous/logical_processing_centre	3 letter key to be used in DPGS headers “Logical_Processing_Centre” field	N/A	DME
general/miscellaneous/num_threads	Number of threads to be used while processing (can be set to any number and will be an input to the OpenMP functions)	N/A	Numeric value (default 1)
general/miscellaneous/adf-file-class	File class of the auxiliary data files to be used as input	N/A	“test”, “operational”, “validation” or reprocessing
general/miscellaneous/l0-file-class	File class of the L0 product files to be used as input	N/A	“test”, “operational”, “validation” or reprocessing
general/miscellaneous/l1-file-class	File class of the L1 product files to be generated as output	N/A	“test”, “operational”, “validation” or reprocessing
general/miscellaneous/l0_data_delay	Delay between input file detection and start of data processing in seconds	N/A	Numeric value (default 1s)
general/miscellaneous/orchestrator_time_window	Time window for the orchestrator to mimic DPGS staggered processing (e.g. by orbit passes)	N/A	Numeric value (default 0s)
~~general/miscellaneous/fwf_timeline~~	~~Time interval in seconds for gathering FWF products for consolidation. (This variable will be deprecated when the new calibration consolidation is implemented)~~	~~N/A~~	~~Numeric value (default 3600s)~~

Table 4: General Configuration Elements

Data Provider Configuration Elements

Configuration item	Description	Possible Values
dataType	Designation of the data type to be loaded into cache	HKTM, CORN, UNCN, CORU, UNCU, ANIR, FWAS, FWAU, NIR, PLM, PMS, LCF, SPAR, FAIL, BWGHT, PATT, PATT99, DGG, RFI, MASK, LSMASK, ORBSCT, FLATT, GLXY, JMAT, GMAT, VTEC, SUNT, MOONT, BSCAT, BFP, MISP, APOD, BULLB, CNFL1P
fileNamePattern	The pattern of the filename that contains the auxiliary data (# stands for the numbering of the file, in case there are multiple versions, e.g., there 72 AUX_PATT files)	TLM_MIRA1A, MIR_ACNN1A, MIR_AUNN1A, MIR_ACNU1A, MIR_AUNU1A, MIR_ANIR1A, MIR_AFWS1A, MIR_AFWU1A, AUX_NIR___, AUX_PLM___, AUX_PMS___, AUX_LCF___, AUX_SPAR__, AUX_FAIL__, AUX_BWGHT__, AUX_PATT##, AUX_PATT99, AUX_DGG___, AUX_RFI___, AUX_RFILST_, AUX_MASK__, AUX_LSMASK_, MPL_ORBSCT, AUX_FLATT_, AUX_GLXY__, AUX_JMAT__, AUX_GMAT__, AUX_VTEC__, AUX_SUNT__, AUX_MOONT_, AUX_BSCAT_, AUX_BFP___, AUX_MISP__, AUX_APOD##, AUX_BULL_B, AUX_CNFL1P [9,10]
strategy	Processing strategy to handle the data	“product”, “auxiliary”, “interpolation”, “closestValue”, “noPriority”
directory	The absolute path to the directory where the auxiliary file is located	
Number_of_Products	The number of samples to be kept in cache Note: Currently this value is only being used for HKTM and L1a calibration data	

Table 5: Data Provider Configuration Elements

[9] The “fileNamePattern” field possible values (TLM_MIRA1A, AUX_NIR___, AUX_PLM___, AUX_PMS___, AUX_LCF___, AUX_SPAR__, AUX_FAIL__ …) correspond respectively to the same sequence of “dataType” possible values: HKTM, NIR, PLM, PMS, LCF, SPAR, FAIL…

[10] The file patterns listed here are for standard L1PP ADFs. The patterns for the DPGS V3 ADFs are slightly different but maintain the same approximate names

Orchestrator Configuration Elements

In the following table, several configuration items contain a "mandatory" attribute. When the "mandatory" attribute is set it will not be possible to modify the "value" attribute in the GUI, i.e., the only way to modify the "value" attribute will be manually, by editing the configuration file.

Mandatory attribute is false when the field is editable from the GUI, and it is true when it cannot be edited from the GUI (and it is not recommended to be altered).

Configuration item	Description	Key	Possible Values
processed-data	The absolute path to the directory where data will be moved after being processed	N/A	A string containing the path
unprocessed-data	The absolute path to the directory where data will be moved if it cannot be processed for any reason	N/A	A string containing the path
breakpoint-dir	The absolute path to the directory where breakpoints will be stored	N/A	A string containing the path
l1a-dir/input-dir	The absolute path to the directory where l1a input data will be read from	N/A	A string containing the path
l1a-dir/output-dir	The absolute path to the directory where l1a output will be set	N/A	A string containing the path
l1b-dir/input-dir	The absolute path to the directory where l1a input data will be read from	N/A	A string containing the path
l1b-dir/output-dir	The absolute path to the directory where l1b output will be set	N/A	A string containing the path
l1c-dir/input-dir	The absolute path to the directory where l1a input data will be read from	N/A	A string containing the path
l1c-dir/output-dir	The absolute path to the directory where l1c output will be set	N/A	A string containing the path
l1_output/l1a	Specification of the data to be produced by the prototype on level 1a. Setting the "value" attribute to true means that it will create an output file containing this product. Setting the same attribute to false means that it will not be created.	"L1A HKTM"	Value attribute can be true or false
		"NIR Calibration Products (In-orbit Auxiliary)"	Value attribute can be true or false

