CGS Constraints and Rules for PPS

	Responsibility - Company	Date	Signature
Prepared by S. MARCHADOUR		15/05/02	
Approved by (Technical) F. DIDELOT	Technical Manager - ALCATEL	15/05/02	
Approved by (Quality) M. QUERE	Quality Manager - ALCATEL	21.5.02	
Authorised by D. JAN	Project Manager - ALCATEL	15.5.02	

Configuration Item number:
if applicable

Summary:

Document submitted to EUMETSAT for:

☒ **Approval** ☐ **Review** ☐ **Information**

Signature

ALCATEL

CHANGE RECORDS

ISSUE	DATE	§: CHANGE RECORD	AUTHOR
1.0	15/05/01	First issue	
2.0	07/09/01	Issue including : comments from EPS-ASPI-MN-290 RIDs from CGS PDR : - RID#28 in §3.2.3 - RID#33 in §3.3 - RID#34 in §4.1 - RID#35 in §4.2 - RID#273 in §3.1 - RID#283 in §4.3.3.7 - RID#1156 taken into account (note that the CGS Constraints and Rules on PPS is not an applicable document for Alcatel's subcontractors) - RID#1254 in §2.3 - RID#1256 in §2.3.2 - RID#1260 in §5.3 - RID#1262 in §3.2.3 - RID#1265 - RID#1269 in §2.3.2 - RID#1455 Action 4 from EPS-ASPI-MN-309. -	SM
2.1	30/01/02	- Paragraph 3.2 on the error handling added - §3.3.4 and §3.3.5 : no MPH is needed for the auxiliary data file, only a naming covention is defined - §4.3 updated according to internal review - PPF-0645 : deleted (the PPF is synchronised through the step command) - PPF-0680 : deleted (STOP command not to be used as a SUSPEND) - PPF-811 : added	SM
2.2	15/05/02	Event charts in section 6.2 have been moved to IASI ICD. Correction of file naming rule in section 4.3 Rules about context files naming have been added in section 4.3.5.2 Addition facility delivery and installation constraints §6.3 and PPF contextual information §6.5.	FF

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE IS INTENTIONALLY LEFT BLANK

1. INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

This document is the Constraints Requirement Specification for Product Generation Element (PGE) to Product Processing Software (PPS).

1.2 DOCUMENT STRUCTURE

This document contains 10 chapters, which are structured as follows:

Chapter 1 This chapter.

Chapter 2 provides some background and context to the PPS.

Chapter 3 gives an introduction to the constraints and rules.

Chapters 4 to 9 gives the EPS PPS constraints and rules.

Chaoter 10 contains a glossary of abbreviations, acronyms and terms.

1.3 OPEN ISSUES AND ASSUMPTIONS

N/A

1.4 APPLICABLE AND REFERENCE DOCUMENTS

Document Title	Identifier	Internal Reference
Product Processing Software to Product Generation Element	EPS/SYS/IRD/980255	[AD2]
EPS Generic Product Format Specification	EPS/GGS/SPE/96167	[AD3]
Glossary of terms and abbreviations list	EPS-ASPI-LI-0010	[AD4]
EPS Product Convention Document	EPS/SYS/TEN/990007	[AD5]
CGS Facility Common Design Requirements	EPS-ASPI-SP-0103	[AD8]

THIS PAGE IS INTENTIONALLY LEFT BLANK

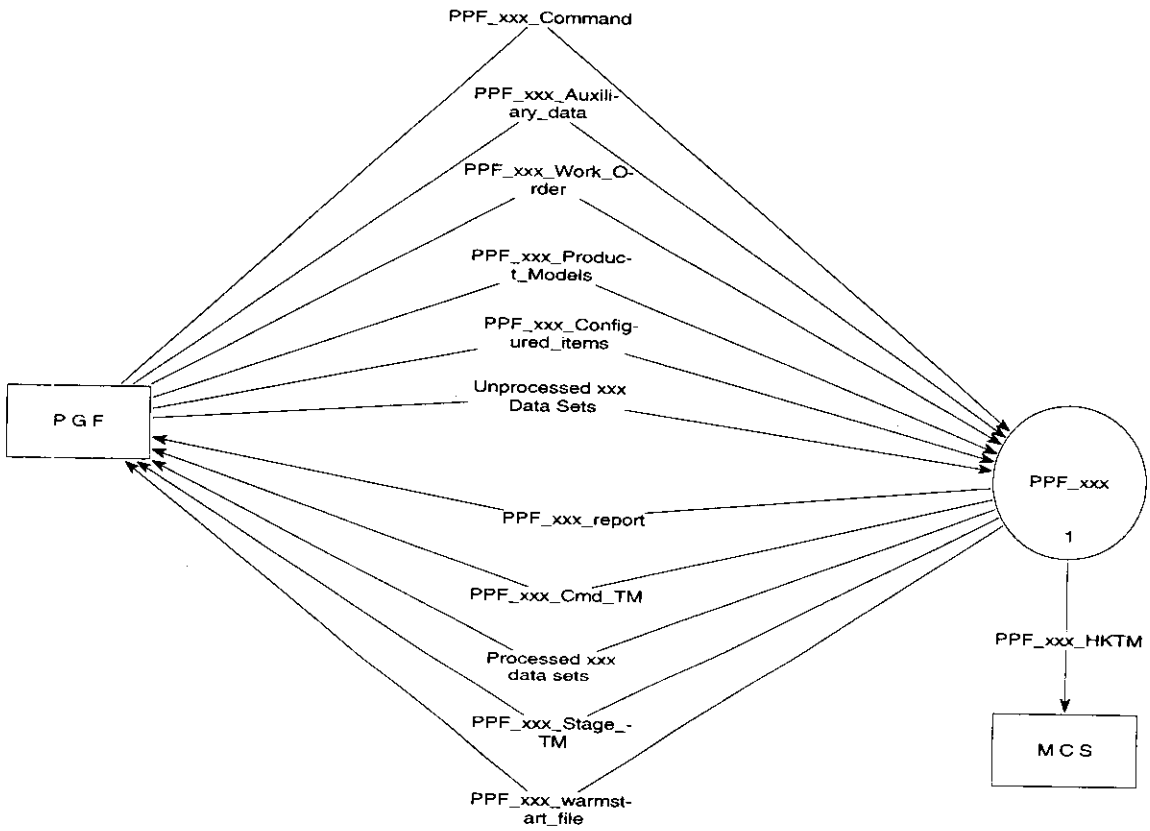
2. GENERAL DESCRIPTION

2.1 PPF CONTEXT

The PPF interfaces physically only with the PGF and the MCS Local Agent (MLA).

Data flows input or output between PPFs and PGF are identified through separate interfaces flows summarised on the following diagram.

Figure 2.1-1 Generic PPF context diagram



PPF are first fed up with needed activation parameters and auxiliary input data. They are then started by the PGF host structure. They receive the data to process and generate progress status and reports up to the end of corresponding production. They finally make available higher level products on their file system. The PGF is interfaced with the PPF through a set of generic PGE services offering a standard way to prepare, start, control and collect the results from any PPF S/W execution context.

2.2 PPF OVERVIEW

The production facilities architecture relies on the isolation of instrument processing software in PPF (Product Processing Facility) and a high level of automation with :

- the pipeline processing,
- the isolation of instrument processing software in PPF (Product Processing Facility).

The PPF is one element within the EPS Core Ground Segment. It is one stage in the overall processing chain converting telemetry data (NOAA and MetOp data) into products that can be interpreted by end users. It interfaces mainly with one other facility, the PGF (Product Generation Facility), which supplies it with data, algorithms and detailed instructions, and receives back the completed products. Each PPF is oriented on specific instrument data processing.

The PPFs are currently the following:

- The PPF_ATOVS is made of 2 items:
  - PPF_ATOVS_L1 is responsible for the generation of AVHRR, HIRS, MHS, AMSU-A, level 1 products from level 0 product of Metop or NOAA satellite. It processes, on-line calibrates and controls the quality of its production.
  - PPF_ATOVS_L2. processes the level 2 products (common to the 4 instruments).
- PPF_IASI_L1, supplied by CNES
- The PPF_IASI_L2 is responsible for the generation of level 2 product, it takes IASI level 1c as input.
- The PPF_GOME, PPF_GRAS, PPF_ASCAT are responsible for the generation of level 1 (corresponding level 2 are not produced in the ground segment). The instruments are on board the Metop satellite only.

The other functions being responsible for the upstream handling of the downlinked instrument data, the supervision of product generation, the administration of products to end users, the engineering support to product generation and services to users of the CGS are described in the CGS Architecture Document.

Each Product Processing Facility is one of the facilities which together make up the Product Processing Software (PPS). The several instances of PPF, with FEP, PGF and DIF facilities constitute the Processing facilities. The purpose of the Product Processing Facility within the Processing facilities is to generate a product above level 0 from an instrument on board the satellite Metop or NOAA. The main interface to it is that to the PGF, which supplies it with all of the necessary information to perform its task.

Several instances of the same PPF exist in the two Ground System of the EUMETSAT central site. These instances may have different hardware and software configurations. A series of building block components will be defined by the PPF architecture such that the required hardware and software configurations can be generated from generic units.

Two activation mode are available for PPFs :

- Dynamic mode : the PPF is started for each L0 slice or L1 PDU to be processed. At the end of the L0 slice or L1 PDU processing, the PPF stops by itself,

- Static mode : the PPF is started for each dump. A production request is afterwards sent through a specific command for each L0 slice or L1 PDU to be processed. At the end of the dump the PPF stops by itself.

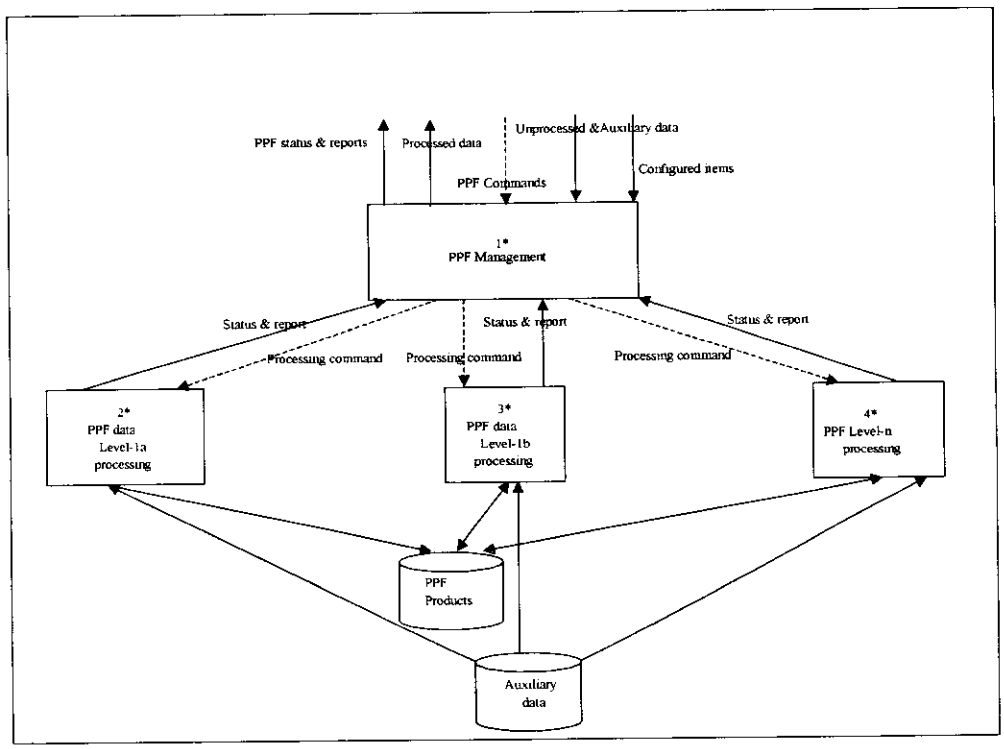
For the PPFs ATOVS, GOME, ASCAT, GRAS and IASI L2 a dynamic allocation mode will be used.

For the PPF IASI L1, a static allocation mode will be used.

2.3 PPF ARCHITECTURE

The architecture of all the PPF follows the same approach whatever the allocation mode as depicted below in a simplified way.

Figure 2.3–1 A PPF generic software architecture



Note that the blocks 2, 3 and 4 are depicted as an example and are PPF dependent : one block by product level processed (e.g. for IASI L1 : one block for IASI L1a, one for L1b and one for L1c).

Each PPF contains a PPF Management process (also called MP) in charge of managing the instrument processing tasks, as well as communication with the PGF.

- This process implements the following operations commanded by PGF: start, step, stop, break, abort, resume and suspend.

The Monitoring and control API provides an access to the Monitoring and control local agent and guide the architecture of the MP (Main Process) embedded in the PPF software.

The PPF is started on a specific *working root directory* giving access to a whole tree where it finds its different inputs and generate its outputs. Moreover, during the processing, the PPF can receive commands (e.g. stop) and will send regular status on asynchronous channels.

Each process of the PPF sends to the MP status information (change in process stage, process stage finished, intermediate report ready, products completed, error during processing).

The MP then reports the status information to PGF and MCS local agent.

The PPF computers have got each, all the instrument S/W installed. Their profiles can be easily set up to process each, a combination of the following states :

- all kinds of products,
- set of instruments specific products,
- separate levels products (ie L1 or L2 or L1/L2)
- separate satellite products.

The configurability is allowed by the use of Production rules.

A single computer can execute more than one instrument processing at a time, due to its multiple processors availability. Consequently, the PGF can activate a number of parallel productions onto the a same PPF computer, with respect of the number of processors it includes and to the number of CPUs allocated to each production.

