DUE GLOBVAPOUR

Processing System
Validation Plan

Issue 1 Revision 0

19 January 2012

Project nr:	ESRIN/AO/1-6090/09/I-OL
Project Coordinator:	Marc Schröder Deutscher Wetterdienst marc.schroeder@dwd.de

Document Change Record

Document, Version	Date	Changes	Originator
0.1	05.07.2011	TOC	Bö
0.5	13.07.2011	Version for BC-internal review	Bö
0.8	14.07.2011	Version for review	Bö, OF, RQ
0.9	09.01.2012	Test procedures	Bö
1.0	19.01.2012	Version 1.0	Nadine Schneider

Table of Contents

1 Introduction 4

1.1 Purpose and Scope 4
1.2 Applicable and Reference Documents 4
1.3 Terms and Acronyms 4
1.4 Document Overview 5

2 Processing System Overview 6

2.1 Role of the Processing System in the GlobVapour Project 6
2.2 Processing System Components 6
2.3 Operational System Scenarios 9

3 Software Integration, Verification and Validation Process 11

3.1 Organisation and Schedule 11
3.2 Items under Test 11
3.3 Test Environment 11
3.4 Test Data Identification 12

4 Test Cases 14

4.1 Test Cases Overview 14
4.2 Test Cases for Processor Runs 14
4.3 Test Cases for Production with Processing Chains 16
4.4 Contingency Test Cases 18
4.5 Test Cases for Data Management 19

5 Test Procedures 20

5.1 Test Procedures for Processor Runs 20
5.2 Test Procedures for Production with Processing Chains 25
5.3 Test Procedures for Contingency Cases 28
5.4 Test Procedures for Data Management 27
5.5 Test Summary 28

6 Requirements Traceability 32

6.1 Verification Methods 32
6.2 System Requirements Traceability 32

1 Introduction

1.1 Purpose and Scope

This document describes the plan of test cases for the GlobVapour stand-alone processing system. The stand-alone processing system is a deliverable of the ESA GlobVapour project. It shall ensure that besides the project partners ESA can in principle run the generation of GlobVapour products.

The document identifies the test cases, describes the test data and the test environment, and traces the requirements to test cases and vice versa. The document will be the basis of the development of test procedures for the acceptance tests. It also sets the scope for the test report to be generated for the acceptance review at the end of the acceptance tests.

1.2 Applicable and Reference Documents

The following documents are applicable to this test plan.

ID	Title	Issue	Date
[AD 1]	DUE GLOBVAPOUR Statement of Work (SoW) EOEP-DUEP-EOPS-SW-09-0003	1.1	13 May 2009
[AD 2]	DUE GLOBVAPOUR Proposal	1.3	09 July 2009
[AD 3]	DUE GLOBVAPOUR Software Development Plan (SDP)	1.0	16 April 2010
[AD 4]	DUE GLOBVAPOUR Processing System Design (DD)	0.8	10 July 2011

The following documents are referenced in this test plan.

ID	Title	Issue	Date
[RD 1]	DUE GlobVapour PM3 Meeting Minutes	1.0	29 Mar 2011
[RD 2]	DUE GlobVapour PM4 Meeting Minutes		
[RD 3]	Technical Note on GV Processing Chains	0.7	19 May 2011
[RD 4]	ECSS E-ST-40C Space Engineering - Software	C	6 March 2009
[RD 5]	ECSS Q-ST-80C Software Product Assurance	C	6 March 2009

1.3 Terms and Acronyms

The following specific acronyms are used within this document:

Acronym	Definition
(A)ATSR	(Advanced) Along Track Scanning Radiometer
DUE	ESA's Data User Element program
ECSS	European Cooperation on Space Standardisation
GOME	Global Ozone Monitoring Experiment
GV	GlobVapour

MERIS	Medium Resolution Imaging Spectrometer
SCIA	Sciamachy instrument on ENVISAT
SSM/I	Special Sensor Microwave/Imager
TCWV	Total Column Water Vapour

The following terms are used within this document with a specific meaning:

Term	Definition
monitor	Software entity that encapsulates a resource and cares for the synchronisation of calls that use this resource. The processing monitor presented in this document encapsulates concurrent threads and processes on different hosts and ensures a well-defined level of concurrency.
processor	Software entity that transform earth observation input data into output data, usually one level above the input in the CEOS nomenclature [RD 5]

1.4 Document Overview

The structure and content of this document follows [RD 4], annex K (ECSS software test plan document requirements).

After this formal introduction

section 2 provides an overview over the GlobVapour stand-alone processing system

section 3 describes the validation process with items under test, test environment, and test data

section 4 identifies and defines the test cases

section 5 provides test procedures to execute and document the tests

section 6 traces system requirements to test cases

2 Processing System Overview

The GlobVapour stand-alone processing system integrates three different processing chains to generate TCWV products and WV profiles.

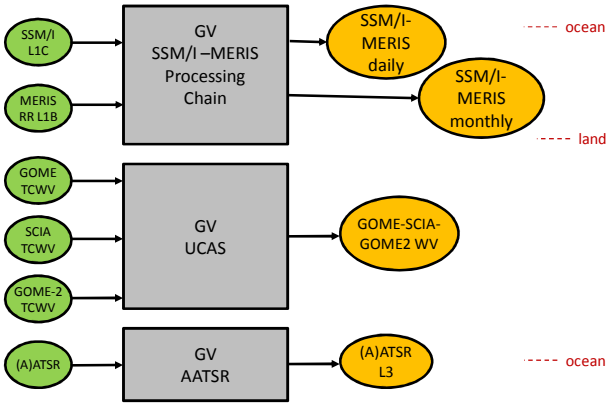

Figure 2-1: GlobVapour Processing Chains

Some of the chains blend results from two different missions to generate a global product with daily, weekly or monthly temporal resolution:

- SSM/I and MERIS
- GOME, SCIAMACHY and GOME-2
- (A)ATSR (experimental, only over ocean, no blending)

The IASI-SEVIRI chain has been dropped for the stand-alone processing system.