Configuration item	Description	Key	Possible Values
		"Correlator Offsets Products (In-orbit Auxiliary)"	Value attribute can be true or false
		"PMS and FWF Calibration Products (In-orbit Auxiliary)"	Value attribute can be true or false
l1_output /l1b	Specification of the data to be produced by the prototype on level 1b. Setting the "value" attribute to true means that it will create an output file containing this product. Setting the same attribute to false means that it will not be created. There is also the possibility of creating an RFI Histogram Map by using multiple L1A input products. In case this option is selected, L1PP will not generate any L1B output products, and its output will be restricted to this RFI Histogram Map.	"Flat Target Products"	Value attribute can be true or false
		" G Matrix"	Value attribute can be true or false
		"J+ Matrix"	Value attribute can be true or false
		RFI Detection	Value attribute can be true or false
l1_output /l1c	Specification of the data to be produced by the prototype on level 1c. Setting the Value attribute to true means that it will create an output file containing this product. Setting the same attribute to false means that it will not be created.	"Brightness Temperature Swath"	Value attribute can be true or false
algorithm/l1a	Whether to apply Redundant Space Calibration (not yet implemented). This field is not editable.	"Apply Redundant Space Calibration"	Value attribute can be true or false

Configuration item	Description	Key	Possible Values
	Whether to apply a PMS Voltage correction based on the Heater Status of each LICEF. This correction is applied at the level of L0 to L1a HKTM processing.	"PMS Heater Correction"	Value attribute can be true or false
	if TRUE, uses the calibration baseline tested during commissioning, in which the PMS External data was consolidated. If FALSE, uses the existing baseline where only internal PMS calibration is consolidated. The setting of this of this flag to TRUE implies that L1PP will use a CRSx1A schema that is not yet distributed officially within the DPGS.	"PMS Cold Sky Calibration"	Value attribute can be true or false
	Whether to apply FWF Closures method, as opposed to the amplitude and phase system of equations. The closures method is not the processing baseline. However, its use is supported and L1PP accuracy will not be adversely affected. It is recommended to use it only for scientific validation studies	"FWF Closures estimation method"	Value attribute can be true or false
	Use Quaternion Spherical Interpolation instead of linear propagation with angular velocity when computing AOCS data in HKTM	"Use Quaternion SLERP"	Value attribute can be true or false
	Use specific tailored NIR calibration configuration to speed up the detection process	"Speed-up NIR Calibration"	Value attribute can be true or false
	Value to be used during ~~commissioning~~ tests to decimate the LO calibration sequences and simulate a longer inter-calibration period.	"Local Oscillator Minimum Separation"	Value in seconds

Configuration item	Description	Key	Possible Values
algorithm/l1b	Generate IVT G/J+ matrices using Near Field algorithms (needs ivtConfiguration.xml file). This should only be used to generate/use G/J+ for IVT data processing. This flag is also used whenever processing IVT data, in order to correct several approximations needed: • NIR antenna temperature during IVT NIR calibration is the room temperature instead of the sky • IVT J+ matrix needs an SVD cutoff higher than the nominal J+ matrix L1b breakpoints in IVT are generated with Blackman apodisation and no scaling factor (sqrt(3)d^2/2)	"Near Field G-Matrix (IVT)"	Value attribute can be true or false
	Algorithm to average J matrix redundant baseline rows before computing its pseudo-inverse	"Average Redundant Baselines"	Value attribute can be true or false
	RFI Brightness Temperature threshold to consider a point as contaminated by RFI	"Min BT to consider pixel as RFI"	Value in Kelvin
	Flag to be used during RFI detection mode to restrict the search for RFI sources to the alias-free FOV instead of using the complete unit circle FOV	"Use strictly the AF-FOV for RFI-Detection"	Value attribute can be true or false
algorithm/l1c	Specification of steps of the algorithm to be performed on level 1c: Consolidate L1c data so that pixels in the first scenes with few measurements will not be reported in L1c product	“Consolidate L1c data”	Value attribute can be true or false
	Compute pixel measurement footprint with higher degree of accuracy but high computation time as well, or use the default Blackman 3dB contour for computing the pixel projection.	"Compute footprint size precisely”	Value attribute can be true or false
	Reduce number of pixels in L1c products to those contained inside the Alias Free FOV (smaller products and faster L1c processing)	"Restrict L1c output to Alias Free FOV"	Value attribute can be true or false
	Apply Strip Adaptive processing using a configurable pixel circular radius	"Strip Adaptive Processing Pixel Radius"	Value in km of the desired circular pixels

Configuration item	Description	Key	Possible Values
prototype/workarounds	Correct S-parameters usage from SEPS-GS data. SEPS-GS data does not simulate the whole path from LICEF to Noise Source, whereas real data must use the complete path. This flag must be active to process SEPS-GS data and deactivated to process real instrument data. The flag corrects the following SEPS limitations: • No electronic path simulated from the Noise Source to the cable input • No intermediate frequency delay is simulated in the delayed correlations (FWF shape) • No complex lambda is simulated on LICEF-NIR correlations	"Correct SEPS S parameters"	Value attribute can be true or false
	Use Galaxy map off centre to correct for wrong usage in SEPS v4	"Correct SEPS Galaxy Map"	Value attribute can be true or false
	Use default values for the NIR calibration parameters measured at an antenna temperature of 77.35K during TUD campaign. This value is needed to take into account that SEPS-GS NIR simulated data is performed with an antenna temperature of 77.35K instead of the Sky temperature (3.5K)	"Use Ground NIR Calibration"	Value attribute can be true or false
	Merge Nominal and External Calibration Data into a single type of nominal L1a Calibration products, irrespective of the origin of the L0 data. ~~This flag will be used during commissioning, in conjunction with the "PMS Cold Sky Calibration" flag, to test a potential new calibration baseline.~~	"Merge Nominal and External Calibration Data"	Value attribute can be true or false
	Build a hexagonally expanded G-Matrix and use it when applying Foreign Sources corrections for extended sources.	"Use Expanded Hexagonal Domain G-Matrix"	Value attribute can be true or false
	Use the four closest points wrt the Sun position and corresponding G-matrix rows to estimate the Sun BT.	"Interpolate Sun position using 4 closest points"	Value attribute can be true or false
prototype/output	Toggles the generation of breakpoints that are active: ❑ Print L1a breakpoints ❑ Print Foreign Sources breakpoints		