2.3.1 Dynamic mode

While the PGF is liable to run a PPF on different computers at each new execution, each PPF will usually run on the same one for long periods. It is the responsibility of the PGF to transfer context files and other configuration files or auxiliary data to any relevant computers according to PPF allocation and activation.

For PPF in dynamic mode, only one CPU is allocated to the PPF at each activation.

When a parallelisation is needed to meet the NRT requirements (i.e. when the computation duration is longer than the sensing duration of the data to process), the PGF is able to implement a time symmetry paralelisation scheme, managing several execution of the same PPF software in parallel on different data set.

Note that in this case (time symmetry paralelisation)the context files to be transferred between one PPF activation for the next PPF activation is forbidden, as far as a new PPF activation will be started before the previous one has finished.

2.3.2 Static mode

Each PPF is allocated a configurable number of processors at each activation within the ones available on the node; the PPF is then in charge of managing the parallelisation of the algorithm.

IASI L1 will be activated in static mode, with 4 CPUs allocated at each activation.

2.4 PPF ENVIRONMENT

The association of a PPF software to a computer node is enforced by the creation of a directory tree specific to the PPF.

There is then no difficulty in installing several PPFs on the same node providing a good flexibility. (e.g. ASCAT and ATOVS L1 may share the same node, each requiring only 1 CPU).

This also provides a useful redundancy when spare computer are added.

All the PPF computers are interconnected to the PGF servers via the IBM SP2 switch.

The throughput of the switch is at least 100 Mbyte/s point to point, to be compared to a maximum disk throughput of 10 Mbyte/s.

It guaranties that a file transfer from one node to another is not slower that a local file copy. This promote the use of a file system global to PGF and PPF (IBM GPFS).

This design approach results in a very simplified hardware architecture made of the following main hardware elements (quantity for only one GS):

- **PPF SP2 nodes** : 10 identical nodes IBM (Power3-II) with 4 CPU each.
- **SP high speed switch** : the switch provides a 2x150 MB/s (one input and one output) between nodes.

Figure 2.4–1 PPF SP2 Node Configuration

Model	SP2 Thin Node
CPU Type	Power3-II
Mhz	375
L1 Instruction Cache	32 KB
L1 Data Cache	64 KB
L2 external Cache	8 MB
CPU Nb per Node	4
SPECint95	22
SPECfp95	46
Memory	2 GBytes (shared between 4 CPUs)
Maximum Memory Capacity	8 GBytes
System Disks	2x9 GBytes
Available PCI Slots	2
Internal Switch Throughput	2x150 MBytes/s

3. INTRODUCTION TO REQUIREMENTS

3.1 PGE SERVICES

The PPF is implemented as a distinct facility, to be introduced and configured into PGF host structure, and to be connected with generic standard interfaces, that the PGF will manage through PGE services.

The PGF is the main facility interfaced with the PPFs. No other CGS facility (i.e. FEPs, DIF, QCF) have got any visibility on the PPFs executions. All the PPFs are activated and managed with the same set of PGF interfaces, whatever the instrument processing they are in charge off. It provides a lot of flexibility for the design and for the production (for example, the use of stubs or restricted software when definitive ones are not yet available).

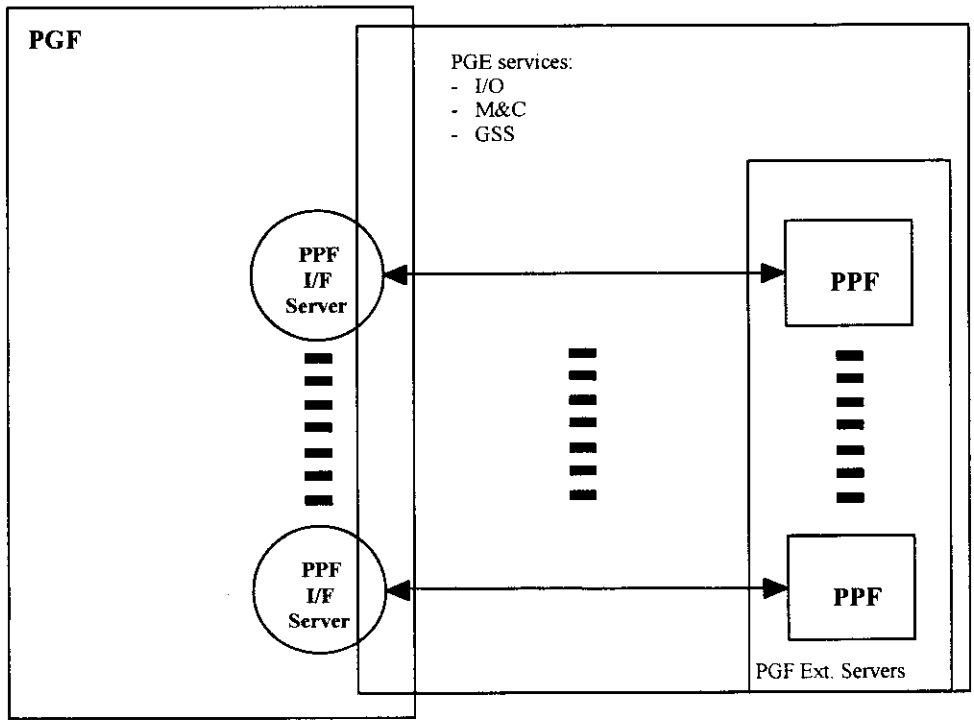
The PPF computers have got each all the PPF instrument software installed (the PPF_ATOVS, PPF_IASI, PPF_ASCAT, PPF_GOME, PPF_GRAS). Their profiles can be easily configured in view to be able to process each

- All kinds of products,
- Sets of instruments specific products,
- Separate levels products (i.e. L1 or L2),
- Separate satellite products.

Each entry from the PGF internal work list can be constrained by external events, before being processed. For example,

- ATOVS level 1 production needs HIRS, AMSU, MHS and AVHRR level 0 with same sensing coverage as inputs. The ATOVS L1 PPF activation shall wait for this Level 0 files availability. The maximum waiting time shall be configured within the PGF, in view to be able to start processing ATOVS L1 with one or many L0 missing.
- IASI L1 is requiring AVHRR L1 for L1c production. The PGF shall wait as long as possible (time-out) the availability of AVHRR L1b products needed for IASI L1 production before requesting a higher level production from a IASI L0 slice. To process the IASI L0 slice N, the AVHRR L1b PDU N and PDU N+1 are required by the PPF.

Figure 3.1-1 PGE services



The PGE services provide services to the PPF. The different categories of data crossing the interface between the PGE and the PPFs are all accessible to the PGF :

- Dataflow of science data,
- Dataflow of auxillary data,
- Flow of Control – Commands etc,
- Flow of other data

The PGE regroups three types of functions :

- Monitoring and Control (MCS). This interface will provide for the exchange of monitoring and control data between the PGE and the PPF. Control functions will cover the need of the PGE to send commands to the PPF. Monitoring functions will be used by the PPF to report to the PGE its internal status. A S/W activation service will be provided to start/stop etc. the PPF and a S/W characterisation service will provide information on the S/W version and configuration.

- Support Services (GSS). The interface will allow the PPF to have access to a pre-defined number of functions in support to its internal processing such as orbit propagation, earth location, time correlation. This will include mathematical/statistical functions from a suitable set of support libraries.

- /O Services (IOS). The interface will allow the PPF to read various types of data and to output it across the interface. This includes static Auxillary data and Dynamic data.

3.2 ERROR HANDLING

Each CGS facility shall manage on a coherent way, the typical error cases which can happen on any interface management (e.g. Errors on messages or data transfers through TCP/IP ...).

In order to keep a system level consistency and non-ambiguity, each error status is allocated a unique key.

Error management rules and error status per facility are provided through [AD2].

3.2.1 Checking the inputs

The program shall check the validity of its input in a thorough way, typically:

- parameters that must be positive or in a reasonable range of value (e.g. a value greater than 10^{+20} is usually not a sensible value while it is not uncommon for an uninitialized variable)

- check that arrays expected to be sorted are actually read in an increasing (or decreasing order).

These checks may be packaged in independent routines and may be called from different thread of execution and even constitute a separate proofing executable.

3.2.2 Numerical errors

NaN: The occurrence of a numerical error (square root of a negative number, 0-divide) shall not systematically trigger the abortion of the processing. The IEEE standard reserves specific codes as results of theses operations: not a number (NaN), infinity (–INF, +INF), and some others. These codes are then propagated down the computation flow so that a single test at the end of each chunk of computation is sufficient to detect an error. An IBM library supplies routines *fp_read_flag(), fp_clr_flag(), fp_set_flag()* for this purpose.

The use of the value NaN actually allows to continue the processing for data that are valid with no increase of the software complexity.

With the same aim of not aborting a computation, of reducing the complexity and avoiding execution speed penalties, the use of signal trapping and exception catching should be restricted to the following cases:

- During debug activities, both signal and exception can be usefully trapped in order to locate the source of an error.
- For inter-processes communications, signals allows efficient solutions.
- At the outer levels of the program, catching exceptions is useful as a last chance for recovery and logging.

Internal checking: More generally, each routine should be protected against improper values of its inputs. C++ supplies the *assert* keyword for this concern, it can be easily simulated in C by the definition of a macro. Such verifications are relaxed in release version by setting a compiler switch.

3.2.3 Data gaps

A data gap (due to lost or corrupted data in the input) is implemented through a single dummy record (one record represents a whole data period). Such missing data may occur in the middle, at the end or at the beginning of the input slice. It shall trigger a degraded processing and a corresponding dummy record in the output products.

When a data gap encompasses a slice boundary, a dummy record is present both at the end of the first slice and at the beginning of the second.

When a data gap encompasses a whole slice, the void slice is not created nor processed.

3.3 PGF INTERFACE

The PPF is started on a specific *working root directory* giving access to a whole tree where it finds its different inputs and generate its outputs (note that certain subdirectories may be located on remote disk shared by several facilities). Then when 2 concurrent run will be launched, they will be allocated separate working root directories and should not interfere.

Moreover, during the processing, the PPF can receive commands (e.g. stop) and sends regular status on asynchronous channels. The data exchanged include:

- The *work order* specifies the time span of the data to process as well as auxiliary files and the type of products to generate (e.g. level 1a and/or 1b).
- The input data: measurements and auxiliary data, are delivered granule after granule by the PGF at a more or less regular pace (a granule is a full-featured product with MPH-SPH-IPRs-data). Level 0 and aux data are each delivered in different subdirectories of the working root directory.

- Context information, managed as aux data to allow implementing warmstarts.
- Commands, received asynchronously on a specific channel.
- Status, sent by the PPF on a regular basis to report on its progress. These status are addressed to the MCS through a call to a library routine (PGE services).
- The output data: higher level products are generated in pace with the delivery of input data (lower level products). Input and output are PPF dependent, e.g. for IASI L1 PPF, input data are IASI L0 product, and output data are IASI L1a, L1b and L1c products.
- Configuration files are available at launch of the PPF and are not changed during the run.

These features are detailed in the following subsections.

3.3.1 Launch & run

The PPF is an executable, launched with command line arguments such as: its working root directory, a mode (nominal versus investigation), and other potential parameters related to the context of execution. Practically, the PPF executable is embedded in a script file used to set potential UNIX environnement variables and to allow extra flexibility.

The number of allocated processors is configurable. For IASI L1 PPF, up to 4 processors will be allocated (configurable).

3.3.2 Work-order (an ASCII file)

The work order, mainly includes a time interval specifying the data window to process, this time interval can encompass several files as well as it can restrict the input data to a subset of the delivered L0 (resp. L1).

3.3.3 Unprocessed L0 files (resp. L1)

They are written to a specific subdirectory in the working root directory at an unpredictable pace. When a production will be requested to a PPF ("step" command) all its LC input will be available. For PPF requesting several L0 files (e.g. ATOVS PPF requesting AVHRR L0, HIRS L0, MHS L0 and AMSU L0) it will manage a degraded mode policy in the case L0 files are missing.

The PPF must process each L0 (resp. L1) as soon as it is available and than the computing resources are available.

L0 (resp. L1) files are assumed to be delivered by the PGF sorted by time with possible gaps. Then if file F starting à T is delivered while no file is available for times anterior to T (and assuming T is in the work-order time span), F will be processed and the missing data given up (after that, if a L0 (resp. L1) with a starting time lower than T is delivered, it will no longer be processed).

L0 files may overlap: The beginning of granule n+1 may have already been delivered in granule n. The size of the overlap (that can be 0 if no overlap is needed) is defined according to the algorithm needs.

Note: the L0 files delivered as input usually encompasses a sensing duration larger than the duration specified in the work-order.

3.3.4 Auxiliary data (secondary inputs)

Auxiliary data are found in a subdirectory of the working root directory or any other directory, according to the path specified in the workorder.

Several files may be available for the same processing time as well as no file may be available. To deal with these cases, the PPF implements a selecting policy which allows it to pick up the file best fitting the current processing.

When the PPF cannot find an auxiliary data suited to its current processing, a default policy is implemented according to each type of aux file – this policy can be, to log an error message and end the processing.Aux data are not modified and can be shared by several facilities.

All aux-data are assumed to be available on reception of a Step command. PGF will wait for these auxiliary data before requesting a production to the PPF (time-out policy implemented at PGF level). When receiving a production request and when not finding the appropriate auxiliary data, the PPF will manage a degraded mode policy.