2.1 Role of the Processing System in the GlobVapour Project

Operational processing of the GlobVapour dataset will be performed at DWD, DLR, and FUB on separate systems during the runtime of the project. It is not the purpose of the stand-alone processing system to serve this operational requirement.

Purpose of the GlobVapour stand-alone processing system is to demonstrate the functional chains based on a subset of the overall inputs. The system shall be able to process any data from the GlobVapour input time interval if the data is provided. It will be integrated on a single machine. The approach selected shall in principle allow scaling the system to support operational re-processing of the GlobVapour dataset.

2.2 Processing System Components

The stand-alone processing system is functionally decomposed into three main components as shown in Figure 2-2.

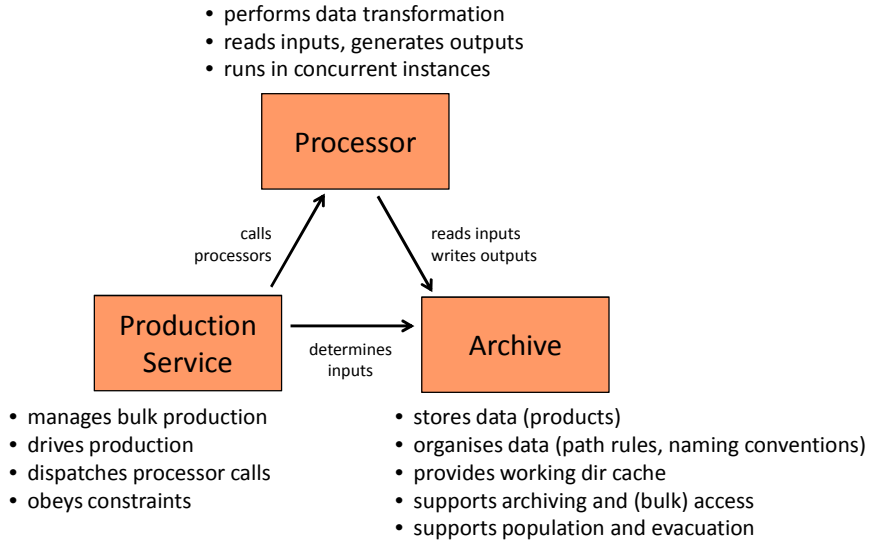

Figure 2-2: GlobVapour processing system with processors, archive and production service

- The processors are software entities that perform the earth observation data transformation. They read inputs from the archive and store their outputs into the archive, either as temporary data or as result products. For the GlobVapour processing system processors are provided by DWD, DLR and FUB.
- The archive stores input and output earth observation data products of different levels. It organises data by path rules and naming conventions. In the GlobVapour processing system the archive is implemented by a file system directory tree. It also provides different working directories for processing to concurrent processes.
- The production service controls the execution of processing chains and manages processing resources and concurrency. In the GlobVapour processing system the production service is implemented as a set of simple (Python) scripts and a processing monitor that maintains dependencies and manages resources.

The processors of the GlobVapour processing system are listed in Table 2-1.

Table 2-1: Processors of the GlobVapour processing system

Processor	Provider
SSM/I level 2 *1dvar*	DWD
SSM/I level 3	DWD
MERIS level 2	FUB
MERIS level 3	FUB
MERIS monthly aggregation	FUB
SSM/I-MERIS blending	FUB
GOME level 3 **UCAS**	DLR
AATSR level 2	FUB

The archive file system has a structure depicted in Figure 2-3. It comprises directories for inputs and outputs under eodata, the cache with directories created for each processing task, and the software and processing system instance directories.

```
/data/gv
/
        eodata/
                MER_RR_1P/      r03
                                /
                                        2005
                                        /
                                                01/
                                                        01/
                                                                MER_RR__1PNUMP20050101_134558_...0001.N
                                                                1
                                                        .../
                                                        31/
                                                ...
                                                /
                                                12/
                                        .../
                                        2009
                                        /
                MER_WV__2P/     v1/
                                .../
                MER_WV__DC/     v1/
                                .../
                MER_WV__MM/     v1/
                                .../
                SSMI_L1C/       ?/
                                .../
                SSMI_L2/        v1/
                                .../
                SSMI_L3/        v1/
                                .../
                SSMI_MERIS_DC/  v1/
                                .../
                SSMI_MERIS_MM   v1/
                /
                                .../
                GOME_TCVW/      ?/
                                .../
                SCIA_TCVW/      ?/
                                .../
                GOME2_TCVW/     ?/
                                .../
                GOME_SCIA_DC/   ?/
                                .../
                GOME_SCIA_MM    ?/
                /
                                .../
                ATS_TOA_1P/     ?/
                                .../
                ATS_L2/         ?/
                                .../
        cache/
                job_0001/
                                task_0001_000001/
                                                        intermediate files
                                .../
                .../


        software
        /
                1.0/
                                gv-meris-l2-1.0.0/
                                                bin
                                                /
                                                        gv-meris-l2.sh
                                .../
                                pmonitor-1.0.0/
                                                bin
```

```
                                                                    /
                                                                        .../
        inst/
                    gv-ssmi-meris/
                                                mygv
                                                requests/
                                                log/
```

Figure 2-3: GlobVapour processing system directory tree

The user interface to the production service consists of a set of requests in form of executable scripts and status and report files logging progress and results of processing. The scripts define which data is to be processed by which processing chain, and where the output shall be archived. The processing system schedules concurrent and optionally distributed processing. The running processes for the different processing steps are also part of the processing system.