Configuration item	Description	Key	Possible Values
	❑ Print Foreign Sources extra breakpoints (not described below, used for internal verification only) ❑ Print L1b breakpoints (u, v domain) ❑ Print L1b breakpoints (xi, eta domain) ❑ Print L1c breakpoints (scenes) ❑ Print L1c breakpoints (swaths) Mandatory attribute can be true or false	"Print L1A Breakpoints"	Value attribute can be true or false
		"Print FS Breakpoints"	Value attribute can be true or false
		"Print FS Extra Breakpoints"	Value attribute can be true or false
		"Print L1B Scenes"	Value attribute can be true or false
		"Print L1B Scenes BT"	Value attribute can be true or false
		"Print L1C Scenes"	Value attribute can be true or false
		"Print L1C Browse"	Value attribute can be true or false

Table 6: Orchestrator Configuration Elements

Note: In the current version of the prototype one configuration file can be loaded and edited at a time.

The values presented in the configuration file can be edited and changed using the GUI, as shown in section 2.4.1.3.

2.5.2. Logging configuration file

This configuration file defines one log file per processing unit (*log4crc*).

Configuration item	Description	Possible Values
layout	Configures the layout of the information displayed (time, date, etc.)	Name attribute Type

Configuration item	Description	Possible Values
category	The category tag configures the logging of a module. It contains three attributes: name, priority and appender. The name attribute specifies the name of the category The priority specifies the level of logging. The appender specifies the relative path and the name of the file that will contain the log data	Name attribute – “root”, “l1pp.l1a.orchestrator”, ”l1pp.l1a.sys_temps”, ”l1pp.l1a.auto_calibration”, ”l1pp.l1a.complex_corr_proc”, ”l1pp.l1a.error_correction”, ”l1pp.l1a.unit_converter”, ”l1pp.l1a.unoise_injection”, ”l1pp.l1a.cnoise_injection”, ”l1pp.file_acessor”, “l1pp.data_provider”, “l1pp.l1b.image_reconstruction”, “l1pp.l1c.ionospheric_correction”, “l1pp.l1c.geolocation" Priority attribute – “debug”, “info”, “notice”, “warn”, “error” or “fatal” Appender – can assume any file name value

Table 7: Logging configuration elements

Note that the logs are “cumulative”, meaning that each message will be logged into the file defined for the corresponding processing unit, as well as into the files of any “parent” log. The “root” log will contain all messages that are logged into each individual log.

2.5.3. Sensitivity Study configuration file

The Sensitivity Study configuration file is used to set up the behaviour of the L1PP when the user wants to introduce errors in the internal processing of the data. It is currently being used to analyse the impact produced on the Radiometric Accuracy of the MIRAS instrument and it configures:

- Errors in the measurement of the on-ground data;
- Errors in HKTM processing and calibration parameters computations;
- Errors in the estimation of the necessary corrections (Foreign Sources Removal and Total Electron Content estimates);
- Errors induced by external sources.

If the Sensitivity Studies configuration file (*configurationFileSensitivyStudy.xml*) is present in the “config” directory, it will be read by the L1 Processor Prototype and the defined errors will be introduced directly in the code. In this manner, there is no need to produce new sets of ADF files or external parameter sets for each type of error the user wants to introduce. The computation errors are introduced right after the perturbed quantity is first computed. As a safeguard, the user always knows if a Sensitivity Studies Config file has been read, from the L1PP output to the terminal.

If the user wishes to run the L1PP without any error introduction, the Sensitivity Studies configuration file should be removed from the "config" directory. In the L1PP distribution an example file is provided named as "configurationFileSensitivyStudyExample.xml".

The configurable parameters are:

Table 8: Sensitivity Study configurable variables

Name	Value	Comment
ss_variable_type	S_PAR_POWER_DIVIDER_AMP	Power Dividers S-Parameters amplitude
	S_PAR_POWER_DIVIDER_PHASE	Power Dividers S-Parameters phase
	S_PAR_NOISE_SOURCE_AMP	Noise Sources S-Parameters amplitudes
	S_PAR_NOISE_SOURCE_PHASE	Noise Sources S-Parameters phase
	S_PAR_CABLES_AMP	Cables S-Parameters amplitude
	S_PAR_CABLES_PHASE	Cables S-Parameters phase
	S_PAR_SWITCHES_AMP	Switches S-Parameters amplitude
	S_PAR_SWITCHES_PHASE	Switches S-Parameters phase
	OHMIC_EFFICIENCY	Ohmic Efficiency values
	PMS_GAIN	PMS gains
	PMS_OFFSET	PMS offsets
	BEST_FIT_PLANE_YAW	BFP yaw
	BEST_FIT_PLANE_PITCH	BFP pitch
	BEST_FIT_PLANE_ROLL	BFP roll
	NIR_L1	NIR attenuator
	NIR_L2	NIR attenuator
	NIR_LNC	NIR attenuator
	NIR_LA	NIR attenuator
	NIR_LDA	NIR attenuator
	NIR_LDC	NIR attenuator
	NIR_IDSW	NIR attenuator
	CO_POLAR_AMP	Antenna Patterns Co-Polar measurement amplitude