However, a new auxiliary data file may appear in the aux file directory at any time, it shall be taken into account at the next Step command.

From the system point of view, an auxiliary file is any file whose name comply to auxiliary data file convention. The filename will in particular encode a type and a start of validity date. These 2 fields are the only fields used for management by PGF. Finally, an auxiliary file has a free format specific to its type.

3.3.5 Context data (in/out) - optional

One or several files allowing to save state variables of the processing from an execution to another – these state variables are often sort of long term averages needed for calibration purposes. These files are found at launch time in a subdirectory of the working root directory, they are intended to be modified by the PPF. The need for a context file must be carefully assessed and agreed with Alcatel as it brings an important constraint on the system.

From the system point of view, a context file is an ordinary auxiliary file. Actually, its format is to be defined by the implementation team and its name shall comply to the general rules for aux data file convention. The filename will in particular encode a *type* and a *start of validity* date.Generated products (output)

L1a and L1b files (L2 with regard to ATOVS2) are generated on a time slice basis specific to the PPF, usually the duration of an output product is the same as the duration of the input ones. A file is written as soon as the data compounding it are computed, then a specific status is reported to the PGF mentioning the availability of the file. They are written in a subdirectory of the working root directory.

Conversely to the main products, some files are generated once at the end of the processing for the entire time span of the work-order (or, if the processing has aborted prematurely, from the work-order start to the ending time. Abortion occurs in case of time out on waiting for an input file, in case of command STOP...).

3.3.6 Report (a file in XML format)

A report is delivered in response to each workorder. It includes the actually processed time period, the name of the generated files: L1a, L1b, L2, dynamic aux data, context ... .

There is a single report responding to each work-order.

In case of a non-nominal event leading to a loss in the data generated, a single error message is added.

3.3.7 Commands

Start allows to start the PPF. No effect if already started or suspended.

Step allows the processing of a work-order. A new work-order is issued for each new Step command

Stop terminates prematurely the PPF execution (software shutdown). No other command will be accepted after a stop.

Break terminates prematurely the production related to the previous step command.

Suspend pauses the processing but does not free the resources. It has no effect if the PPF is not started

Resume restart the processing after a supend. It has no effect if the PPF was not suspended

The processing of a stop shall comply with the following:

the PPF responds in less than 15s by a status ack and terminates within 1 mn.

The PPF chooses a time t in [workorder.start, workorder.stop] and set the whole working directory tree in the same state as if the work-order stop had been t. However, already delivered products cannot be cancelled, that is, they must not be modified nor removed from the working directory.

Then it generates a report accordingly. t, the end-of-processing time is mentioned in the

report. When all files are consistently updated, the execution ends (all the processes created on launch are terminated)

Commands are not needed in the standalone version, the processing beginning directly on launch and stop not being possible in this version.

3.3.8 Status

Refer to §4.3.3.

3.4 MODES

Any PPF software has 2 modes of operation:

- **Mode Nominal**: the PPF is getting extracted data, input data and commands from the PGF. Nominal mode can occur as either NRT or off-line (backlog, reprocessing). The difference between these 2 modes is, for the PPF, the use of a different directory of execution along with possibly different inputs. The use of a different directory of execution implies separate warmstart files.

- **Mode Investigation** : the PPF facility is behaving as per operational mode except that it logs more information to the system log file or separate data files (created for extensive intermediate results). This mode is intended for debugging, validation, and verification purposes.

It is expected that a self test will exist that can be run on the PPF that is independent of the PGF. This self test is expected to consist of a simple script chaining up a run of the PPF (possibly along with the PGF simulator) with a run of the direct comparison tool.

For this self test, the PPF takes its inputs in a working root directory present in the delivery, the comparaison is done between the product generated and a reference (factory generated) also part of the delivery. The test return passed or failed.

The aim of the self test is to validate an installation (including the configuration of the host and the validity of the executable generated from the make).

4. HIGH LEVEL REQUIREMENTS

4.1 HIGH LEVEL FUNCTIONAL REQUIREMENTS

PPF-0010	CGS-SYS-4.1.2-0040 CGS-DPG-5.3.5-0010 CGS-DPG-5.25-0040	FUNCT	TEST
The PPF shall encompass all the functions (algorithms and scientific functions, supporting function, product formatting function) required for the generation of the products as required.			
PPF-0030	CGS-SYS-4.1.2-0060 CGS-ONC-5.5-0010 CGS-QCS-4.5.1-0010	FUNCT	ANALYSIS
The PPF shall provide quality controlled and validated products, by supporting the following capabilities: 1) on-line quality control by generation of PCDs 2) on line calibration Note: a PCD (Product Confidence Data) is a number intended to qualify the validity of a measurement or a computation or even a summary standing for a certain sensing duration.			
PPF-0040	CGS-SYS-4.2.1-0070	FUNCT	ANALYSIS
For backlog and reprocessing, the PPF shall support simultaneously at least two official operational versions of the algorithms on the same computer.			
PPF-0050	-	DES	ANALYSIS
Different versions of the PPF software shall coexist on the same computer node, allowing several PPFs of different version to be run concurrently.			
PPF-0060	CGS-DPG-5.3.3.4-0140 CGS-DPG-5.3.5-0100	FUNCT	TEST
The PPF shall be configurable from PGF to support the generation (when relevant) of : level 1a from level 0 , level 1b from level 1a , level 1b from level 0, level 1c from level 0 (IASI L1), level 1c from level 1a (IASI L1), level 1c from level 1b (IASI L1), level 2 from level 1b (ATOVS), level 2 from level 1c (IASI L2), calibration on request.			
PPF-0070	CGS-DPG-5.3.3.1-0030	FUNCT	ANALYSIS

Several instances of the same PPF shall be able to run at the same time on the same computer (each PPF being allocated a different CPU) without a drastic impact on the processing performance.			
PPF-0080	CGS-SYS-4.1.2-0040	FUNCT	ANALYSIS
The products shall follow the common structure defined in [AD3].			
PPF-0090		FUNCT	ANALYSIS
In the case of non-availability of some part of the algorithms specifications during the implementation, the PPF shall use a configurable processing stub simulating the behaviour of the final algorithm in terms of CPU load, I/O load and external interfaces.			
PPF-0100		FUNCT	ANALYSIS
Re-use of non-widely commercial software, developed under contracts other than the CGS development that are protected by intellectual Property Right or limited in their use or distribution, shall be limited to exceptional cases and must previously receive a written approval.			
PPF-0110		FUNCT	ANALYSIS
Inclusion of Public Domain software in the PPF shall be submitted for approbation.			
PPF-0120	CGS-GSW-8.4.1-0090	FUNCT	ANALYSIS
The capacity to perform software diagnostics within the operational environment shall exists. Mainly, in case of abnormal termination, the PPF shall backup the files necessary to reproduce the problem.			
PPF-0140	CGS-DPG-5.3.4-0080	FUNCT	ANALYSIS
The PPF shall make available to the PGF a report providing detailed information on generated products as well as on-line quality results (information to embed in the status).			
PPF-0160		FUNCT	ANALYSIS
The PPF shall use templates for the file names of the output products as defined in [AD5].			

4.2 PERFORMANCE, AVAILABILITY AND SIZING REQUIREMENTS

PPF-0200	CGS-SYS-4.2.2-0060 CGS-SYS-4.2.2-0070	PERF	TEST
For a PPF in static allocation mode, it shall perform the generation of all its higher level products on a single CPU Power3 at 375Mhz and with up to 2 GB of RAM in a bounded time. For a PPF in dynamic allocation mode, it shall perform the generation of all its higher level products on a single CPU Power3 at 375Mhz and with up to 512 MB of RAM in a bounded time. Timing starts from when PPF has received the necessary data for processing. Timing stops when PPF has made available all the products required and has signalled the end of the processing.			

PPF-0210	CGS-SYS-4.2.2-0060	PERF	TEST
The perfomance ratio (processing time/sensing duration) shall be maintained in case of smaller input data: i.e. the processing time of a granule representing e.g. 1/30 of an orbit shall be achieved within 1/30 of the time allocated for the full orbit.			
PPF-0218		PERF	ANALYSIS
The processing time shall be bounded: In the worst case of repeated abnormal inputs leading to extra non-nominal processing, the processing time shall not be more than E * nominal_for_the_same_sensing_duration. coef E to be defined between 1.2 and 2.			
PPF-0220	CGS-SYS-4.2.2-0080	PERF	ANALYSIS
The product quality degradation introduced by the PPF in implementing the product generation function shall be less than 1% of the target overall product accuracy .			
PPF-0230	CGS-SYS-4.2.2-0080	PERF	ANALYSIS
The computation shall be written in a way that yields the better accuracy following the state of the art of scientific computation.			
PPF-0250		PERF	ANALYSIS
The source code shall be optimised regarding the execution time.			
PPF-0271		PERF	TEST
The instrument mode at sensing time shall be determined from the PPF input data flow : - L0 or L1a for ASCAT, GOME, GRAS and ATOVS L1 - L0 or L1a or L1b for IASI L1 - L1b for ATOVS L2 - L1c for IASI L2			

4.3 INTERFACES REQUIREMENTS

The CGS design is built of facility installed and configured according to system level data. These data covered :

- Directory entries stored and managed as part of the NAMS common services of DIF,
- CGS TM definition, and content,
- Facility directory tree,
- Logical file names,
- Mandatory facility function required to support the CGS role policy.

Besides these facility centric definitions, other CGS registration data mechanism applies to users and users roles.

4.3.1 Logged information Registration

The logged information is collected by a service of the Local agent. The facility shall provide the following information:

- An identification string
- Textual information (ready for display purpose)

Logged information will be splitted between :

- Events forwarded to MCS through the LogEvent API,
- Trace information (useful for investigation, but not to be displayed during operation) stored locally by the facility through the LogTrace API.

4.3.1.1 Events forwarded to MCS

4.3.1.1.1 Event Identification string format

As facilities are able to report errors by channels like report, inter-facility co-ordination, etc, the use of this electronic means leads to the definition of predefined errors registered at facility level. In order to keep a system level consistency and unambiguity, each event forwarded to MCS is identified by a specific string built according to the rules described in this section.

The identification string contains the following fields (12 characters):

Table 4.3-1 Event identification string

Field	Number of characters	Possible values	Comments
Facility ID	2	Refer to Table 4.3-2	TBC : could be added by the API
Location ID	2	A1 for CDA-1 A2 for CDA-2 GS for GS-1 or GS-2	
Severity ID	1	A for Alert W for Warning I for Information	Severy ID is hard coded in the facility errors registration and is not dynamically defined.
Type	1	Refer to Table 4.3-3	
Service	3	Facility dependent	
Additional identification field	3	Facility dependent	

Service and Additional identification fields are allocated by the facility developer. It gives the flexibility to allocate identification string within a facility per sub domain as needed.

Table 4.3-2 Facility ID

Reserved	XX
DIF	DI
FDF	FD
FEP	FE
MCS	MC
MPF	MP
PGF	PG
PSF	PS
QCF	QC
PPF_ASCAT	AS
PPF_ATOVS_L1	A1
PPF_ATOVS_L2	A2
PPF_GOME	GO
PPF_GRAS	GR
PPF_IASI_L1	I1
PPF_IASI_L2	I2
Communication services	CO

Table 4.3-3 Type identification

Type	Id
Products	P
Auxiliary data	A
Schedule	P
Software	S
Resource	R
Network	N
Media	M
Payload	I
Platform	V
Others	

4.3.1.1.2 Logged textual information

The textual information will be a string of up to 80 characters (TBC).

Each message produced by a facility and able to be displayed to the operator is documented per facility.

One event identification string is associated to one logged information (1 to 1 relation). This act as the primary key to index the logged information directory of the facility

For each item of the logged information registery, the following information is documented (refer to §4.3.6.1)

- Displayed string format (including place holder for variable fields). The string will be fully formatted prior the call to the LogEvent service of the MLA [1],
- Item textual description,
- Recommended action when the event occurred.

4.3.1.2 Trace information

4.3.1.2.1 Trace Identification string format

As in 4.3.1.1.1 an identification string will be used for each debug information and will be set according to the following rules.

Table 4.3-4 Trace identification string

Field	Number of characters	Possible values	Comments
Facility ID	2	Refer to Table 4.3-2	TBC : could be added by the API
Location ID	2	A1 for CDA-1 A2 for CDA-2 GS for GS-1 or GS-2 RU for RUS	
Severity ID	1	A for Alert W for Warning I for Information D for debug	Severy ID is hard coded in the facility errors registration and is not dynamically defined.
Type	1	Refer to Table 4.3-3	
Service	3	Facility dependent	When relevant the service field will be used to identify the related satellite (M01,M02,M03,N16,N17)
Additional identification field	3		

4.3.1.2.2 Trace textual information

The textual information will be a string of up to 80 characters (TBC).

[1] (%s or equivalent format will be used in the flatfile)

The textual information shall contain the 8 first characters of the "source file name" and the "line number".

Each message produced by a facility and able to be displayed to the operator is documented per facility.

For each item of the logged information registery, the following information is documented :

- Logged string format (including place holder for variable fields). The string will be fully formatted prior the call to the LogTrace service of the MLA [2],
- Item textual description,
- Recommended action when occurred.

4.3.2 NAMS

N/A (TBC) for the PPFs.

4.3.3 CGS HK-TM

Every Facility HK-TM inherit of the following general rules, unless otherwise stated in the relevant « per facility section ». So, when no specific behaviour is described, the general rules shall apply.