The call sequences of a concurrent execution with a processing monitor is shown in Figure 2-4 (temporal subset, there are more calls in reality).

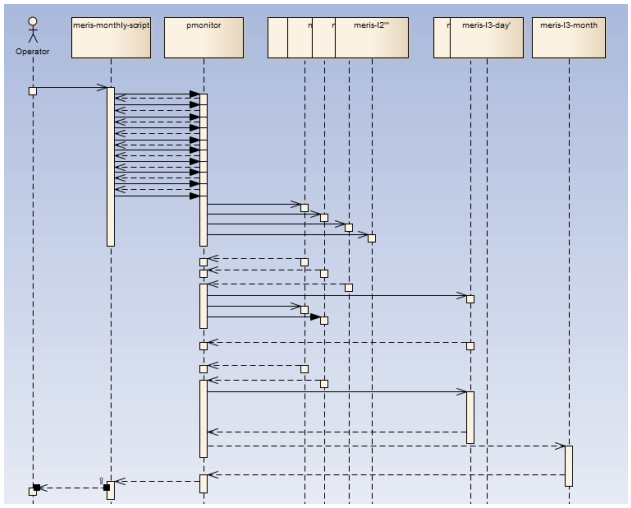

Figure 2-4: Processor call sequence of a GlobVapour processing script

Features of the processing monitor layer are:

- Immediate return of control to the script after calls for meris-l2, meris.l3 and aggregation
- parallel execution with limited concurrency
- input dependencies between meris-l2 and meris-l3-day etc. considered

2.3 Operational System Scenarios

Besides the different processing scenarios there are some necessary system scenarios to provide the inputs and get out the outputs. The system scenarios are shown in Figure 2-5.

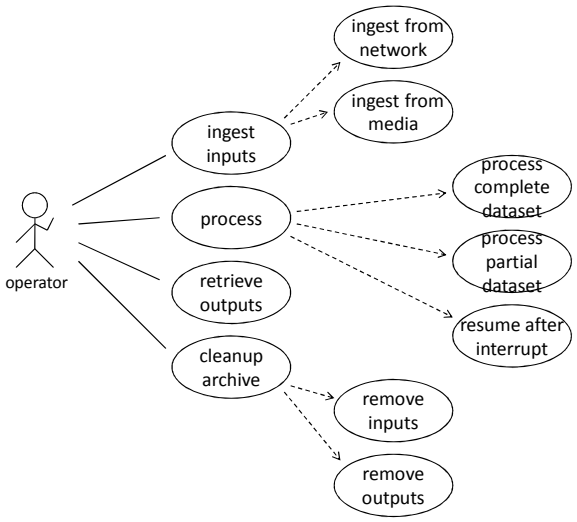

Figure 2-5: GlobVapour processing system use cases

- An operator copies inputs into the processing system archive from a network source by FTP or from a medium, e.g. an external disk. The data is stored according to the archiving rules.
- The operator processes data of one month using one of the provided python scripts, e.g. for SSM/I and MERIS processing.
- The operator suspends and resumes processing.
- The operator resumes processing after failure, retrying or skipping the failed step.
- The operator re-processes certain parts of the input data or certain steps.
- The operator copies the output files to a network target by FTP or to a medium, e.g. an external disk.
- The operator deletes inputs and outputs from the archive to get space for new inputs and outputs.

Other scenarios are the initial installation or the update of processors to new versions.

3 Software Integration, Verification and Validation Process

3.1 Organisation and Schedule

Corresponding to the role of the stand-alone processing system for the GlobVapour project the validation process is planned to be simplified in comparison to the acceptance of operational software systems. At progress meeting 4 [RD 2] it has been decided to merge QR and AR for the stand-alone processing system into one review done at ESA premises. In preparation to this a dry-run of the tests will be performed by Brockmann Consult GmbH and DWD.

The tests will be developed by Brockmann Consult GmbH that also has developed the processing system software. This means that there is no independence of the software development and the test development. But the verification of results is planned to be done by DWD for the dry run and by ESA for the acceptance test.

The result of the test runs will be documented in a test report. The report will mainly comprise the test procedures tables for the steps performed and an additional column with observations and results. An optional partial repetition of verifications depends on AR results.

The dry run can be performed after the final processor versions are available. They are expected during August, with the exception of the final version of the GOME processor. The acceptance test can be performed after the successful dry run. A date shall be fixed after the processors are available. If the GOME processor currently available gets an update after the test runs certain test cases may be re-run to verify the update.

3.2 Items under Test

Table 3-1 identifies the processors and processing system software versions that shall be tested. The versions may change with the delivery of the final processors and maybe with updates of the processing system after the dry run.

Table 3-1: Items under test

Item	Version	Provider
SSM/I level 2 *1dvar* processor	2.00	DWD
SSM/I level 3 processor	2.01	DWD
MERIS level 2 processor	1.5	FUB
MERIS level 3 processor	1.5	FUB
MERIS monthly aggregation processor	1.5	FUB
SSM/I-MERIS blending processor	1.5	FUB
GOME level 3 UCAS processor	0.9	DLR
AATSR level 2 processor	not yet available	FUB
GlobVapour processing system instance	1.0	BC
Processing monitor layer	1.0	BC

3.3 Test Environment

The environment of the stand-alone processing system is a machine with the software pre-installed and runnable. At progress meeting 4 [RD 2] it has been decided to provide the system in

a virtual machine. In addition to this the software items are delivered as source code packages that can be installed in other environments.