Name	Value	Comment
	CO_POLAR_PHASE	Antenna Patterns Co-Polar measurement phase
	CROSS_POLAR_AMP	Antenna Patterns Cross-Polar measurement amplitude
	CROSS_POLAR_PHASE	Antenna Patterns Cross-Polar measurement phase
	ANTENNA_POSITIONS	Antenna Positions
	HKTM_POSITION	Instrument position
	HKTM_VELOCITY	Instrument velocity
	HKTM_ATTITUDE	Instrument attitude
	HKTM_ANGULAR_VELOCITY	Instrument angular velocity
	NIR_PULSE_LENGTH	NIR output
	SUN_MOON_BT	Tb values for Sun and Moon
	SUN_GLINT_BT	Tb values for Sun glint
	BACKLOBES_BT	Tb value for backlobe radiation
ss_error_type	RANDOM	Error will be introduced by generating a random number up to the limit configured
	OFFSET	Configured limit is introduced directly
ss_error_limit	*numerical value*	Error limit to be used
ss_error_value	FIXED	Limit is a absolute value
	RELATIVE	Limit is a percentage of the configured quantity

As an example, a configuration file to introduce errors in the Antenna Patterns, both in the co-polar and cross-polar measurements will look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<sensitivity_studies_config>
  <test_dir>/test1011/</test_dir>
  <ss_variable>
    <ss_variable_type>CO_POLAR_AMP</ss_variable_type>
        <ss_error_type>RANDOM</ss_error_type>
        <ss_error_value>RELATIVE</ss_error_value>
        <ss_error_limit>10</ss_error_limit>
  </ss_variable>
  <ss_variable>
    <ss_variable_type>CROSS_POLAR_AMP</ss_variable_type>
        <ss_error_type>RANDOM</ss_error_type>
        <ss_error_value>RELATIVE</ss_error_value>
        <ss_error_limit>10</ss_error_limit>
  </ss_variable>
</sensitivity_studies_config>
```

There is no limit on the amount of errors that can be introduced at the same time.

2.5.4. Image Validation Test file

In order to process data for the Image Validation Tests, a Near Field version of the G-matrix generating algorithm is implemented in the Image Reconstruction Module of the L1PP. The selection of wether using the Near Field Algorithm is done through the L1PP configuration file (see Section 2.5.1).

The IVT configuration file, *ivtConfiguration.xml*, should contain the configuration of the room walls where the Image Validation Tests are performed. These are given in the centre of the instrument coordinate frame (in millimetres). An example of the IVT configuration file (*ivtConfiguration.txt*) is provided below. The Number of Planes (i.e. room walls) is an arbitrary positive integer, and each wall is defined through a List of Points. Exactly three points are needed to define each plane and each Point is defined through its X, Y and Z coordinates. In creating the *ivtConfiguration.xml* file care must be taken so that the three points defining each plane are not collinear, since a plane cannot be defined in this if they are.