4.3.3.1 Facility HK-TM content

The Facility HK-TM shall be sent to the MCS at least every minute (configurable by HKTM category), and asynchronously in case of failure, using a Local Agent, across an Ethernet link.

The Local Agent monitors the facility MP for the MCS. The facility MPs use the SendFunctionStatus API of the Local Agent to send status messages. The status messages hold a entity validity flag and a resource value as computed by the facility. The structure of the Facility HK-TM (refer to [R-I-UM011]) is :

- a status Identification string
- a status validity flag, from :
  - UNKNOWN : value not relevant, not available ;

[2] (%s or equivalent format will be used in the flatfile)

- FULL : value relevant ;

▼ A value encoded in a string according to MLA SUM description. A value being

- a string ready for display purpose,
- a integer,
- a float,
- a boolean,
- a date.

Note that the status validity flag does not reflect the content of the facility HK-TM, but only the relevance of this information.

The set of its Facility HK TM is split into five main categories :

1. Operational readiness, which describes the state of each main task within the processing chain and their mapping with the facility nodes, sent within 5 seconds after occurrence of the event origin of the change.

2. Interface readiness, which describes the state of each interface with external facilities sent within 5 seconds after occurrence of the event origin of the change.

3. Resource capacity, including the monitoring of the CPU, RAM consumed by the facility, and the CPU, RAM DISK available for each node sent every minutes (configurable) or within 5 seconds after occurrence of the event origin of level change.

4. Performance metrics, giving a snapshot of each node working at a time sent every minutes(configurable), always FULL when relevant.

5. For each satellite dump, a pipeline status, to provide a progress status on the basis of the processed granules sent within 5 seconds after occurrence of the event origin of the update.

4.3.3.2 CGS HK-TM status Identification string building rules

The identification string contains the following fields (7 characters):

Table 4.3-5 HKTM string id.

Field	Number of characters	Possible values	Comments
Facility ID	2	Refer to Table 4.3-2	
Category	1	OPERATIONAL READINESS : O INTERFACE READINESS : I RESOURCE CAPACITY : R PERFORMANCE METRIC: P PIPELINE STATUS : S	
Location ID	2	A1 for CDA-1 A2 for CDA-2 GS for GS-1 or GS-2	
Type	1	Refer to Table 4.3-3	
Service	3	Facility dependent	When relevant the service field will be used to identify the related satellite (M01,M02,M03,N16,N17)
Additional identification field	3	Facility dependent	

4.3.3.3 Operational readiness status rules

Operational readiness (O) statuses describe the state of main functions allocated to the facility. Each facility shall at least define one O status. The O status has a string value holding the description of current processing stage.

An operational readiness status will be sent to MCS :

- Each time main function has performed one of the processing stages (including its initialisation) successfully with the complementary information, describing the processing stage.
- In case of a pre-defined fatal error (disk failure, unrecoverable arithmetic fault,...) within the processing, with the complementary information, describing the error which occurred.

Additional information on the function can be brought using Resource Capacity or Performance metrics statuses

4.3.3.4 Interface readiness rules

The identification string will identify the related interface.

The attached value of an Interface readiness HK-TM is ON or OFF (the communication is down).

At the beginning the validity flag is UNKNOWN, and is updated after the first connection attempt). The status remains ON unless an error condition is encountered, and OFF until connection restore, with the validity flag set to FULL

Additional information on the interface can be brought using Resource Capacity or Performance metrics statuses

4.3.3.5 Resource capacity rules

Resource Capacity (R) status are composed of a numeric value carrying the quantity of a resource available to the facility. These status are defined along with sets of thresholds used to derived a severity, within the MCS database.

Note that the ressource capacity sent by the PPFs to MCS shall include one TM related to the host-id.

4.3.3.6 Performance metrics rules

The software version shall be returned as a status (string format) of type « Performance Metric » with additional identification string set to VER. It is sent only at the facility start-up.

Other Performance Metric (P) statuses shall be sent every minute (configurable). These statuses carry information on which its is not planed to applies limits checking.

4.3.3.7 Pipeline status rules

The pipeline status concern : FEP, PGF, DIF and PPFs.

The set of pipeline statuses for each facility shall be time-stamped with the same current UTC time to allow to link the information together.

The pipeline statuses shall be sent to the MCS :

- at the completion of a PDU/granule production/transmission (even in case of failure),
- periodically for the PPFs (every minute (configurable)).

At least the following statuses per product will be forwarded to MCS by FEP, PGF, and DIF:

- One will contain the UTC sensing stop time of the generated/transmitted PDU/granule,
- One will contain the UTC sensing start time of the generated/transmitted PDU/granule,
- One will contain the "start orbit number" of the dump to which the related PDU/granule belongs,
- One will contain quality information (e.g. % of available MDRs in the generated PDU),
- One will contain a stage information with at least the values "ok" and "failed" (for instance, failed will be set for a PDU which has been dropped by DIF following a timelines check).
- At least the following statuses per product will be forwarded to MCS by the PPFs (TBC):
- One will contain the UTC sensing time currently handled by the PPF,
- One will contain the UTC sensing start time of the PDU to be generated,
- One will contain the "start orbit number" of the dump to which the related PDU belongs,
- One will contain quality information (e.g. % of available MDRs in the generated PDU),
- One will contain a stage information with at least the values "ok" and "failed" (for instance, failed will be set for pipeline statues forwarded by the PPF when the PPF has been commanded to stop by the PGF due to an out of timeline production).

In case no change has occured since the previous transmission, as the other statuses, the pipeline statuses shall be sent to the MCS with the Validity_flag set to "false".

For the PPFs, in addition to the above status, ALL the other statuses will be handled accoding to the pipeline statuses rule, i.e. all the statuses shall be synchronised on the pipeline statuses and then the set of statuses defined for a PPF shall be sent with the same UTC current time stamp :

- Interface readiness with the PGF,
- Operational readiness,
- Ressource capacity.

4.3.3.8 CGS HK-TM data collection

For each of the Facility HK-TM the following information is documented :

- ▼ Numeric value Unit,
- ▼ Numeric value Scale,
- ▼ List of possible displayed string format,
- ▼ Item textual description,
- ▼ Recommended action when occurred,
- ▼ Limits, when relevant.

4.3.4 Directory tree

4.3.4.1 Delivery tree

The delivery directory structure shall reflect the architectural design of the Facility software.

The following types of file shall be contained in separate directories :

- ▼ source code,
- ▼ configuration files,
- ▼ data,
- ▼ operational scripts,
- ▼ build scripts,
- ▼ test tool source files,
- ▼ test data.

Note that the directory used for the configuration files shall include only the parameters specific to the facility. Some configuration parameters will be identified as "system parameters" and will be put in other files in the test data directory.

4.3.4.2 Installation unit tree

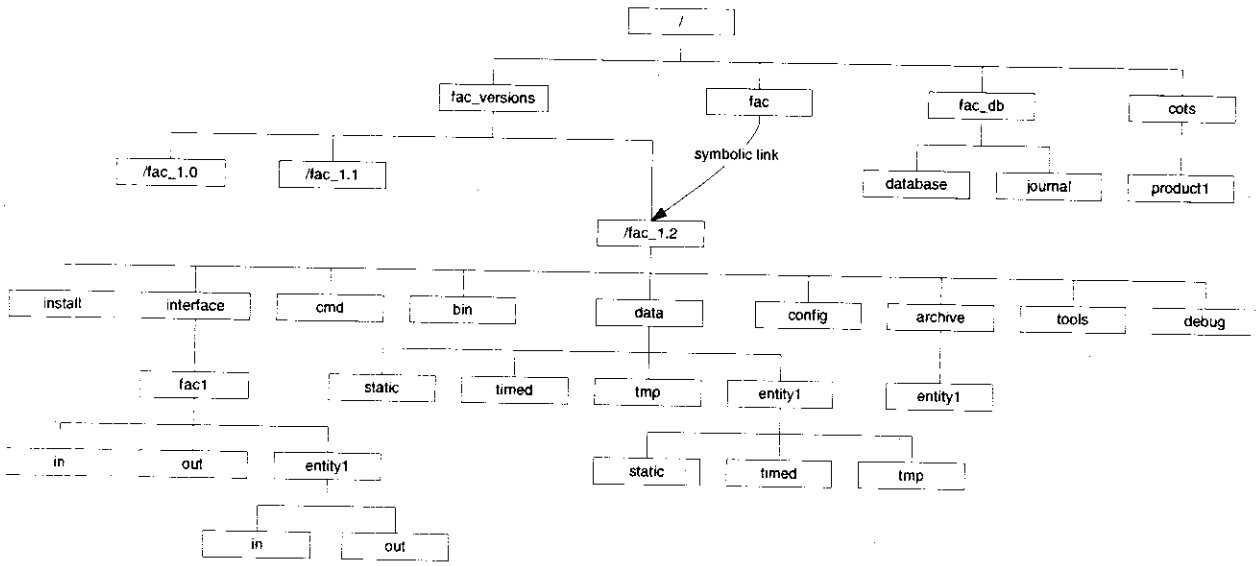
No particular requirements.

4.3.4.3 Run time tree

The Facility run-time directory structure shall comply with the structure illustrated in Figure 4.3-1 Facility Directory Structure (TBC), and described by the requirements below. Only directories actually used by the application shall be created.

There shall be no constraints on how this structure is mapped onto physical and logical disk structures. A subdirectory in this structure may be in the same file system as its parent, be the mounting point for a file system, or be a symbolic link to a directory in another file system.

Figure 4.3-1 Facility Directory Structure



The Facility data space shall be accessed via two directories : /fac and /fac_db. A third directory, /fac_version shall hold successive versions of the facility application. A fourth directory /cots is where COTS products are installed, each product in its own subdirectory.

The /fac directory shall be a symbolic link to a subdirectory of /fac_version. The directory pointed to by /fac is termed the current version.

The /fac_db directory shall have two sub-directories : database for the Facility's Oracle database files and journal for the Facility's Oracle journal files. These subdirectories shall be in distinct file systems and shall reside entirely on separate physical disks.

The subdirectories of /fac_version shall be named fac_*version-number* and shall have the subdirectories listed in Table 4.3-6 Facility Version Subdirectories. Note that in the following tables, the term "entity" refer to metop1, metop2, metop3, noaan, noaanp, sim1, sim2, x_suitcase, or s_suitcase.

Table 4.3-6 Facility Version Subdirectories.

Subdirectory	Data type	Description
install	Installation files	Receives the contents of the Facility installation media. Following installation of the version, it shall be possible to delete this directory with no effect on the Facility.
bin	Binary executables	
cmd	Script executables	
/interface/fac1/in	Entity-independent data received from Facility fac1	Entity-independent data which comes from Facility fac1 according to the applicable ICD.
/interface/fac1/out	Entity-independent data sent to Facility fac1	Entity-independent data which are destinated to Facility fac1 according to the applicable ICD.
/interface/fac1/entity1	Entity-dependent data from/to Facility fac1	Facility fac1 writes/reads data concerning entity1 in this directory, according to the applicable ICD. Subdirectories are as for /interface/fac1.
data	Entity-independent data	Entity-independent data not assigned to one of the subdirectories.
data/static	Entity-independent static data	Entity-independent data which are not time dependent (no validity period).
data/timed	Entity-independent time-dependant data	Entity-independent data which is time-stamped and evolves over time.
data/tmp	Entity-independent temporary data	Entity-independent data which has a limited lifetime such as working files. This data typically becomes obsolete if the Facility is restarted.
data/entity1	Entity-dependent data	Data concerning entity1 not assigned to one of the subdirectories. Subdirectories are as for /data.
config	Configuration data	Static data which parameterises the Facility functionality.
archive	Entity-independent data archive	Entity-independent on-line archive.
archive/entity1	Data archive for entity1	On-line archive concerning entity1.
tools	Test and maintenance tools	Tools and associated data which are not part of the operational Facility. Subdirectories are Facility-specific.
debug	Investigation tools and data	Tools for detailed investigation of Facility anomalies. Debug data files.

4.3.5 File name conventions

4.3.5.1 EPS Products

EPS products shall be named using the following naming convention, which provides a product name that uniquely identifies any product and provides a summary of its contents.

The name will be composed of a number of product name fields separated by the underscore character("_").

The product naming conventions is defined in [E-AD-98] and is composed of the following fields :

<INST>_<PROD_TYPE>_<PROC_LEVEL>_<SPACECRAFT>_<SENSING_START>_<SENSING_END>_
<PROC_CENTRE>_<PROC_MAJOR_VER>_<PROC_MINOR_VER>_<FORMAT_MAJOR_VER>_
<FORMAT_MINOR_VER>_<PROC_TIME>_<PROC_MODE>_<SUBSET_FLAG>_<SUBSET_TIME>

The fields may contain only upper case letters [A...Z], lower case 'x' (to represent a blank item), and numbers [0...9].

As the size of each product name field is fixed, the product name is always 108 (TBC) characters in length.

The format for <SENSING_START>, <SENSING_END>, <PROC_TIME>, <SUBSET_TIME> is :

YYYYMMDDHHmmssZ.

Each product name field is directly related to the field of the same name within Main Product Header Record (MPHR) (refer to [E-AD-36]). As far as the MPH will be ASCII encoded, no conversion is needed (TBC).