As the data volume required for one month is considerable, an external disk is planned to be used as data archive. The disk will be mounted to the virtual machine.

A GlobVapour operator user account gvop will be used for all tests.

The tests will be performed in the instance directory of the processing system, with the directory structure as shown in chapter 2.2. All items under test listed in the previous section are installed in this directory tree. All input data for processing is in the eodata archive in this file system (mounted to the external disk). Input data for verification is provided in a separate directory verification-data/ below the gvop home directory. The test data is listed in the next section.

3.4 Test Data Identification

The test data comprises one month of inputs from SSM/I, MERIS and GOME. It comprises a day from AATSR. This is listed in Table 3-2 together with the number of items and the estimated size. This data will be stored in the archive directory tree.

Table 3-2: Test inputs data

Type	Coverage	# Items	Volume
SSM/I L1C (F13, F14)	1 month (07/2007)	~62	~9 GB
ERA Interim	1 month (07/2007)	~31	~2.9 GB
MERIS RR L1B	1 month (07/2007)	~432	~250 GB
GOME L2 TCWV	1 month	~31	~0.6 GB

Further test "inputs" are pre-computed parts of the outputs in order to speed up tests for the monthly mean products. They are listed in Table 3-3.

Table 3-3: Pre-computed outputs test data

Type	Coverage	# Items	Volume
SSM/I L2 (F13, F14)	30 days	60	2169 MB
SSM/I L3 DC	0 days	0	
MERIS RR L2	~30.8 days	454	59838 MB
MERIS L3 DC	30 days	30	5563 MB
SSMI+MERIS DC	0 days	0	

The verification data comprises one item per output type. These products are generated with the operational systems at DWD, FUB and DLR. They will be used for comparison with the results of the stand-alone processing system during the tests. Table 3-4 lists the data items foreseen.

Table 3-4: Verification data

Type	Coverage	# Items	
SSM/I L2 (F13, F14)	1 day (01/07/2007)	2	
SSM/I L3 DC	1 day (01/07/2007)	1	
SSM/I L3 MM	1 month (07/2007)	1	
MERIS RR L2	1 orbit from 01/07	1	
MERIS L3 DC	1 day (01/07/2007)	1	

Type	Coverage	# Items	
MERIS L3 MM	1 month (07/2007)	1	
SSMI+MERIS DC	1 day (01/07/2007)	1	
SSMI+MERIS MM	1 month (07/2007)	1	
GOME L3 MM	1 month (07/2007)	1	

To test failure handling a defective input from the period that is not pre-processed shall be used. It is listed in Table 3-5.

Table 3-5: Defective input data

Type	Coverage	# Items	Volume
MERIS RR L1B	1 file (31/07/2007)	1	0

To test data management additional inputs from a different period are used. They are listed in Table 3-6.

Table 3-6: Additional input data

Type	Coverage	# Items	Volume
MERIS RR L1B	01/08/2007	15	4010 MB

4 Test Cases

This chapter identifies the test cases and defines them with inputs and expected output.

4.1 Test Cases Overview

Table 4-1 lists the test cases for the GlobVapour stand-alone processing system. Each test case is described by an identifier GV-TC-<number>, a title, and the system requirements from the design document [AD 4] to be verified by the test case (forward tracing). The inverse trace from requirements to test cases is listed in the next chapter.

Table 4-1: GlobVapour stand-alone processing system test cases

ID	Test Case Title	System Requirements
GV-TC 10	SSM/I 1DVAR processor run	GV-SR-50, GV-SR-100
GV-TC 20	SSM/I level 3 processor run	GV-SR-50, GV-SR-100
GV-TC 30	MERIS level 2 processor run	GV-SR-60
GV-TC 40	MERIS level 3 processor run	GV-SR-60
GV-TC 50	SSM/I MERIS blending processor run	GV-SR-60
GV-TC 60	SSM/I MERIS chain production	GV-SR-50, GV-SR-60, GV-SR-90, GV-SR-110, GV-SR-120, GV-SR-150, GV-SR-260, GV-SR-270
GV-TC 70	GOME chain production	GV-SR-70, GV-SR-90, GV-SR-110, GV-SR-130
GV-TC 80	AATSR production	GV-SR-80, GV-SR-90, GV-SR-110, GV-SR-140
GV-TC 90	Interruption and resume	GV-SR-290, GV-SR-300
GV-TC 100	Processing failure handling	GV-SR-280, GV-SR-300
GV-TC 110	Input ingestion	GV-SR-190
GV-TC 120	Result retrieval	GV-SR-200
GV-TC 130	Partial cleanup	GV-SR-210

Each of the test cases is defined with its inputs, expected results and pass/fail criteria in the following subsections.

In addition to this other requirements can be verified by inspection during the test. See section 5 for a list of requirements and their verification method.

4.2 Test Cases for Processor Runs

The first five test cases verify the processor integration of those processors that can be run with a few input products and that do not depend heavily on the complete chain. These test cases also verify that the results generated are equal in content to the operational ones.