```
<?xml version="1.0"?>
<IVT_File>
  <List_of_Planes count="1">
    <Plane>
      <List_of_Points count="3">
        <Point>
          <X unit="mm">+10000.000</X>
          <Y unit="mm">+00000.000</Y>
          <Z unit="mm">+10000.000</Z>
        </Point>
        <Point>
          <X unit="mm">-10000.000</X>
          <Y unit="mm">+00000.000</Y>
          <Z unit="mm">+10000.000</Z>
        </Point>
        <Point>
          <X unit="mm">+00000.000</X>
          <Y unit="mm">+10000.000</Y>
          <Z unit="mm">+10000.000</Z>
        </Point>
      </List_of_Points>
    </Plane>
  </List_of_Planes>
  <Room_Temperature unit="K">+0293.690</Room_Temperature>
</IVT_File>
```

2.5.5. Strip Adaptive Processing

Although in itself Strip Adaptive Processing does not require a special configuration file, it is mentioned here for all the clarifications that are needed in its respect.

Strip Adaptive Processing is automatically initiated by L1PP when it detects that the AUX_APOD## Auxiliary file applicable is of type AUX_APOD99. This file contains the coefficients for the 2D Kaiser Apodisation Window as a function of the SMOS beam deformation.

Additionally, L1PP takes the value of the desired ground circular pixels (in km) from the L1PP configuration file (tag "Strip Adaptive Processing Pixel Radius").

It must be mentioned that the model implemented is a "scientific prototype", so the performances are much worse than nominal processing (i.e. Blackman window). Further studies in the frame of the project will determine the need for streamlining the performances.

2.5.6. J-matrix Compression

The J matrix is compressed and expanded according to the baseline weights ADF (BWGHT). It can be compressed and expanded both in rows and, if needed, in columns. The user is responsible to set any desired baseline to 0 in the ADF.

When the L1PP processes a scenario that generates a J+ matrix, these baselines will be removed from the inverted J+, effectively removing them from any future Image Reconstruction processing that uses this new J+ (please refer to [AD.4] for more detailed information).

2.5.6.1. Compression in Rows

Each row of the matrix corresponds to a visibility formed by a pair of LICEFs, and, if they marked in the BWGHT ADF, the corresponding J-matrix line(s) should be removed.

2.5.6.2. Compression in Columns

The columns of the J-matrix are the *(u,v)* pairs that form the number of non redundant visibilities.

For dual polarisation there are 1395 pairs of non redundant visibilities for each group of correlations (Re (H), Im (H), Re(V) and Im(V)). In addition, each polarisation has the zero baseline at *(u, v) = (0, 0)* set manually.

For each pair of LICEFs its *(u,v)* coordinates are computed and the star redundancy is filled, resulting in the map shown in the following figure.

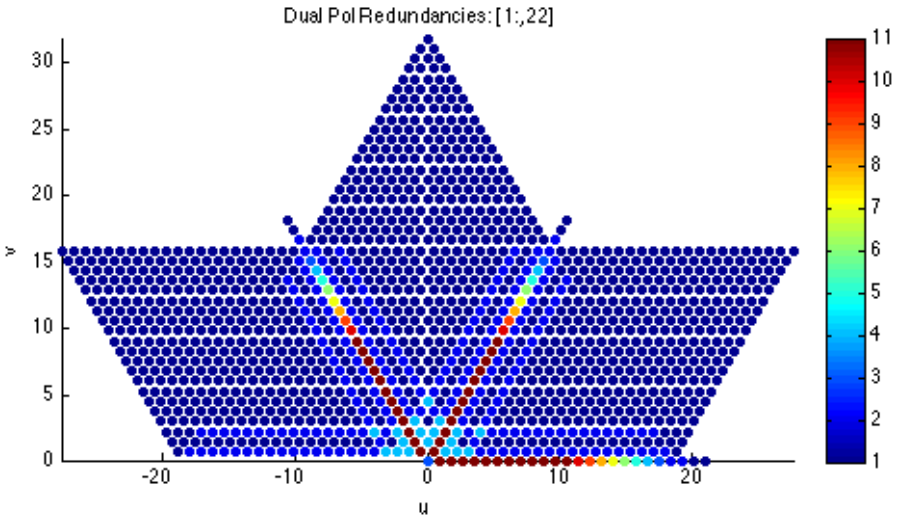


Figure 2 - (u,v) redundancies in dual polarisation for all baselines - scaled image

If the baseline that is to be removed corresponds to a *(u,v)* point with redundancy equal to one, then the corresponding column(s) needs to be removed as well.

In full polarisation there are 2791 non-redundant baselines from a total of 3303. This means that the degree of redundancy is significantly different that in the dual case and that the probalility of excluding an *(u,v)* point is higher in full pol. The next figure shows the redundancies in the *(u,v)* star for full pol.

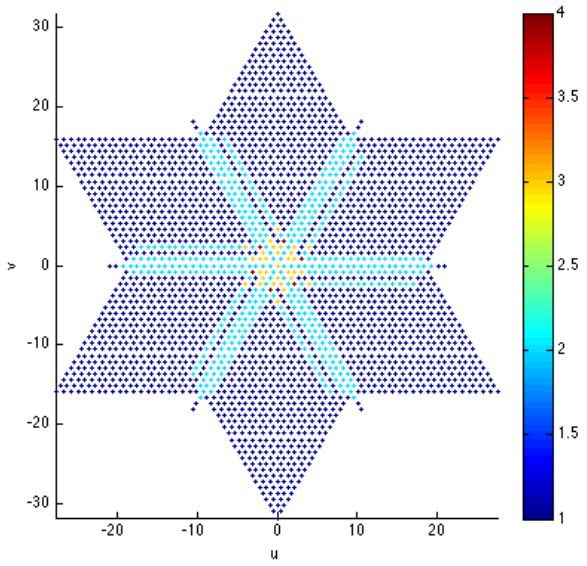


Figure 3: (u,v) redundancies in full polarisation for all baselines

2.6. Known Limitations and Bugs

The current implementation of the prototype presents the following limitations/bugs:

- Configuration of the SMOS products processing order by the Orchestrator is not possible;
- Scientific validation of the Full Polarisation mode processing was not yet fully achieved;

2.7. Degree of Portability

The L1PP has been tested on the following environments:

- RedHat WS4 Update 4 (Linux);
- SUSE 9.3 and 10.3 (Linux);
- Ubuntu 7.4, 7.10 and 8.4 (Linux)

- Darwin 8.10.1 (Mac OS X 10.4.10). Although an SPR has been reported in this last platform related to a problem reading variable array DPGS products with the XML RW API.

More operating systems shall be added as the multi-platform testing proceeds.

3. PROCESSING PROCEDURES

The following sub-sections describe a set of procedures required to process l0, l1a and l1b data.

Please note that as of v1.3.0, the L1PP SW is also able to process .XBAND files from the EGSE platform, and even .DBL files created by the FEP system. To process any type of these files, simply place them in the input directory. L1PP will transform them into L0 files and continue the processing from there.

The header of these L0 products will only contain the minimum information needed for a successful processing (i.e. sensing times and Number of MDR), as the objective here was not to replicate the L0 Processor at the entrance of the L1 Prototype but to provide a simple interface to process real instrument data.

3.1. L0 Data

In order to process L0 data files with the L1PP, the user shall perform the following steps:

1. Start the GUI;
2. In case there are configuration files, select one and load it. If the configuration file exists and no changes are to be performed, jump to step 7;
3. Configure the directories' path (input – l1a,-in, l1b-in, 1c-in -, output – l1c-out-, processed data,etc);
4. Select the processor starting level in the corresponding tab (see section 2.4.1);
5. Select the outputs to be produced (l1a, l1b or l1c data) in the corresponding tab (see section 2.4.1);
6. Save the configuration file;
7. Place input L0 data in the input directory[11];
8. Start the prototype by pressing the "Run" button (under "Run" menu).