Table 4.3–1 Product name fields

Product Name Field / MPHR Field	Description	Size in characters	Possible values
INST	Instrument identification	4	[E-AD-36] generic PFS
PROD_TYPE	Product Type	3	[E-AD-36] + GTS
PROC_LEVEL	Processing Level identification	2	[E-AD-36]
SPACECRAFT	Spacecraft identification	3	[E-AD-36]
SENSING_START	UTC Time of start of Sensing Data	15	NA
SENSING_END	UTC Time of end of Sensing Data	15	NA
PROC_CENTRE	Processing centre Identification	4	[E-AD-36]
PROC_MAJOR_VER	Processing chain major version number	5	NA

PROC_MINOR_VER	Processing chain minor version number	5	NA
FORMAT_MAJOR_VER	Product Format Major Version Number	5	NA
FORMAT_MINOR_VER	Product Format Minor Version Number	5	NA
PROC_TIME	UTC time at start of processing for the product	15	NA
PROC_MODE	Identification of the mode of processing	1	[E-AD-36]
SUBSET_FLAG	Identified is a product has been subsetted	1	[E-AD-36]
SUBSET_TIME	Time that the product was subsetted	15	[E-AD-36]

The field PROD_TYPE shall be set to GTS by the PPFs for the products following the BUFR format for the GTS network.

4.3.5.2 Auxiliary data and other files naming conventions

The following convention apply to the files :

- provided by an external source through the GFT (used as auxiliary data for the production, or for quality analysis...)
- provided to an external destination through the GFT (disseminated auxiliary data to UMARF, to CAL/VAL entity, source code sent to UMARF...),
- files which remain internal to the CGS, except for the ones identified inTable 4.3–4 Filename convention for PGF/PPF interfaces.

The file naming convention for the above identified files will be derived from the following convention:

<INST>_<FILE_CAT>_<PROC_LEVEL>_<SPACECRAFT>_<VALIDITY_START_TIME>_<VALIDITY_STOP_TIME>_**<CREATION_DATE>_<SOURCE>_<FILE_TYPE>**

When relevant one value of the following list shall be used for FILE_CAT field :

- OSE : to identify that the file is related to an OSE request,
- OSR : to identify that the file is related to an OSE report,
- OBS : to identify that the file is related to an OBSWM request,
- OBR : to identify that the file is related to an OBSWM retrieve,
- FDP : to identify that the file is related to an FD Products,
- RPT : to identify that the file is related to a report,
- CON : to identify that the file is related to a configuration file,
- OGS : to identify that the file is related to a on-ground software update,
- OGP : to identify that the file is related to an on-ground parameter table update,
- IOR : to identify that the file is related to an Instrument operation request,
- CTX : to identify a context file.

Moreover, in the specific cases of a context file:

- the field VALIDITY_START_TIME shall be set to the end (sensing time) of the generated product.
- the field VALIDITY_STOP_TIME shall be set to 'xxxxxxxxxxxxx' .
- the field FILE_TYPE shall include the stop orbit number found in field ORBIT_START of the unprocessed MPHR as detailled in tableTable 4.3–3 File type field.

Table 4.3–2 File naming conventions - fields

Auxiliary data Name Field	Description	Size in characters	Possible values
INST	Instrument identification	4	[E-AD-36] +ADCS or xxxx if not relevant
FILE_CAT	File category identification	3	[E-AD-36] + OSE : to identify that the file is related to an OSE request + OSR : to identify that the file is related to an OSE report OBS : to identify that the file is related to an OBSWM request+ OBR : to identify that the file is related to an OBSWM retrieve FDP : to identify that the file is related to an FD Products + RPT : to identify that the file is related to a report + CON : to identify that the file is related to a configuration file OGS : to identify that the file is related to a on-ground software update OGP : to identify that the file is related to an on-ground parameter table update IOR : to identify that the file is related to an Intsrument operation request TBD facility
PROC_LEVEL	Processing Level identification	2	[E-AD-36] or xx if not relevant
SPACECRAFT	Spacecraft identification	3	[E-AD-36]
VALIDITY_START	UTC Sensing Time start of validity	15	NA
VALIDITY_END	UTC Sensing Time stop of validity	15	NA Can be also 15'x' if stop time is not relevant or not known.
CREATION_DATE	Creation time of the file	15	NA
SOURCE	Identification of the file provider	4	IAST, CALV, NOAA, ECMW
FILE_TYPE	Identification of the data type	10	Refer to the Table 4.3–3 + TBD facility

A preliminary list of the possible values for FILE_TYPE is provided hereafter (to be completed at least to match the needs identified in the Auxiliary Data Inventory) :

Table 4.3–3 File type field

Filetype id	Description
REPORTSCGS	Report (e.g. A-DCS Activity Report)
OPERATPLAN	Operation Plan from CGS (e.g. for A-DCS)
OPERATRESU	Ops Result (e.g. for A-DCS)
CALIVAREPO	Calibration/Validation Report
GNSSEPHEME	GNSS Ephemerides
GRASTRACKI	GNSS GRAS Tracking Data
GRASGROUND	GNSS Ground Data
INSTRUHKTM	Instrument HKTM (e.g. IASI, A-DCS...)
OBPARAUPDA	On-Board Parameter Table Update
OBSOFTUPDA	On-Board Software Update
OGSOFTUPDA	On-Ground Software Update
KEY_SATMAN	Metop Encryption Keys for SatMan
KEY_CKMCME	Metop Encryption Keys from C-KMC
METOP_HKTM	Metop HKTM for IASI TEC, OSE...
METOOBTUTC	Metop OBT/UTC for NOAA
METOORBPAR	Metop Orbit Par. from CLS
METOPOSV	Metop OSV for CLS, GRAS, NOAA...
METOPTCOSE	Metop TC history to OSE
MHSTELEMET	MHS Telemetry
NOAAOBTUTC	NOAA OBT/UTC to QCF
NOAAOSVCGS	NOAA OSV for FD and QCF
REQMETOPTM	Request from OSE for Metop HKTM
REQMETOPTC	Request from OSE for Metop TC
RUSENCRKEY	RUS Encryption Keys
CALIPARAPR	Calibration parameters
CALIPARASE	Secondary calibration parameters
COEFLANSEA	Coefficient for land and sea surface temperature det.
NAVIGPARAM	Navigation parameters
LANDMARKDA	Landmark database
SCENEBRIGH	Threshold for brightness temperature difference tests in the scene analysis
SCENECOEFF	Coefficient for threshold determination (scene analysis)
COEFFISRFS	Normalised window response functions and central wavenumbers

AVHRRSCENE	Ouput of AVHRR scene analysis
xxxxxZZZZZ e.g. xxxxx00177	Context file of each of the various instruments: 'xxxxx' might be changed in case several type of context should be differentiated. ZZZZZ shall be set to the start orbit number, copied from the field ORBIT_START of the MPHR of the unprocessed data. It is formatted on exactly 5 characters and left padded with zeros.
...	

The following files will follow a specific naming convention:

Table 4.3–4 Filename convention for PGF/PPF interfaces

identification	file name convention
PPF_*XXXX*_Work_Order	xxxxxy_**wo**_zzz where: ⇒ xxxxx = MP id for PPF S/W where xxxxx is amongst the following : IASI1 IASI2 ATOVS GOME_ GRAS_ ASCAT ⇒ y = required product level number (1 or 2) ⇒ zzz = counter
PPF_*XXXX*_reports	xxxxxy_**wo**_zzz_uuu.**rpt** where: ⇒ xxxxx = MP id for PPF S/W (the one from the corresponding work order) ⇒ y = product level number (the one from the corresponding work order) ⇒ zzz = counter (the one from the corresponding work order) ⇒ uuu = 001 = counter for the intermediate reports

4.3.6 Facility data registration presentation

Each facility data registration will be documented according to the following template.

4.3.6.1 Facility Logged Event Registration

<<String ID>>	<<Facility>>	<<Severity>>	<<Type>>	<<Service>>	<<Additional identification>>
<<Displayed String>>					
<<Item Description>>					
<<Recommendation>>					

DIAPIN001	DIF	ALERT	PRODUCT	INGEST	001
Time out in product %s ingestion					
The DIF ingest task was not able to collect an expected product within its expected time frame					
Check the PGF output server health					

4.3.6.2 Facility NAMS

Table 4.3-7 <<MAP>> Keys around the facility

Location	Ressource	Qual.	Description
<<location>>	<<Ressource>>	<<Qual.>>	<<Description>>

Table 4.3-8 CGS services Keys around the facility

Location	Ressource	Qual.	Description
	DisseminatedProduct	NOAA_NRT	Service source point for operationnal product from NOAA S/C to NRT users

4.3.6.3 Facility CGS HK-TM

<<String ID>>	<<Facility>>	<<Category>>	<<Type>>	<<Additional identification>>
<<Displayed String>>				
<<Item Description>>				
<<Recommendation>>				
<<limit definition>>				

PGOP001	PGF	Operational	Product	001
Idle GAC ingestion from FEP for dump sensing start %s sensing stop %s completed ...				
PGF Input data server readiness				
TBD				
N/A				

5. FUNCTIONAL REQUIREMENTS

5.1 GENERIC PPF FUNCTIONAL REQUIREMENTS

PPF-0300	CGS-MCF-5.2.5.5-0050	FUNCT	TEST
The PPF shall extract from the ancillary data all the information necessary for the monitoring of Metop payload from the Instrument source packet, and made them available to the MCS in specific status messages or via a specific product.			
PPF-0310	CGS-MCF-5.2.5.6-0020	DES	ANALYSIS
The complete status and performance of the PPF shall be obtainable from status and product quality summary without need for reference to command history or event log.			
PPF-0340	CGS-DPG-5.3.3.1-0010 CGS-DPG-5.3.5-0040	FUNCT	TEST
The PPF generates products from input data in accordance with the EPS generic Product Format [AD3].			
PPF-0350	CGS-DPG-5.3.4-0080 CGS-DPG-5.3.5-0130	FUNCT	TEST
The PPF shall provide information to the PGF. As a minimum, the following events and parameters need to be reported on a per execution basis: 1) start of processing (including the identification of input data); 2) stop of processing 3) end of processing including processing report (e.g. Quality, validity, completeness of generated product, summary of problems encountered, output data identification, etc.); 4) significant processing events. 5) PPF resources statuses			
PPF-0370	CGS-DPG-5.3.5-0110	DES	INSPECTION
All information used by the PPF which can be expected to be modified (e.g. parameters and thresholds for on-line QC, physical constants etc.) shall be configurable, i.e. the information is not hard coded in the PPF.			
PPF-0390	CGS-MCF-5.2.5.6-0060	DES	ANALYSIS
Each status TM shall be sent via a CFI routine as follows: The identity of the PPF is given in unambiguous way. The information hold a timestamp using the sensing time (TBC)			

PPF-0400	CGS-DPG-5.3.5-0140	FUNCT	ANALYSIS
At least the following situations are considered significant processing events and shall be reported: 1) appended information can not be produced in its entirety; 2) appended information is corrupted or invalid; 3) inability to produce the corresponding level product; 4) absence of required input data; 5) operation in a non-nominal mode; 6) any non-nominal processing behaviour.			

PPF-0410	CGS-DPG-5.3.5-0150	FUNCT	ANALYSIS
All failures detected and recovery actions taken shall be unambiguously reported using the PGF services provided for this purpose. However when the same failure occurs for many records, only one message summarising the failures shall be logged.			

PPF-0420	CGS-DPG-5.3.5-0180 CGS-DPG-5.3.5-0290	FUNCT	TEST
The PPF shall support nominal mode and investigation modes, as described in section 3.			

PPF-0430	CGS-DPG-5.3.5-0190	FUNCT	TEST
In case of unrecoverable processing states, the PPF shall provide a mechanism to enable it to abort gracefully within the PGF processing framework, in particular graceful abortion implies to respect the normal protocol of exchanges with the PGF.			

PPF-0440	CGS-QCS-5.4-0010	DES	TEST
During the processing the PPF shall perform quality monitoring used to populate: 1) the PCD parameters included in the products 2) the statuses and processing events sent to PGF and MCS Note: a PCD (Product Confidence Data) is a number intended to qualify the validity of a measurement or a computation or even a summary standing for a certain sensing duration.			

PPF-0460	CGS-DPG-5.3.5-0200	FUNCT	INSPECTION
The PPF shall detect and handle problems that occur during product processing.			

PPF-0470	CGS-DPG-5.3.5-0200	FUNCT	INSPECTION
The following arithmetic errors : division by zero, log of negative value, square root of negative values, overflows/underflow, ... must not lead to a crash of the PPF, their impact shall be limited to the minimum. Their occurence shall appear in the product Confidence Data.			

PPF-0480	CGS-DPG-5.3.5-0200	FUNCT	TEST
The processing PCD attached to the product shall indicate occurrence of arithmetic errors as well as the faulty application software module. This information must be detailed within the various PCD structures (error code , error parameters specifying the nature of the error), within the MPH.			

PPF-0490	CGS-DPG-5.3.5-0200	FUNCT	TEST

The general error handling strategy in the product is:

Blank MDRs may be generated : same structure as normal MDR with the MDR PCD set to -1 within the MDS. The only valid data contained in the blank MDR are the UTC time, localisation, and the instrument mode identification. All other fields of the blank MDR must be set to default values.

When Blank MDRs are generated, corresponding PCD & flags shall set accordingly in the product header.

At processing time, if an instrument source packet is valid but not error-free, then calibration values contained in the auxiliary data of the packet must be ignored and the previous values, obtained from the last error-free source packet must be used. However, when the product format allows to manage a gap in the record sequences, and when a whole record would be blank, gaps will be created in the output products.

When such gaps are allowed by the product format of the input data, these gaps must be dealt with properly.