ID	GV-TC-10
Title	SSM/I 1DVAR processor run
Description	Runs the SSM/I 1DVAR processor by the GlobVapour stand-alone processing system call script in a working directory, providing parameters for the input file and an empty output directory; reads the command line outputs; finally inspects the output directory for the output product; uses ncdump and diff to determine that the generated output is the same as an operational product for verification.
Inputs	• one SSM/I L1C input `cmsaf.cbtf.ssmi.f13.2007-07-01.l1c.nc` • the corresponding auxilary `ERA_Interim_FC_20070701_0000+12` (in the archive) • output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • one SSM/I L2 product `SSMIxxxxxxx_L2_xx_F13_20070701000000_E_20110712164554.nc` • no differences except for the filename compared to operational output

ID	GV-TC-20
Title	SSM/I level 3 processor run
Description	Runs the SSM/I level 3 processor by the GlobVapour stand-alone processing system call script in a working directory, providing parameters for the input files and an empty output directory; reads the command line outputs; finally inspects the output directory for the output product; uses ncdump and diff to determine that the generated output is the same as an operational product for verification.

Inputs	• two SSM/I L2 products from the two satellites
	`SSMIxxxxxxx_L2_xx_F13_20070701000000_E_20110712164554.nc`
	`SSMIxxxxxxx_L2_xx_F14_20070701000000_E_20110712165005.nc`
	• output file from operational production for verification

Expected results and Pass/Fail Criteria	• processing succeeds • one SSM/I DC product `SSMIxxxxxxx_L3_DC_Fxx_20070701000000_E_20110712232650.nc` • no differences except for the filename compared to operational output

ID	GV-TC-30
Title	MERIS level 2 processor run
Description	Runs the MERIS L2 processor by the GlobVapour stand-alone processing system call script in a working directory, providing parameters for the input file and an empty output directory; reads the command line outputs; finally inspects the output directory for the output product; uses ncdump and diff to determine that the generated output is the same as an operational product for verification.
Inputs	• one MERIS RR level 1 input

	MER_RR__1PRACR20070701_000250_000026412059_00274_27883_0000.N1 • output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • one MERIS L2 product • no differences except for the filename compared to operational output

ID	GV-TC-40
Title	MERIS level 3 processor run
Description	Runs the MERIS level 3 processor by the GlobVapour stand-alone processing system call script in a working directory, providing parameters for the input files and an empty output directory; reads the command line outputs; finally inspects the output directory for the output product; uses ncdump and diff to determine that the generated output is the same as an operational product for verification.
Inputs	• the 14-15 MERIS L2 products for one day 2007-07-01 • output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • one MERIS DC product • no differences except for the filename compared to operational output

ID	GV-TC-50
Title	SSM/I MERIS blending processor run
Description	Runs the SSM/I-MERIS blending processor by the GlobVapour stand-alone processing system call script in a working directory, providing parameters for the input files and an empty output directory; reads the command line outputs; finally inspects the output directory for the output product; uses ncdump and diff to determine that the generated output is the same as an operational product for verification.
Inputs	• one SSM/I DC product SSMIxxxxxxx_L3_DC_Fxx_20070701000000_E_20110712232650.nc • the corresponding MERIS DC product for 2007-07-01 • output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • one SSM/I-MERIS DC product • no differences except for the filename compared to operational output

4.3 Test Cases for Production with Processing Chains

The second set of test cases runs the different processing chains with inputs and outputs in the archive directories. For chains requiring long processing times parts of the results are pre-

computed and the feature of the processing system to continue processing from where it had been interrupted is used to keep test cycles shorter.

ID	GV-TC-60
Title	SSM/I MERIS chain production
Description	Runs the SSM/I-MERIS processing chain on a month of data, with parts of the results pre-computed for this test case such that only the remaining steps are processed; does so by a request script that runs the chain for the month. The processing system retrieves inputs from the archive directory tree, provides cache directories for processing, and finally archives the outputs in the directory tree. It calls the processor scripts used in the first set of tests. The test case finally inspects the output directories for the output product; uses ncdump and diff to determine that the generated MM outputs are the same as the operational product provided for verification.
Inputs	• one month of input data • report file listing the results already partially pre-computed for this test case • archive with pre-computed part of the result • MM output files from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • daily and monthly composites and blended products complete in archive • no differences except for the filename compared to operational output

ID	GV-TC-70
Title	GOME chain production
Description	Runs the GOME processing chain on a month of data. The processing uses the archive for inputs and outputs. It uses the processor wrapper script for the UCAS processor in the processing system; uses ncdump and diff to determine that the generated MM output is the same as the operational product provided for verification. This test case also covers the test of the wrapper script as the processor is only used on a complete month of inputs, and there would be no difference between testing the processor and testing the chain.
Inputs	• one month of input data • MM output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • monthly composite in archive • no differences except for the filename compared to operational output

ID	GV-TC-80
Title	AATSR production

Description	Runs the AATSR processing chain on a day of data. The processing uses the archive for inputs and outputs. It uses the processor wrapper script to be developed for the AATSR processor; uses ncdump and diff to determine that one of the generated outputs is the same as the operational product provided for verification.
Inputs	• one day of input data • L2 output file from operational production for verification
Expected results and Pass/Fail Criteria	• processing succeeds • outputs for the day in archive • no differences except for the filename compared to operational output

4.4 Contingency Test Cases

The following test cases consider cases of interruption and failure handling.