9. A progress bar is displayed during the processing. During this time, the user may view the log generated by pressing the button "View Log" in the main window (see section 2.4.1).

The prototype will be continuously fetching new data files that you may place in the input directory; therefore if you have new L0 data files it is only needed to copy them to the input directory and the prototype shall start processing them automatically. If at any time the user wishes to stop the processing, the stop button should be pressed.

[11] There is a set of test data available. A description of this data may be found in [SVP]

Every time a L0 data file is processed, it will be moved to the directory specified in step 3 described above.

Log files are produced as well and you may view them either by using the GUI (see section 2.4.1) or by accessing the directory logs and opening the txt files directly.

Note: as an alternative to the above steps description, the user may consider using the command line:

1. Place L0 input files in any directory;
2. Update the configuration file by editing the corresponding fields (see section 2.5.1);
3. Start the prototype from the command line (see section 2.4).

3.2. L1a Data

In order to process L1a data files with the L1PP, the user has two options:

1. The user puts L1a data in the L1b input directory and then executes the L1PP;
2. The user followed steps described in the previous section and L1a data are automatically processed;

The following list describes what has to be done in the first scenario:

1. Start the GUI;
2. In case there are configuration files, select one and load it. If the configuration file exists and no changes are to be performed jump to step 7;
3. Configure the directories' path (input, output, processed data, etc);
4. Check L1a to L1b processing checkbox;
5. Select the outputs to be produced (l1a, l1b or l1c data) in the corresponding tab (see section 2.4.1);
6. Save the configuration file;
7. Place input L1a data in the input directory[12];
8. Start the prototype by selecting "Run" from the "Run" menu;
9. A progress bar is displayed during the processing. During this time, the user may view the log generated by pressing the button "View Log" in the main window (see section 2.4.1).

The prototype will be continuously fetching new data files that you may place in the input directory; therefore if the user wants to process new L1a data files, they should be copied to the input directory

[12] There is a set of test data available. A description of this data may be found in [SVP]

and the prototype shall start processing them automatically. If at any time the user wishes to stop the processing, the stop button should be pressed.

Every time a L1a data file is processed, it will be moved to the directory specified on step 3 described above.

Log files are produced as well and you may view them either by using the GUI (see section 2.4.1) or by accessing the directory logs and opening the txt files directly.

Note: as an alternative to the above steps description, the user may consider using the command line:

1. Place L1a input files in any directory;
2. Update the configuration file by editing the corresponding fields (see section 2.5.1);
3. Start the prototype from the command line (see section 2.4).

3.3. L1b Data

In order to process L1b data files with the L1PP, the user has two options:

1. The user puts L1b data in the L1c input directory and then executes the L1PP;
2. The user followed steps described in the previous section and L1b data is automatically processed.

The following list describes what has to be done in the first scenario:

1. Start the GUI;
2. In case there are configuration files, select one and load it. If the configuration file exists and no changes are to be performed jump to step 7;
3. Configure the directories' path(input, output, processed data,etc);
4. Check L1b to L1c processing checkbox;
5. Select the outputs to be produced (l1a, l1b or l1c data) in the corresponding tab (see section 2.4.1);
6. Save the configuration file;
7. Place input L1b data in the input directory[13];
8. Start the prototype by selecting "Run" from the "Run" menu;
9. A progress bar is displayed during the processing. During this time, the user may view the log generated by pressing the button "View Log" in the main window (see section 2.4.1).

[13] There is a set of test data available. A description of this data may be found in [SVP]

The prototype will be continuously fetching new data files that you may place in the input directory; therefore if the user wants to process new L1b data files, they should be copied to the input directory and the prototype shall start processing them automatically. If at any time the user wishes to stop the processing the stop button should be pressed.

Every time a L1b data file is processed it will be moved to the directory specified on step 3 described above.

Log files are produced as well and you may view them either by using the GUI (see section 2.4.1) or by accessing the directory logs and opening the txt files directly.

Note as an alternative to the above steps description, the user may consider using the command line:

1. Place L1b input files in any directory
2. Update the configuration file by editing the corresponding fields (see section 2.5.1)
3. Start the prototype from the command line (see section 2.4)

4. TEST TOOLS

As part of the SMOS L1 Prototype activities, some test tools have also been created to generate L0 test data under different conditions. These tools are currently delivered in the prototype directory tools.

The following chapters describe them in detail and what are the procedures to use them.

4.1. SMOS Data Converter

This tool is available for download directly from the Tools section in the L1PP Webpage after accessing as a registered user. It is a binary data converter from EGSE (XBAND) and FEP data into L0 data both L1PP and DPGS formatted.