PPF-0500	CGS-DPG-5.3.5-0220	FUNCT	TEST

PPF shall perform on-line quality control to establish the internal consistency of the generated product, to monitor the completeness, the availability and the quality of all necessary input data, and set pre-defined quality flags (Product Confidence Data, PCD) as defined in the ATOVS product generation specification documents [AD11, AD13, AD15, AD17] and [AD19]

PPF-0510	CGS-DPG-5.3.5-0230 CGS-QCS-5.4.2-0010	FUNCT	TEST

On-line quality control shall be performed automatically and on a continuous basis for all products generated by the PPF.

PPF-0520	CGS-DPG-5.3.5-0280 CGS-DPG-5.3.5-0050	DES	INSPECTION

The PPF shall follow a number of rules as specified in a PPF coding and implementation rules document [AD8].

PPF-0530	CGS-SYS-9.3-0070	TEST	TEST

The PPF shall be implemented in a way to support testing scenarios covering:
1) processing data volume shorter than one dump;
2) processing data volume longer than one dump (a dump is approximately 1 orbit long).

PPF-0550	CGS-MCF-5.2.5.6-0050	DES	TEST

The processing status shall be sent at a minimum rate of once per 15 seconds. The sending of a status must correspond to an actual progress of the computation, it mentions quality information summarizing the data processed since the previous status.

PPF-0554	CGS-DPG-5.3.5-0300	DES	ANALYSIS

The PPF shall be implemented such that a new PPF can be installed following routine maintenance procedures.

PPF-0580	CGS-DPG-5.3.1-0040	FUNCT	ANALYSIS
In case of unrecoverable processing states, the PPF shall provide mechanisms to localise the impact of these processing states and to confine it to the affected process and its child processes.			

PPF-0610	CGS-DPG-5.3.5-0170	DES	ANALYSIS
The PPF shall 1) be robust against duplication of data between data overlap, 2) limit the propagation of truncation errors, 3) be robust against any incompatibilities (e.g. different PPF version, product format version, aux. data version etc.).			

PPF-0630	CGS-DPG-5.3.5-0270	FUNCT	INSPECTION
In cases where PPF implements online parameter estimation (e.g. misregistration estimation) this shall satisfy the following constraints: 1) restart a new estimation without considering historical information if, e.g. manoeuvres have occurred, the instrument mode has changed; 2) the estimation function needs to work in all identified processing modes; 3) the online function can resume from any appropriate (e.g. older) 'warmstart' information.			

PPF-0640		FUNCT	TEST
The PPF generates one PDU per STEP command, the time span of each PDU follows a rule specific to each PPF in order to respect the natural boundaries of the products.			

PPF-0641		FUNCT	TEST
The PPF with L0 data as input shall process only entire scan lines (when scan line is relevant for the related instrument).			

PPF-0642		FUNCT	TEST
The PPF shall be requested to produce a HL product on a sensing window [Tstart_req_HL, Tstop_req_HL] with an input covering [Tstart_input,Tstop_input] (with Tstart_req_HL ≥ Tstart_input and Tstop_req_HL ≤ Tstop_input) The output product shall be as close as possible from the requested window. The output product shall in particular fulfill the following rules : Tstart_output ≤ Tstart_req_HL Tstop_output ≤ Tstop_req_HL			

PPF-0645		FUNCT	TEST
Deleted			

PPF-0670	CGS-ONC-5.5-0020	FUNCT	TEST
The products obtained from 2 runs on consecutive time intervals shall contain the same values than the product obtained in a single run. Formally: if product P is the result of a run on interval [t1, t2], Q the result of a run on interval [t2, t3], and R the result of a run on interval [t1, t3], it shall be ensured that R = P U Q . Note that the input supplied for processing an interval [t, t'] will often encompass a larger window. Practically this guaranties the continuity across dump boundaries.			

PPF-0680		FUNCT	TEST
Deletec			
PPF-0681		FUNCT	TEST
The result of a production on interval [t1, t2] aborted by a BREAK command shall be the same than the result of a run on interval [t1, t] for a certain time t specified in return in the report.			
PPF-0690		FUNCT	TEST
Stop terminates prematurely the processing. In this case the PPF shall respond in less than15s to a STOP order by a status ack and and shall terminate within 1 mn.			
PPF-0693		FUNCT	TEST
BREAK terminates prematurely the production. In this case the PPF shall respond in less than15s to a BREAK order by a status ack and and shall terminate within 1 mn.			
PPF-0691		FUNCT	TEST
Once a granule is generated and confirmed by the corresponding status sending, it shall no longer be modified.			
PPF-0692		FUNCT	TEST
On termination, the PPF shall clean-up its environement (except the intentional backup in case of abnormal processing) and free all resources.			
PPF-0700	CGS-DPG-5.3.5-0270 CGS-DPG-5.3.3.4-0130 CGS-DPG-5.3.3.4-0160	FUNCT	TEST
In the case the PPF needs a context file, it must ensure its backward compatibility (a file generated by version n of the PPF shall be understood of the version n+1). Besides, the context file shall include the version of the software which wrote it.			
PPF-0701		FUNCT	TEST
If the PGF fails to deliver a context file, the PPF shall use default values from a configuration file.			

5.2 OTHER FUNCTIONAL CONSTRAINTS

Defining the interface between the PGF and the PPF will be the first critical task of the project. The interface mechanisms (PGE) and the protocols have to be settled as early as possible to reduce development time and risk. Based upon experience with similar integration tasks we propose to define the interface as simple as possible according to the following principles:

- The PPF must be able to function as a stand-alone program.

  I.e. it must not depend upon any part of the PGF being present in order to run. Making the PPF independent makes both development of the PPF and the PPF simulators/test-stubs easier and less risky.

- The PPF shall be possible to start and control from the command shell.

All information the PPF needs to gather the required input and to generate the required output shall be described in the command line arguments.

Immediate commands shall be possible to issue using standard shell commands.

- While it runs the PPF shall give feedback (events and status) using the 'stdout' mechanism.
- Input files and report files shall be plain ascii wherever this is possible.

Having the work order and reports in human readable form enable us to verify and test the PPF interface without having to develop test-tools (which may be erroneous themselves).

- The PGE shall be based upon industry standards wherever possible.

Advantages:

- PPF development does not depend on PGF development,
- PPF can be tested independently from the PGF,
- PPF test stubs can be made with less effort,
- Reduce risk of errors introduced by any interface library,
- Reduce development time and risk of PGE API,
- Ease integration task, reduces risk of incompatibility at integration time.
- Industry standards are immediately recognized by developers, using them instead of project specific APIs makes the code more readable for programmers and hence easier to maintain. Also industry standards leaves no room for misunderstanding wrt syntax or semantics.

5.3 ANALYSIS OF PGE REQUIREMENTS

This section contains responses to PGE requirements, identified in the PPF to PGE document [AD2]. The analysis shows how PGE requirements are met through the strategy proposed for the PPF interface.

The interface shall operate in an operational environment, in which the product processing has to satisfy timeliness constraints and to cope with possible non-nominal system behaviour. The performance of PGE services shall be such that they enable the to fulfil the timeliness requirements as defined in [CGS-RD]

The PGE is implemented using industry recognized standards (e.g. POSIX) directly, whenever possible. Performance wise the best is to use these mechanisms directly, having an overlay will only contribute to more overhead since the overlay functions eventually have to use the standard mechanisms anyway.

The interface shall provide access to a number of PGE services. Three types of services are provided to the :

- -I/O services (IOS),
- -Monitoring and control services (MCS),
- -General support services (GSS).

The PGE will list acceptable API functions for all the services listed in its SUM.

The PGE shall interface with any PPF written as a PAM, usign the functionality provided by the PGE through its PGE services. It shall be possible to interface to written in different ANSI standard programming languages, as a minimum C,C++, FORTRAN

Since the PGE is based upon industry standards it will be available in the listed languages and portable.

The interface shall provide a mechanism through which the PPF can be commanded to start, stop, suspend, resume and abort. The interface shall provide a mechanism through which a processing mode enabling, disabling can be commanded in a PAM.

This interface is used for starting the PPF, sending commands to the PPF and for sending status from the PPF.

The start mechanism is a command line with argument vectors. This is easily achieved via standard language mechanisms (e.g. fork/exec in C). A description of legal arguments must be defined.

Direct commands can be realised using standard UNIX signals as for the abort of by message passing between the PGF and the PPF MP. The OS offers standard mechanisms for pausing, resuming and aborting a process (process group), the languages offers standard mechanisms for catching and handling other signals such as "stop".

The interface shall provide a mechanism by which the monitoring information generated by the PAM is accepted. The interface shall provide as a minimum notification of the following monitoring events : start of processing, stop of processing, suspend of processing, resume of processing, end of processing, significant events e.g. errors, warning, etc.

The most portable and straightforward mechanism for sending monitoring information is by using the ASCII text via stdout (formatted printing in C). A convention for contents must be defined. Stdout can easily be remapped on any I/O descriptor within the PGE environment prior PPF start.

The interface shall support a synchronisation capability to enable the PPF processes to synchronise and communicate between different processes.

The PPF may use any POSIX IPC mechanisms to communicate between internal processes and CGS communication services to communicate with the PGF.

The interface shall accept log entries created by the PPF

Logging can be done via events to the MCS local agent, part of the CGS common services (AD4).

The interface shall provide a PPF charecterisation mechanism that accepts/exchanges information to describe e.g. version, configuration, ressource requirements etc as provided by the

Communication between the MCS and the PPF are part of the CGS common services (AD4).

The interface shall provide a mechanism through which a PPF shall access support services.

Since the PPF is running in user-space the OS protects against uncontrolled use of system function. The standard OS functions should be available to the PPF.

The PPF shall use POSIX exception handling mechanisms.

All other service functions shall be available as a linkable library.

The interface shall provide access to a memory management function to allow the PPF to dynamically allocate/de-allocate memory.

The language's standard memory management run time functions shall be used.

The interface shall provide access to a Ressource Locator capability. This capability will allow to generate the use of I/O services for the PPF. It will support the use of wildcards to simplify the data access for frequently requested data attributes such as, current orbit, instrument, spacecraft, granule, etc.

This is the function of the NAMS directory of the CGS communications. services depicted in AD4.

The interface shall provide a mechanism to enable the PPF to create/delete temporary files and toread/write temporary files. All data exchange across this interface shall be performed using "ascii" data formats complying with existing standards (e.g. IEEE standard for number representation).

The language standard file handling functions shall be used. File formats shall be ASCII where possible.

Data "shared between PPSs" are handled as auxiliary data by the PGF

The interface shall provide a mechanism by which a PPF can access a configuration parameter file which stores information including and not limited to product naming and formating.

Configuration files shall be ASCII with a well defined format, and can be accessed via standard language features (e.g. fopen, fgets, sscanf in C).

The interface shall provide a mechanism through which PPF modules shall access static (e.g. World database, climatological data, instrument parameters, calibration data, thresholds for on-line Q/C, physical constants, etc.) auxillary data. The interface provide a mechanism by which a PAM can access all instrument characterisation parameters.

These data are part of the auxiliary data and are read from file using standard language file access features.

The interface shall provide a mechanism that supplies all dynamic input data required by a PPF including and not limited to : Science input data Products Dynamic auxillary data

The PGE services have in charge the auxiliary data dispacth to the PPF environement, the PPSs select the auxiliary data they requires according to a plocy defined in their processing requirements.

The interface shall provide a mechanism that accepts all dynamic data or product elements generated by the PPF . The interface shall also provide a mechanism that accepts all intermediate data from the processing that are generated e.g. under specific investigation mode.

Data generated by the PPF shall be exchanged via files.

THIS PAGE IS INTENTIONALLY LEFT BLANK

6. DESIGN REQUIREMENTS

6.1 GENERAL REQUIREMENTS

PPF-0710	CGS-SYS-4.2.1-0020	PERF	TEST
The PPF shall be designed so that is can be scaleable and portable for use on a upgradable target platform.			
PPF-0750	-	DES	ANALYSIS
The architecture of the PPF shall be kept as simple as possible (while satisfying all its functionnalities), in particular with regards to inter-processes operations.			
PPF-0760	-	DES	TEST
Memory errors (arrays indexed out of bounds, use of already freed memory, memory leaks) shall be checked systematically, either using a specific tool (e.g. Insure++), either by manual instrumentation of the code (e.g. use of macros to access arrays).			
PPF-0770	-	DES	TEST
The PPF shall provide a wrapper shell script used to start the PPF. The script set the whole PPF environment in a consistent way before launching the executable itself.			
PPF-0780	CGS-DPG-5.3.5-0030	DES	INSPECTION
The PPF shall be implemented as a Portable Application Module (PAM in Eumetsat terminology): The PPF is delivered as a set of source files, its associated make files, executables, configuration files, and along with installation procedures as defined in [AD8]. The make allows to build the operational version of the software with its "plugs" toward the PGF facility. Alternatively, it can also be built as a standalone executable, able to run without any external component – This executable takes no commands, however it takes its inputs from a directory tree as the operational version does. It has the same processing capabilities.			
PPF-0790	CGS-DPG-5.3.5-0060	DES	INSPECTION
The PPF implementation shall use a programming language from a PGF defined set of programming languages (i.e. ANSI C, C++, FORTRAN).			
PPF-0800	CGS-DPG-5.3.5-0070	DES	INSPECTION
The PPF shall be implemented using only ANSI standard compliant features of the language.			
PPF-0810	CGS-DPG-5.3.5-0080	DES	INSPECTION
If the PPF needs to be implemented using parallel programming this shall be done using PGF supported industry standards such as HPF, MPI, Open MP.			
PPF-0811		DES	ANALYSIS

It shall be possible to switch the floating point accuracy for all computations from double to long double (10 bytes) – e.g. a typedef double real; and no direct reference to double or float except for casting purposes (writing a product).