ID	GV-TC-90
Title	Interruption and resume
Description	Runs the SSM/I-MERIS processing chain as in GV-TC-60, interrupts processing by cancelling the script on the shell; resumes processing by calling the script again.
Inputs	• same as GV-TC-60
Expected results and Pass/Fail Criteria	• successfully completed steps are being reported as skipped • processing resumes with the steps that had been interrupted • generated products for the month are completely archived

ID	GV-TC-100
Title	Processing failure handling
Description	Inserts defective test products into the archive as replacement for the corresponding inputs, runs the SSM/I-MERIS processing chain as in GV-TC-60, waits for the run to terminate, determines processing status; exchanges the defective products with the correct ones again, resumes processing by calling the script again alternatively: Runs the SSM/I-MERIS processing chain as in GV-TC-60, determines Unix process that executes the 1dvar processor (with the ps command), kills this process; waits for the run to terminate. Resumes processing by calling the script again.
Inputs	• same as GV-TC-60
Expected results and Pass/Fail Criteria	• defective inputs leads to two steps being reported as failed in command line output and status file • (alternatively: killed process leads to one step being reported as failed in command line output and status file)

	• processing terminates after all processing of independent steps is done • after resume successfully completed steps are reported as skipped • processing resumes with the steps that had been cancelled • processing continues with steps that depend on the cancelled step's output

4.5 Test Cases for Data Management

The final three test cases verify data management with addition of new inputs and retrieval of processing outputs.

ID	GV-TC-110
Title	Input ingestion
Description	Copy one additional month of SSM/I inputs to the archive using Unix commands (mount, rsync)
Inputs	• test data for SSM/I for August 2007 on a medium
Expected results and Pass/Fail Criteria	• test data for SSM/I for August 2007 in the archive

ID	GV-TC-120
Title	Result retrieval
Description	Copy one month of SSM/I DC and MM products from the archive to a folder prepared for burning them on a CD using Unix commands (rsync, mkisofs)
Inputs	SSM/I DC and MM products for 2007-07 in the archive
Expected results and Pass/Fail Criteria	SSM/I DC and MM products for 2007-07 in a folder outside the archive file system

ID	GV-TC-130
Title	Partial cleanup
Description	Verify space available in the archive (df command), remove additional inputs ingested in GV-TC-110 according to procedure in user manual (Unix commands cd, ls, mv, rm), verify space available again
Inputs	• test data for SSM/I for August 2007 in the archive
Expected results and Pass/Fail Criteria	• test data for SSM/I for August 2007 no longer in the archive • archive free space increased

5 Test Procedures

This chapter lists the instructions to perform the tests for the GlobVapour stand-alone processing system defined in the previous chapter.

Version record		
Steps	Notes	Rerun
Record the date of the test run.		
Record the names of the test engineer and test observers.		

Determine the versions of processors used:
`cd inst`
`cat mygv`
`cd`
This prints out paths to processors. Add the versions listed in the notes column in the following steps.		

Record the instance version.		
Record the SSMI 1dvar processor version.		
Record the SSMI l2tol3 processor version.		
Record the MERIS FUB version of all MERIS processors.		
Record the GOME processor version in the notes.		
Record the BEAM version in the notes.		

5.1 Test Procedures for Processor Runs

GV-TC-10 SSM/I 1DVAR processor run		
Steps	Notes	Rerun

To run the SSM/I 1DVAR processor:
`cd inst`
`. mygv`
`cd ~/cache`
`mkdir test-10`
`cd test-10`
`ssmi-l2.sh \`
`/home/gvop/eodata/SSMI_L1C/v1/2007/07/01/cmsaf.cbtf.ssmi.f13.2007-07-01.l1c.nc \`
`/home/gvop/cache/test-10/2007/07`
This will write out messages on stdout initially and will run for some time.
You may monitor CPU usage for the tools "cdo", "collocate.x" and "SSMIS_SAFProg.o" with
`top`		

in another terminal.		

To check the return code
`echo $?`
`# expected value: 0`
To check availability and size of the output file:
`find 2007/07 -ls`
`# expected size: 75797000`		

To check the content
`ncdump 2007/07/SSMI*nc > output.cdl`
`ncdump \`
`~/eodata/SSMI_L2/v1/2007/07/SSMIxxxxxxx_L2_xx_F13_20070701000000_*.nc \`
`> expected.cdl`
`diff expected.cdl output.cdl`
Compare with expected output. Only a single line should differ. Example:
`1c1`
`< netcdf SSMIxxxxxxx_L2_xx_F13_20070701000000_E_20111229203718`
`{`
`---`
`> netcdf SSMIxxxxxxx_L2_xx_F13_20070701000000_E_20120109103017`
`{`		

Delete cache
`cd ~/cache`
`rm -r test-10`		

Enter disposition (passed/to be repeated/failed)		

GV-TC-20 SSM/I level 3 processor run		
Steps	Notes	Rerun

To run the SSM/I l2tol3 processor:
`cd inst`
`. mygv`
`cd ~/cache`
`mkdir test-20`
`cd test-20`
`ssmi-l3.sh /home/gvop/eodata/SSMI_L2/v1/2007/07/*nc \`
`../test-20-output/2007/07`
 	
This will write out messages on stdout and will run for some time.
You may monitor CPU usage for the tools "l2l3_ssmi" with
`top`
in another terminal.		