The following conversions are possible with the tool:

- <1> Convert XBAND files into L1PP L0 files
- <2> Convert XBAND files into DPGS L0 files
- <3> Convert XBAND files into FEP
- <4> Convert FEP files into L1PP L0 files
- <5> Convert FEP files into DPGS L0 files
- <6> Convert FEP files into XBAND

4.1.1. Procedure

1. Run the executable file using the command *./SMOSDataConverter [option] [path].*
2. The option can be any of the numbers presented above
3. The path needs to point to a valid path where files to be converted are found. The tool will generate within the same path the required output files

Note: The source code should be executed in a Linux 32 or 64bits OS due to some specific commands not found in Windows.

5.4. ANNEX: BREAKPOINTS FORMAT

As seen previously in Section 2.4.1.2, the L1 Processor Prototype is able to generate ASCII breakpoints, in order to ease the intermediate validation as well as the analysis of the final results.

The generation of different types of breakpoints can be toggled "On" or "Off" in the GUI, in the "Edit\Preferences" window. Currently the breakpoints are divided in the following groups:

- Level 1A Breakpoints;
- Foreign Sources Correction (FS) Breakpoints;
- L1B Scenes Breakpoints;
- L1C Scenes Breakpoints;
- L1C Browse Breakpoints.

The following sections list the types of breakpoints and corresponding format by group.

5.1.4.1. L1A Breakpoints

L1a breakpoints are activated in the L1PP configuration file, or through the L1PP User Interface by activating the flag "Print L1A Breakpoints". If this flag is set, a set of breakpoint files will be created in the directory $L1PP_ROOT/breakpoint:

- counts_****.txt – file with the correlator counts coming from the ASICS;
- mu_real_****.txt – real part of the computed complex correlations;
- mu imag_****.txt – imaginary part of the computed complex correlations;
- mkj_****.txt – quadrature corrected correlations;
- gkj_****.txt – values of the FWF function at the origin;
- calibvis_****.txt – values of the calibrated visibilities.

All L1a breakpoints are in ASCII format and one version of each breakpoint file will be generated for each processed scene.

In the following tables, the format of the L1a breakpoint files is presented:

Table 9: Correlator raw counts breakpoint file

Name	Size	Line format	Separator	Comment
counts_****.txt	72 x 72 integers	72 integers	Blank space	The file contains the 72x72 matrix with the raw counts from the ASICS

Table 10: Complex correlations, real part, breakpoint file

Name	Size	Line format	Separator	Comment
mu_real_****.txt	72 x 72 floats	72 floats	Blank space	The file contains the 72x72 matrix with the real values of the complex correlations

Table 11: Complex correlations, imaginary part, breakpoint file

Name	Size	Line format	Separator	Comment
mu_imag_****.txt	72 x 72 floats	72 floats	Blank space	The file contains the 72x72 matrix with the imaginary values of the complex correlations

Table 12: Quadrature corrected correlations breakpoint file

Name	Size	Line format	Separator	Comment
mkj_****.txt	72 x 144 floats	144 floats	Blank space	The file contains a representation of the 72x72 matrix with the quadrature corrected correlations. Odd columns have the real part, even columns have the imaginary part.

Table 13: Fringe Wash Function breakpoint file

Name	Size	Line format	Separator	Comment
gkj_****.txt	72 x 144 floats	144 floats	Blank space	The file contains a representation of the 72x72 matrix with the FWF values. Odd columns have the real part, even columns have the imaginary part.

Table 14: System Temperatures breakpoint file

Name	Size	Line format	Separator	Comment
t_sys_****.txt	4 x 2556 floats	4 floats	Blank space	Line format: T_sys for receiver j; T_sys for receiver k; Real part of Mixed Baselines Factor (Lambda_kj); Imaginary part of Mixed Baselines Factor (Lambda_kj);

Table 15: Calibrated visibilities breakpoint file

Name	Size	Line format	Separator	Comment
calibvis_****.txt	72 x 144 floats	144 floats	Blank space	The file contains a representation of the 72x72 matrix with the calibrated visibilities. Odd columns have the real part, even columns have the imaginary part.

Please note that the file format is different between complex correlations, where real and imaginary parts are separated in different files, and for the other breakpoints with complex data, where the real and imaginary parts are separated by columns.

For the case of Correlated Noise Injection epochs, only the first 3 type of breakpoints will be generated, using the “cnoise” prefix and the OBET suffix rather than the snapshot ID which is only meaningful for science data.

For the case of Uncorrelated Noise Injection epochs, only the 4th and 6th type of breakpoints will be generated, using the “unoise” prefix, but in this case the snapshot ID is used. The ordering inside are two columns of data, the first column is the real part and the second column is the imaginary part, and the number of lines is 72x72.

~~5.2.~~4.2. L1B and Foreign Sources Breakpoints

Foreign Sources breakpoints are activated in the L1PP configuration file, or through the L1PP User Interface by activating the flags "*Print FS Breakpoints*" and "*Print FS Extra Breakpoints*". If any of these flags is set, a set of breakpoint files will be created in the directory $L1PP_ROOT/breakpoint.

L1b Scenes breakpoints are activated in the L1PP configuration file, or through the L1PP User Interface by activating the flags "*Print L1B Scenes*" and “*Print L1B Scenes BT*” respectively. If any flag is set a breakpoint will be generated per scene, with the (u,v) baselines or (xi,eta) coordinates and the corresponding real and imaginary parts of the Fourier components of the brightness temperature in the antenna frame, or the brightness temperatures themselves obtained through a FFT