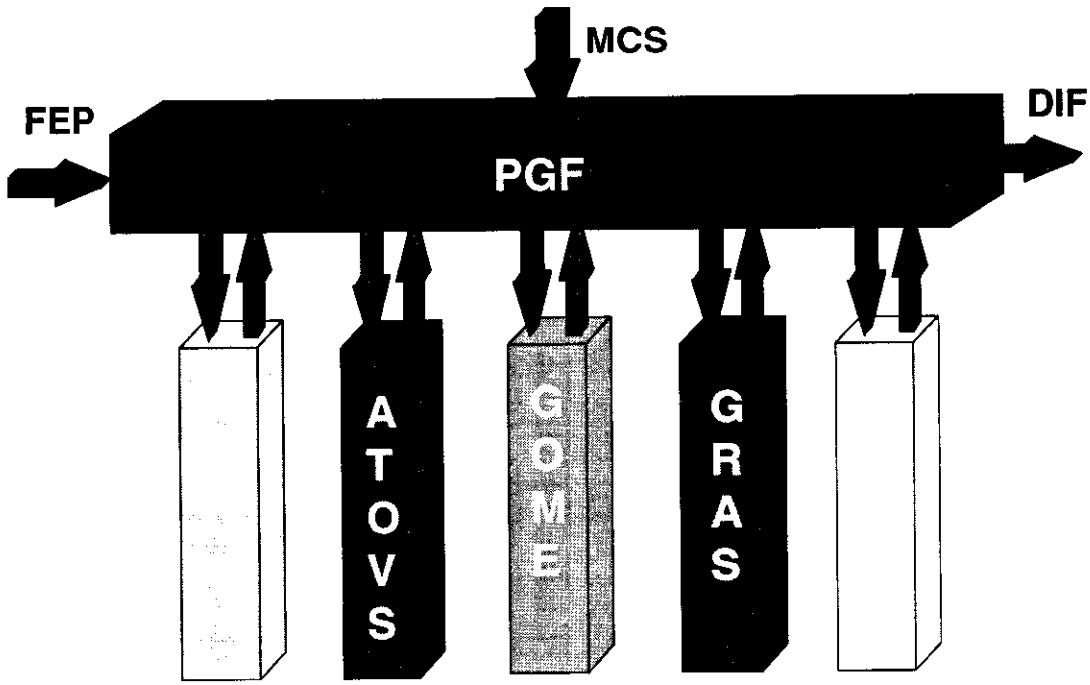
A test case shall be dedicated to running a nominal computation in long double accuracy and comparing the result to the double accuracy results.

6.2 PPFs MONITORING AND CONTROL

The Products Processing Facilities (PPF) are managed by the Product Generation Facility (PGF).

The PPFs are the algorithmic processing modules responsible for the generation of all products above level 0. They are used by the PGF as external resources. They are interfacing only with the PGF which supplies them with data and detailed commands, and receives back the complete products.

Figure 6.2–1 Data Processing and Product Generation Structure



6.2.1 PPFs standard architecture design

Each PPF works in standalone, with no direct interaction with any other PPF. The scope of the PGF host structure is to insure synchronisation between PPFs when the output from a PPF are needed as input by the others.

All the PPF instrument S/W (including IASI Level 1 CFI software) are managed on a similar way by the PGF.

1. Needed activation parameters and auxiliary data are first made available for the PPF on its file system.
2. They are then started by the PGF (Start command).
3. They receive the instrument data to be processed.
4. They receive a production request (Step command)
5. They generate higher level products on their own disks, at any time of the current production;
6. They generate progress status and reports up to the end of the requested production.
7. Each time a higher product is generated by the PPF, the PGF receives a completion status which allows it to recover the file, from the PPF local disk to its own post-processing storage

A PPF in dynamic mode will stop by itself at the end of each production request completion. A PPF in static mode will stop by itself at the end of the dump processing.

The PGF is interfaced with the PPF through a set of generic services offering a standard way to prepare, start, control and collect the results.

This design provides a high level of flexibility as:

- The PPFs are working locally on the computers they are implemented. They can be activated in standalone for algorithm validation,
- The PPFs are scalable. The PGF can deal with a subset of instruments PPF S/W. This provides a high benefit for integration. PPF can be added or upgraded at any time into the production chain, with no impact on the overall production,
- All PPF can be executed on any processing node of the PGF.

6.2.2 PGF to PPF interfaces

Previously to the raw data transmission on which the processing occurs, the PGF will transfer context data to each PPF local disk. This data transmission is performed before the PPF S/W is started:

- The **Auxiliary Data** files are those to be used by the PPF production. The PGF selects the aux. data to be used for the processing,

- The **Product Model** files give a description of each type of product to be generated. Each model is made of a product MPH. The PGF first setting up general information into the MPH (station, satellite ID, etc,...) then the PPF updates the data fields describing the product,

- The **Work Order** file is used to transfer additional information needed for processing, such as auxiliary data and product model files names, and sensing coverage of the data to process.

The PPFs are controlled by the PGF through several commands:

- **Start** activates a PPF after the corresponding computer has been feed up with appropriate auxiliary data and production parameters,

- ***Step***: One "step" command is sent for each new L0 slice or L1 PDU to be processed by the IASI L1 PPF. The step command will be sent by PGF to PPF according to the availability of input data and auxiliary data needed for the production. A "step" command will be linked to a new "Work order".

- **Suspend** pauses the current processing of PPF. When the command is received, the current processing stage is completed, but subsequent stages are not started,

- **Resume** restarts the current processing. After a *Suspend* command, when the *Resume* command is received, PPF continues its processing by starting the next processing stage,

- **Stop** is a soft shutdown. The PPF terminates its own proce,ssing stage, and exits properly after running a cleanup procedure on its local context,

- ***Break*** : stop of the current production, the PPF shall terminate the current production properly (with report generation and products already computed available) and start to process the next "step" command (if any). It is not a stop of the PPF itself.

- **Abort** kills a current processing. The PPF will be forced to terminate by a S/W signal sent by PGF.

The PPF answers to all commands through a **Command TM**, which indicates the Success/Failure of the received command.

When the PPF has been initiated, the PGF provides the **Unprocessed Data** to the PPF. They may contain full level 0, or full Level 1b product(s) including MPH, SPH and MDS records, as described in the Product Specification.

Processing-Reports and **Stage TM** are regularly generated by working PPF, and sent back to the PGF:

- **Stage TM** contains information sent to PGF asynchronously. It is sent whenever there is a change to the processing state (e.g. Processing to Suspended) and when a processing stage is achieved. The last status is 'processing_complete', which indicates to the PGF that the work order is over, it is always coupled to a processing report, highlighting successful and/or failed PPF activities.

- **Processing-Reports** are operator readable files created when products are available, when fatal processing error occurs, or when the stop command is invoked details the processing, it results with indications on products generated (e.g. names, sensing coverage, PCD, etc.), and on eventual failures.

6.2.3 PPF multiple S/W versions

Two versions of each instrument PPF software can coexist in any PPF computer at the same time. There is no interference on the different versions of a same algorithm processing, meaning that only one version can be executed at any time.

The PGF switches from one version if PPF to the other on MCS request.

The capability to use a subsequent version of instrument PPF is particularly useful on G/S-2 PGF, which is nominally dedicated to backlog, reprocessing and algorithm validation.

When a new algorithm version is available on CGS, it is mandatory to validate it with real products. Therefore, It can be installed on the G/S-2 PGF, in complement to the current version. Then, on a case per case basis, the PGF can be scheduled by MCS in view to perform an higher level processing from existing lower level products at UMARF, and to generate higher level product with the new PPF S/W.

1. Through a reprocessing MCS command, the PGF is invoked on the new version of S/W it uses for the required instrument,
2. The PGF switches off the current version,
3. The PGF switches on the new version to be validated,
4. Then, processing goes on with the new algorithm, until completion,
5. Higher levels products are retrieved from PPF,
6. New version is switch off and replaced by the default one still available.

Higher level products quality are then assessed off line, in view to validate the new algorithm. During this time, previous version can be used for further processing onto G/S-2.

Once the new version has definitively been validated, the old version is removed, and the new one will become the default version for any future production.

6.2.4 Multiple PPF Instances

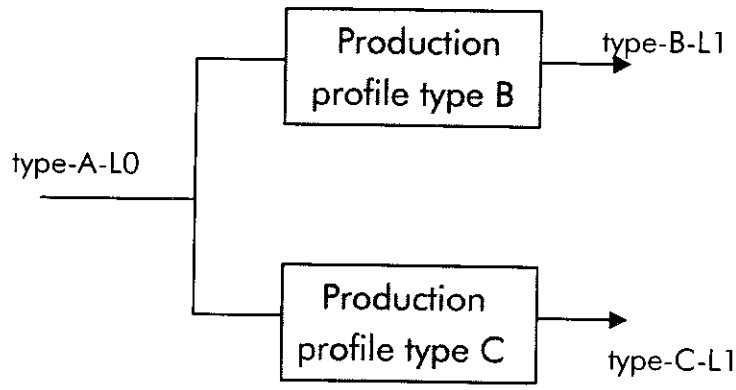
Any NRT L0 product acquired from FEP granules, is be able to be processed to higher level products through different instances of independent PPF S/W in parallel; In the PGF configuration file it is possible to define one or more types of inputs mapping with one or more types of outputs. This is illustrated by the following example.

Figure 6.2–2 Example of production rules/product flow

Production rules

Input type-A-L0
    Output type-B-L1
    Output type-C-L1
End
Output type-B-L1
    Input type-A-L0
End
Output type-C-L1
    Input type-A-L0
End

Product flow



L0 product of type-A is used for generating L1 products of type-B (through one PPF with profile " type-B "), and L1 product of type-C (through another PPF with profile " type-C ").

In the above example Type B PPF and Type C PPF are managed as different "instrument processor" from the configuration management point of view (they are different Configuration Items), even though they process the same onboard instrument data.

The above behaviour allows to have different PPF packages to process the same input flow, providing that the output products naming are different (in other word a given Inputs X outputs products is enabling the selection of a single PPF package).

It shall be noticed that the above depicted production rules configuration file can be tuned to the maximal extend so that,

- Several L0 products can be used for generating one L1 product (ATOVS case),
- Level 2 production can be activated separately from L1 production (ATOVS & IASI case),
- A PPF could be started even if non-mandatory inputs are not still available (AVHRR L1 for IASI).

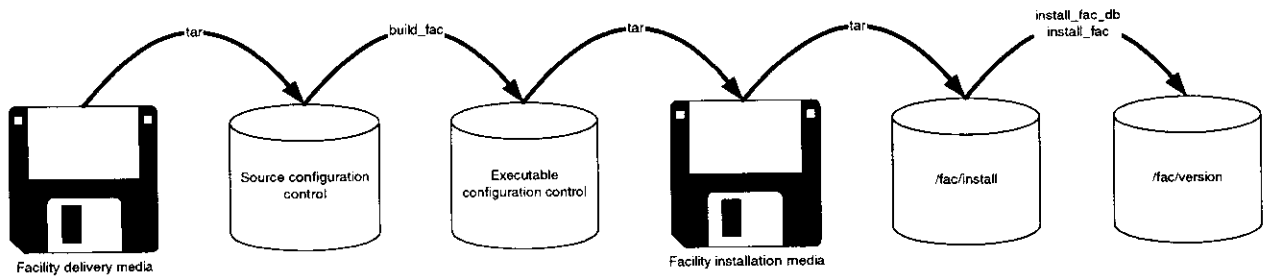
As a summary of the above configuration capabilities, the Production rules allow to specify any type of inputs (not only products) for any type(s) of output(s) (products and actions to be done from these inputs). This constitutes a real warranty that PGF is tuneable, in case new products or PPF are added during the mission.

6.3 FACILITY DELIVERY AND INSTALLATION

6.3.1 Preamble

The Facility delivery format shall be compatible with the process illustrated in Figure 6.3–1 Facility Delivery, Build and Installation Process, according to the requirements below.

Figure 6.3–1 Facility Delivery, Build and Installation Process



6.3.2 Facility Prerequisites

From on-site acceptance onwards, the environment (hardware, operating system, COTS, disk space and organisation, ...) for building and running the Facility application will be established by ASPI. The necessary characteristics of this environment shall be identified by the Facility supplier and agreed with ASPI, such that the Facility is warranted as running correctly in any environment complying with those characteristics.

The initial list of pre-requisites shall be submitted for agreement at Facility CDR. Any modifications to the pre-requisites as the development progresses shall be submitted for agreement at Facility progress meetings. The definitive prerequisites shall have been agreed by Facility VRR at the latest.

The pre-requisites shall appear in the Facility Release Notes.

The COTS product versions given as pre-requisites shall be explicitly certified by the supplier as compatible with the Operating System version given as a pre-requisite.

The procedures (documents, scripts) used to establish the Facility factory environment compliant with the pre-requisites shall be configuration-controlled and delivered with the Facility. These procedures shall start from erased disks and OS/COTS suppliers' delivery media.

The present document contains requirements on the Facility implementation, including the pre-requisites.

6.3.3 Facility Delivery

Deliveries may be of two sorts : full versions or patches. The type of delivery shall be formally agreed between ASPI and the Facility supplier before delivery. Patch deliveries are typically a rapid response to urgent problems.