To check the return code
`echo $?`
`# expected value: 0`
To check availability and size of the output files:
`ls -l ../test-20-output/2007/07/`
`# expected output size of 31 DC files: 4413364`
`# expected output size of 1 MM file: 5450392`		

To check the content		

ncdump \
../test-20-output/2007/07/SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20120109120009.nc \
> mm.cdl
ncdump \
~/eodata/SSMI_L3/v1/2007/07/SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20111230073554.nc \
> mm-expected.cdl
diff mm-expected.cdl mm.cdl
Compare with expected output. Only a few lines should differ. Example:
1c1
< netcdf SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20111230073554 {
--- 	
> netcdf SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20120109120009 {
53c53
< :filename = "SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20111230073554.nc" ;
--- 	
> :filename = "SSMIxxxxxxx_L3_MM_Fxx_20070701000000_E_20120109120009.nc" ;
55,56c55,56
< :timestamp = "20111230073554" ;
< :history = "created on 2011-12-30 07:35:54 UTC (4199)" ;
--- 	
> :timestamp = "20120109120009" ;
> :history = "created on 2012-01-09 12:00:09 UTC (4199)" ;		

Delete cache cd ~/cache rm -r test-20		
Enter disposition (passed/to be repeated/failed)		

GV-TC-30 MERIS level 2 processor run		
Steps	Notes	Rerun
To run the MERIS level 2 processor: cd inst . mygv cd ~/cache mkdir test-30 cd test-30 mkdir -p dim/2007/07/01 beam-pconvert.sh \ /home/gvop/eodata/MER_RR__1P/r03/2007/07/01/MER_RR__1PRACR20070701_000250_000026412059_00274_27883_0000.N1 \ dim/2007/07/01 meris-l2.sh \ `pwd`/dim/2007/07/01/MER_RR__1PRACR20070701_000250_000026412059_00274_27883_0000.dim \ l2/2007/07/01 This will write out messages on stdout and will run for some time. You may monitor CPU usage for the tools "java pconvert" and "python gv_meris_l2_processing.py" with top in another terminal.		
To check the return code echo $? # expected value: 0 To check availability and size of the output files: ls -l l2/2007/07/01 # expected output size: 286028608		
To check the content		

ncdump l2/2007/07/01/MERIS*.nc > output.cdl
ncdump \
~/eodata/MERIS_L2/v1/2007/07/01/MERISxxxxx-L2-xx-xxx-20070701000250-E-20111230024449.nc
\
> expected.cdl
diff expected.cdl output.cdl
Compare with expected output. Only a few lines should differ. Example:
1c1
< netcdf MERISxxxxx-L2-xx-xxx-20070701000250-E-20111230024449 {
--- 	
> netcdf MERISxxxxx-L2-xx-xxx-20070701000250-E-20120109173538 {
44c44
< :filename = "output/2007/07/01/MERISxxxxx-L2-xx-xxx-20070701000250-E-20111230024449.nc" ;
--- 	
> :filename = "output/2007/07/01/MERISxxxxx-L2-xx-xxx-20070701000250-E-20120109173538.nc" ;
51c51
< :timestamp = "20111230024449" ;
--- 	
> :timestamp = "20120109173538" ;
59c59
< :history = "created on 2011-12-30 02:44:49 UTC" ;
--- 	
> :history = "created on 2012-01-09 17:35:38 UTC" ;		

Delete cache cd ~/cache rm -r test-30		
Enter disposition (passed/to be repeated/failed)		

GV-TC-40 MERIS level 3 processor run		
Steps	Notes	Rerun
To run the MERIS level 3 processor: cd inst . mygv cd ~/cache mkdir test-40 cd test-40 meris-dc.sh \ /home/gvop/eodata/MERIS_L2/v1/2007/07/01/MER_RR__2PRACR20070701_*.N1 \ dc/2007/07/01 This will write out messages on stdout and will run for some time. You may monitor CPU usage for the tool "python gv_meris_dc_processing.py" with top in another terminal.		
To check the return code echo $? # expected value: 0 To check availability and size of the output files: ls -l dc/2007/07 # expected output size: 388845844		
To check the content ncdump dc/2007/07/01/MERIS*.nc > output.cdl ncdump \ ~/eodata/MERIS_L3/v1/2007/07/MERISxxxxx_L3_DC_ENV_20070701120000_E_20111230032358.nc \ > expected.cdl diff expected.cdl output.cdl Compare with expected output. Only a few lines should differ:		

1c1
< netcdf MERISxxxxx_L3_DC_ENV_20070701120000_E_20111230032358 {
--- 	
> netcdf MERISxxxxx_L3_DC_ENV_20070701120000_E_20120109163829 {
55c55
< :filename = "output/2007/07/MERISxxxxx_L3_DC_ENV_20070701120000_E_20111230032358.nc" ;
--- 	
> :filename = "output/2007/07/MERISxxxxx_L3_DC_ENV_20070701120000_E_20120109163829.nc" ;
62c62
< :timestamp = "20111230032358" ;
--- 	
> :timestamp = "20120109163829" ;
70c70
< :history = "created on 2011-12-30 03:23:58 UTC" ;
--- 	
> :history = "created on 2012-01-09 16:38:30 UTC" ;		

Delete cache
`cd ~/cache`
`rm -r test-40`		

Enter disposition (passed/to be repeated/failed)		

GV-TC-50 SSMI-MERIS blending processor run		
Steps	Notes	Rerun

To run the SSMI MERIS blending processor:
`cd inst`
`. mygv`
`cd ~/cache`
`mkdir test-40`
`cd test-40`
`ssmi-meris-dc.sh \`
`/home/gvop/eodata/MERIS_L3/v1/2007/07/MERISxxxxx_L3_DC_ENV_2007070112000\`
`0_E_20111230032358.nc \`
`/home/gvop/eodata/SSMI_L3/v1/2007/07/SSMIxxxxxxx_L3_DC_Fxx_200707*nc \`
`dc/2007/07`
 	
This will write out messages on stdout and will run for some time.
You may monitor CPU usage for the tool "python gv_merge_meris_ssmi_processing.py" with
`top`
in another terminal.		