The following table shows, in the third column, the breakpoint names implemented in the Image Reconstruction Module of the current L1b baseline.

The first column indicates the location in the L1PP implementation where the breakpoint is taken (see the diagram of Figure 3 in [RD.6]), and the common prefix in the breakpoints filenames.

Location (Prefix)	Description	Breakpoint Name	Format
After INPUT DATA (l1b_ir_input)	L1a Calibrated Visibilities	visibs_<snapshot_id>_<pol>.txt	2556 x 2 float [real imag]
	Baseline Weights	weights_<pol>.txt	4695 x 1 float
	Average Backlobes Antenna Patterns	patt99.txt	16384 x 14 float [k1 k2 xi eta re(F^X) im(F^X) re(F^Y) im(F^Y) re(X^X) im(F^X) re(X^Y) im(X^Y)]
	G-matrix NIR rows	gMatrixNIR.txt (alternate columns for H and V)	2*16384 x 2 float [NIR-AB] [NIR-BC] [NIR-CA]
	Visibilities for a constant 1K scene	visibs1K_<pol>.txt	4695 x 1 float
After PREPROCESSOR (l1b_ir_prepr)	Xi, Eta in the Hexagon (Dual & Full)	xiEta_hexagon.txt	16384 x 2 float [xi eta]
	(u,v)-baselines and Apodisation Window	baselines_apodisationWindow.txt	1396 x 3 [*u v* apodWin]
	Flat Target Product Visibilities	visibs_ftt.txt	4695 x 2
	Flat Target Product Temperatures	temp_ftt.txt	1 x 1
	L1b Calibrated Visibilities	visibs_<snapshot_id>_<pol>.txt	4695 x 1
	Xi, Eta in the Unit Circle	xiEta_unitCircle.txt	16384 x 2 [xi eta]

In FOREIGN SOURCES MODULE (l1b_fs)	Backlobes Brightness Temperatures	backlobes_tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	Sky Brightness Temperatures in the Hexagon	sky_tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	1K Land Brightness Temperatures in the Hexagon	land1K_tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	Sea Brightness Temperatures in the Hexagon	sea_tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	Sunglint Brightness Temperatures in the Hexagon	sunGlint_tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	Sun Direct Temperature and xi-eta coordinates	sun_tempsSpace_<snapshot_id>_<pol>.txt	xi,eta, BT
	Moon Direct Temperature and xi-eta coordinates	moon_tempsSpace_<snapshot_id>_<pol>.txt	xi,eta, BT
After FS (l1b_ir_frStc)	Calibrated Visibilities with Sun, Moon and NIR backlobes removed	visibs_<snapshot_id>_<pol>.txt	4695 x 1
After EM (l1b_ir_erMtg)	Delta Temperatures removed from the original image (Sky, Sunglint, Gibbs)	tempsSpace_<snapshot_id>_<pol>.txt	16384 x 1
	Delta Visibilities (without Sun, Moon, Backlobes, FTT, Sky, Sunglint and Gibbs) before applying weighting matrix	visibs_<snapshot_id>_<pol>.txt	4695 x 1
After RECONSTR. (l1b_ir_recon)	Delta Visibilities (same as above) after applying weighting matrix	visibs_<snapshot_id>_<pol>.txt	4695 x 1
	Temperatures in Frequency space (Dual & Full)	tempsFreqs_<snapshot_id>_<pol>.txt	2791 x 1
	Temperatures in (xi,eta)-space (IFT of tempsFreqs) (Dual & Full)	tempsSpaceIft_<snapshot_id>_<pol>.txt	16384 x 1

Table 16: L1B Scenes Breakpoints file format

Notation:

snapshot_id: snaposhot identification number;

pol: H, V or HV;

The L1b breakpoints are comma separated values (csv) files.

Since L1PP v3.5.0, an RFI Histogram Map can also be produced when the appropriate configuration is used. The format of this RFI Histogram is the following:

Name	Size	Line format	Separator	Comment
RFI_Histogram.dbl	5*[1440 x720] floats	N.A. (binary file)	N.A. (binary file)	Each probable source is defined as (Number of hits, Lat., Long, T_B^X, T_B^Y.). The size of the RFI_Histrogram.dbl is hardcoded in L1PP as [1440x720], which corresponds to a cellsize of [0.25x0.25] degrees.

5.3.4.3. L1C Scenes Breakpoints

L1c Scenes breakpoints are activated in the L1PP configuration file, or through the L1PP User Interface by activating the flag "*Print L1C Scenes*". If this flag is set a breakpoint will be generated per scene, with the latitude, longitude and real an imaginary part of the pixel brightness temperature:

Table 17: L1C Scenes Breakpoints file format

Name	Number of Lines	Line format
l1c_temp_<*snapshot_id*>.txt	Number of Pixels in the Snapshot	Latitude (float), Longitude (float), BT Real (float), BT Imaginary (float), AF-FOV Flag (0 or 1), LandSea Flag (1, 2 or 3), Xi (float), Eta (float), Radiometric Accuracy (float), Border Flag (0 or 1)

The L1c Scenes breakpoints are comma separated values (csv) files.

5.4.4.4. L1C Browse Breakpoints

L1c Browse breakpoints are similar to the L1C Scenes breakpoints in terms of format, but are swath oriented, instead of breakpoint oriented. They can be activated in the L1PP configuration file, or through the L1PP User Interface by activating the flag "*Print L1C Browse*". If this flag is set a breakpoint will be generated per scene, with the latitude, longitude and real an imaginary part of the pixel brightness temperature:

Table 18: L1C Browse Breakpoints file format

Name	Number of Lines	Line format
l1c_<*land/sea*>_<*polarisation*>_<dummy_*snapshot_id*>.txt	Number of Pixels in the Browse products	10 floats: - *Lat (float)* - *Lon (float)* - *Real BT (float)* - *Imag BT (float)*

The L1c Browse breakpoints are comma separated values (csv) files.

6.5. L1PP DIRECTORY STRUCTURE

```
l1pp-6.0.0
├──bin
├──breakpoint
├──config
│   ├──iono_models
│   │   ├──IGRF
│   │   └──IRI
│   ├──product_headers
│   └──xml_schemas
├──data
│   ├──adf-dpgs
│   ├──adf-eef
│   ├──l1a-in
│   ├──l1b-in
│   ├──l1c-in
│   ├──l1c-out
│   ├──processed-data
│   └──unprocessed-data
├──dpgs_interface
│   ├──lib64
│   │   └──LINUX
│   ├──smos
│   │   ├──config
│   │   ├──products
│   │   ├──schemas
│   │   └──tmp
│   └──xml_rw_api
│       └──headers
├──external_libs
│   └──lib64
│       └──LINUX
├──lib
├──lib64
│   └──LINUX
├──logs
└──scripts
```

The directory structure is populated with data from the Install Kit, from Test Data Package and from the ADF Package.