6.3.3.1 Full Version Deliveries

The Facility shall be delivered in the form of application source files. The term source files is not restricted to source code, but covers all the make files, data files and other editable data which serve to build the Facility run-time environment. Delivery in the form of pre-installed disk images is forbidden (no absolute path in the archive media).

The delivery media shall be CD-ROM.

The delivery shall include a paper copy of the version Release Notes. The Release Notes shall describe the changes in the Facility functionality since the previous release (explicitly referenced). In particular any changes to application configuration files shall be highlighted. Corrected NCRs shall be listed and cross-referenced to the changes in functionality when the solution is apparent in this way. ARs/NCRs still open shall also be listed. Identified work-around solutions to open problems shall be described.

An electronic copy of the Release Notes shall be present on the delivery media.

The application source shall be packaged as a single tar file. The tar file shall contain a single directory tree structure, to be extracted below the current directory (.). The root delivery directory shall be called fac_*version_number*, where *version_number* denotes the delivered version of the Facility.

The root delivery directory shall contain a text file version_contents, listing the versions of all the delivered files, according to the Facility supplier's internal configuration control system. The first line of this file shall identify the preceding version from which the delivered version is derived.

6.3.3.2 Patch Version Deliveries

Patch deliveries may be made by electronic means (network transfer of the tar file).

Such deliveries patch source items. Patches to binary files are forbidden.

The patch Release Notes shall indicate the base version to be patched. The Release Notes shall be a complete document describing the Facility version resulting from the application of the patch to the base version as though it were a fully-delivered version (section 6.3.3).

The tar file shall contain a single directory tree structure, to be extracted within the root directory of the Facility base version.

The set of directories/files resulting from the extraction of the patch within the base version directory shall be compliant with the rules for a full version delivery (section 6.3.3.1). The build and installation process shall then proceed as for a full version delivery (section 6.3.4).

6.3.4 Facility Build

The root delivery directory shall contain an executable file build_fac taking 2 directory names as the first parameters. Running this file shall create the first directory and generate in it the installation directory structure to be transferred to the target hardware. All files generated by the build process and not destined for the installation (eg. log files) shall be created under the second directory.

All file names used in the build process (includes, scripts, ...) shall be relative to an environment variable FAC_ROOT which indicates the root delivery directory.

6.3.5 Facility Installation

The Facility installation procedure shall start assuming an environment compliant with the pre-requisites (section 6.3.2) into which the Facility installation directory structure (section 6.3.4) has been transferred.

The Facility installation procedure shall de divided into 5 distinct, successive main steps : creation of the database physical structure, creation of the database logical structure, initialisation of the database contents, application software installation, application software configuration.

It shall be possible to re-install a Facility beginning at any one of the main steps, providing the preceding steps have not changed from the previously-installed version.

The installation procedure shall be written as far as possible in machine-executable (script) form, rather than human-executable (document) form.

Installation scripts which directly perform installation actions shall have all significant options (names, sizes, privileges, ...) as input parameters. The values of these parameters shall be given by higher-level calling scripts and/or separate data files.

At least at the end of each major step, the installation procedure shall verify that the installation is successful. The procedure need not handle errors raised, but must make available to the installer the diagnostic information qualifying the error.

The root installation directory shall contain 3 executable files create_fac_physical_db, create_fac_logical_db, initialise_fac_db, which shall perform the installation and initialisation of the facility's Oracle database. The database installation shall check for the presence of a pre-existing database and, if found, shall only replace it upon positive confirmation from the installer. The database installation procedure shall be divided into separate scripts for the creation of each type of entity (tablespaces, tables, indexes, ...).

The root installation directory shall contain 2 executable file install_fac, configure_fac which shall install and configure the Facility application in a version directory (section 4.3.4.3

6.4 FACILITY IMPLANTATION IN TARGET ENVIRONMENT

6.4.1 Operating System Installation and Configuration

The Facility application processes shall run under a user named fac, excepting any processes resulting from an interactive user logon or requiring special privileges.

Other Facilities accessing the Facility disk space (/fac/interface/fac1) shall do so under a specific user ID called fac1. The login directory shall be /fac/interface/fac1.

No application software shall run under the user root. If the need arises, privileged user IDs shall be created as required.

The installation procedure shall use only symbolic user IDs.

6.4.2 Application Interface to Target Environment

Command scripts shall be written in Korn Shell (ksh).

The Facility shall provide a command /fac/cmd/fac.env which sets up the application environment variables, including PATH.

The Facility shall provide a command /fac/cmd/fac as follows :

- fac version reports the current Facility version number,
- fac status : reports the Facility application software status as follows : stopped, running nominally, running degraded (with diagnostic),

- fac start : starts the application,
- fac stop : stops the application cleanly,
- fac abort : stops the application immediately, come what may,
- fac clean : restores the Facility data spaces to a state from which the application can be restarted,
- fac forwards : prepares the application (stopped) for a date change into the future, saving any erased data to a backup area,
- fac backwards : prepares the application (stopped) for a date change into the past, saving any erased data to a backup area.

6.4.3 Test Tools Command Line Interface

Test tools shall present a command line interface compliant with Unix conventions.

It shall be possible to execute test tools manually from the shell command line or remotely via rsh.

6.4.4 Test Tools M&C Interface

For the purposes of this specification, test tools are divided into two categories :

- Off-line tools, such as Test Data Preparation (TDP) tools and Test Data Analysis (TDA) tools (analysis part), which are activated manually before or after test runs.

- On-line tools, such as Test Data Generation (TDG) Tools and Test Data Analysis (TDA) tools (ingest part), which run in real time as part of the test set-up.

On-line test tools shall present a monitoring and control interface functionally similar to that of the Facility interface with the MCS, but using the standard Unix process interface as follows :

- stdin : input of commands (TC) to the tool,

- stdout : output of tool status telemetry (TLM) and command acknowledgements (ACK), allowing to monitor test progress,

- stderr : output of log messages (EVT), allowing to determine test results,

- filename following the -log switch : error log file.

The syntax of data exchanged on these interfaces shall be that used in the MCS Local Agent (MLA) API stub input/output files [R1].

The identifier strings used on these interfaces shall follow the same rules as for the Facility M&C interface with the MCS [R2]. They shall not be registered in the Facility data registration document [R2], but in the Test Tool User Manual. The Facility ID shall be TT (Test Tools). The Additional identification field shall identify the tool.

7. OPERATIONAL REQUIREMENTS

PPF-0820		FUNCT	TEST
The PPF shall support the processing or reprocessing of individual data sets or of sequences of data sets extending up to the complete holding of data acquired from the instrument.			
PPF-0830	CGS-SYS-7.1-0040 CGS-SYS-7.1-0050	FUNCT	ANALYSIS
The PPF shall be able to process in NRT the incoming data during commissioning and routine phases			
PPF-0840	CGS-SYS-7.3-0070 CGS-DPG-5.3.3.3-0090	FUNCT	ANALYSIS
In support to the CGS backlog scenario, the PPF shall resume the processing considering the information that was archived in the PPF context when the original NRT processing task was suspended			
PPF-0850	CGS-SYS-7.2-0060	FUNCT	TEST
In case of failure of an external auxiliary data provider, the PPF shall limit the impact of these missing inputs, including: 1) use of alternative auxiliary data for those product where it is possible (previous ones, reference ones..) 2) Flagging the resulting product as not having been generated with the nominal set of auxiliary data			
PPF-0860	CGS-DPG-5.3.1-0030	DES	TEST
The PPF shall be robust against the event that received data is corrupted (aux data or level 0), is duplicated (e.g. L0 down linked twice), is out of time sequence, is incomplete or is missing.			
PPF-0870	CGS-DPG-5.3.5-0210	FUNCT	TEST
In case of missing data needed for the processing (e.g. missing NRT data, missing auxiliary data, missing products from other PPF) or the data being truncated or incomplete, the PPF shall be able to generate a degraded product (the product will be flagged as being degraded).			
PPF-0880	CGS-DPG-5.3.3.4-0120	FUNCT	ANALYSIS
The PPF shall have the capability to reprocess already successfully processed products as well as products previously generated in a degraded way (e.g. because some aux data was missing that is now available).			

THIS PAGE IS INTENTIONALLY LEFT BLANK

8. CORE GROUND SEGMENT WIDE REQUIREMENTS

PPF-0890	-	DES	ANALYSIS
In order to fulfil CGS wide requirements, and according to the Facility breakdown approach, the PPF shall fulfil common system requirements as defined in the document CGS Facility common design requirements AD8.			
PPF-0900	CGS-GSW-8.4.1-0010	MAINT	ANALYSIS
All elements of the PPF shall be maintainable over the lifetime of the CGS			
PPF-0910	CGS-GSW-8.4.1-0020	MAINT	TEST
All PPF elements shall provide the necessary means that allow first level maintenance staff to: 1) investigate, diagnose and correct any failure relevant to first level of maintenance 2) reproduce a problem, in particular, any time it is possible, the configuration leading to the crash shall be saved.			
PPF-0920	CGS-SYS-8.4.1-0100	DES	ANALYSIS
All functions within the PPF shall be designed with a modular upgrade/exchange capability.			
PPF-0940	CGS-GSW-8.4.2-0070	DES	ANALYSIS
The PPF shall offer the necessary level of modularity to allow system functional or performance evolutions with limited impact on the system design (i.e. addition of a new functions has an impact limited to interface; modification of a functional or performance requirement only impacts the element implementing that function).			
PPF-0950	CGS-GSW-8.4.2-0100	MAINT	ANALYSIS
The PPF shall be robust against known anomalies of its COTS elements (e.g. if UNIX is used as an operating system, the CGS must be robust against the known UNIX date problem).			
PPF-0960	CGS-GSW-8.4.1-0120	MAINT	TEST
It shall be possible to release a new version of a PPF item without impact on the availability of the currently installed version.			
PPF-0970	CGS-GSW-8.4.2-110	FUNCT	TEST
It shall be possible to inhibit or enable any automatic function implemented within any PPF function.			

THIS PAGE IS INTENTIONALLY LEFT BLANK

9. TESTING AND TEST TOOLS

PPF-1010	CGS-SYS-9.2-0040 CGS-SYS-9.3-0010	TEST	INSPECTION
The PPF tests will include unit tests, the simulation of data lost or corrupted, disconnection ..., and tools aimed at assessing the pertinence of the computed data.			
PPF-1020	CGS-SYS-9.2-0090	TEST	TEST
The tests shall include the comparison against CFI reference data.			
PPF-1030	CGS-SYS-9.2.2-0100	TEST	INSPECTION
A test tool shall allow to prepare offline the data to form a consistent dataset. It allows, in particular to modify and extend a given product.			
PPF-1040		TEST	INSPECTION
System oriented tests will be defined to check: the proper response to commands, abnormal situations (missing files, corrupted data, ...) and to validate the interfaces with the PGF. They require the simulation of the PGF delivering granules and possible aux data at a regular pace. These tests are defined by the sub-contractor, their result is OK or Failed. They will be chained in a single non-regression test with no manual interaction and a main result: all tests passed or n tests failed.			
PPF-1050		TEST	TEST

The data testing for numerical quality cannot be reduced to a pass/failed result, it can be performed in 2 different ways :
a direct product comparison for testing against reference data, it is launched by a command line of the type
`compare ATOVS_007_xxx ATOVS_AAA_ref config.cfg report7.txt`
and compares the 2 first files for the items specified in the third file and generates a report in the fourth one.
Customised analysis and displays, (e.g. histograms, differences, plots, images) through a scripting language (IDL TBC). Any analysis is then a series of some routine calls, e.g. `extract(file="ATOVS_xxx", field="sigma0", out U[]);`
`extract(file="ATOVS_yyy", field="sigma0", out V[]);`
`W = V - U`
`plot(W[])`			

PPF-1060		TEST	TEST
The direct product comparison shall generate a report listing the items actually compared and the discrepancies found.			
PPF-1070		TEST	TEST

The direct product comparison allow to restrict the comparison to certain items among MPH, SPHs, DSDs (a Data Set Descriptor includes: name, size and location of the associated data in the file), and actual data according to the following hierarchy:

a selection of *Data Sets*,

a selection of *records* in a data set, specified through a given time span,

a selection of *fields* in each record type.

PPF-1080		TEST	INSPECTION
When run from the command line, the selection of items to be compared is given through a config file			

PPF-1090		TEST	TEST
The product comparison tools shall be able to perform time conversion from UTC to MJD and from MJD to UTC so that data (such as the time interval mentioned below) can be input in either format.			

PPF-1100		TEST	INSPECTION

The interactive functionnalities (histograms and display) are embedded in a scripting language (IDL TBC) by means of predefined routines:

routines to extract data from a product,

e.g. `routine` **`extract`**

in: `filename, DSname, record_range or time_span, field`

out: `data[], time_stamps[]`

routines for comparison, histograms, ...,

e.g. `routine` **`compare`**

in: `U[], V[], N, comparison_threshold`

out: `index_where_different[], Nindex,`

routines for plotting curves, images, ...,

e.g. `routine` **`plot`**

in: `x[], y[], N,`

out: *a window with zooming and print capabilities*

PPF-1110		TEST	TEST
The direct product comparison can be run from the command line without any GUI – both input and output are done from and to files.			

PPF-1120		TEST	TEST
A display (e.g. routine plot) results in opening a window drawing a given set of data. The interaction with the window is limited to zooming and print capabilities, no point and click interface is needed.			

PPF-1130		TEST	TEST
A self-test (ready to run) will allow to easily check the correctness of an installation (host config & build)			

10. GLOSSARY

Please refer to the document "Glossary of terms and abbreviations list [EPS-ASPI-LI -0010].

END OF DOCUMENT