To check the return code
`echo $?`
`# expected value: 0`
To check availability and size of the output files:
`ls -l dc/2007/07`
`# expected output size: 492526332`		

To check the content
`ncdump dc/2007/07/*.nc > output.cdl`
`ncdump \`
`~/eodata/SSMI_MERIS_L3/v1/2007/07/SSMI_MERIS_L3_DC_xxx_20070701120000_E_20111230104307.nc`
`\`
`> expected.cdl`
`diff expected.cdl output.cdl`
Compare with expected output. Only a few lines should differ. Example:
1c1
< netcdf SSMI_MERIS_L3_DC_xxx_20070701120000_E_20111230104307 {
--- 	
> netcdf SSMI_MERIS_L3_DC_xxx_20070701120000_E_20120109170220 {
61c61
< :filename =		

"output/2007/07/SSMI_MERIS_L3_DC_xxx_20070701120000_E_20111230104307.nc" ; --- > :filename = "output/2007/07/SSMI_MERIS_L3_DC_xxx_20070701120000_E_20120109170220.nc" ; 68c68 < :timestamp = "20111230104307" ; --- > :timestamp = "20120109170220" ; 76c76 < :history = "created on 2011-12-30 10:43:07 UTC" ; --- > :history = "created on 2012-01-09 17:02:21 UTC" ;		

Delete cache
`cd ~/cache`
`rm -r test-50`		

Enter disposition (passed/to be repeated/failed)		

5.2 Test Procedures for Production with Processing Chains

GV-TC-60 SSM/I MERIS chain production		
Steps	Notes	Rerun

To remove some results from the archive directory tree:
`cd`
`test/backuptestoutputs.sh`
`find eodata/backup -type f | wc -l`
 	
The number of files moved away is expected to be 267.		

To prepare the instance to a state where these outputs are still to be generated:
`cd`
`mv test/gv.report.test inst/gv.report`
`ls -l inst`
 	
The size of the gv.report file is expected to be 375607 bytes.		

To start the processing system:
`cd inst`
`. mygv`
`gvstartup`
 	
This command is expected to start the processing system and write out one summary status line and four lines for concurrent processes being started:
`1034 created, 4 running, 80 backlog, 950 processed, 0 failed`
`r /home/gvop/inst/bin/gome-l3.sh`
`/home/gvop/eodata/GOME2_L2/v1/2007/07/H2O_2007_07_01.global....`
`r /home/gvop/inst/bin/ssmi-l2.sh`
`/home/gvop/eodata/SSMI_L1C/v1/2007/07/31/cmsaf.cbtf.ssmi.f13....`
`r /home/gvop/inst/bin/ssmi-l2.sh`
`/home/gvop/eodata/SSMI_L1C/v1/2007/07/31/cmsaf.cbtf.ssmi.f14....`
`r /home/gvop/inst/bin/beam-pconvert.sh`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1P...`
The system will run for several hours to complete processing.		

To monitor the processing system, use any of the following. To list the running tasks (r as first letter in line) and maybe failures (f as first letter inline): `cat gv.status` or in a separate window:		

watch -n 10 cat gv.status To list the completed tasks (in a separate window): tail -f gv.report To list operating system processes: gvps|grep inst|grep -v bin/sh|grep -v grep|grep -v gvps To list CPU, memory and I/O usage (in a separate window): top The load is not completely balanced over the time. Some tasks are I/O intensive and will not use all CPUs. Others will.		
Check termination by the summary line of gv.status ..., 0 running, ... Double check by using ps auxww|grep gv.py which is expected NOT to list a process. Check success by gv.status that is expected to contain only the summary line: 1034 created, 0 running, 0 backlog, 1034 processed, 0 failed In case of failures the tasks failed are listed. Reasons can be found in the most recent log files: ls -ltr log/		

The last line of gv.report is expected to contain the path to the SSM/I MERIS MM composite for the test month:
#output /home/gvop/eodata/SSMI_MERIS_L3/v1/2007/07
/home/gvop/eodata/SSMI
_MERIS_L3/v1/2007/07/SSMI_MERIS_L3_MM_xxx_20070701120000_E_....nc
Check the content with
cd ~/cache
mkdir test-60
cd test-60
ncdump /home/gvop/eodata/SSMI_MERIS_L3/v1/2007/07/SSMI_MERIS_L3_MM_*.nc
\
> output.cdl
ncdump \
/home/gvop/eodata/backup/SSMI_MERIS_L3/v1/2007/07/SSMI_MERIS_L3_MM_*.nc
\
> expected.cdl
diff expected.cdl output.cdl
Compare with expected output. Only a few lines should differ. Example:
1c1
< netcdf SSMI_MERIS_L3_MM_xxx_20070701120000_E_20111230104307 {
--- 	
> netcdf SSMI_MERIS_L3_MM_xxx_20070701120000_E_20120109170220 {
61c61
< :filename =
"output/2007/07/SSMI_MERIS_L3_MM_xxx_20070701120000_E_20111230104307.nc" ;
--- 	
> :filename =
"output/2007/07/SSMI_MERIS_L3_MM_xxx_20070701120000_E_20120109170220.nc" ;
68c68
< :timestamp = "20111230104307" ;
--- 	
> :timestamp = "20120109170220" ;
76c76
< :history = "created on 2011-12-30 10:43:07 UTC" ;
--- 	
> :history = "created on 2012-01-09 17:02:21 UTC" ;		

Delete cache cd ~/cache rm -r test-60		

Enter disposition (passed/to be repeated/failed)		

GV-TC-70 GOME chain production		
Steps	Notes	Rerun
GOME processing is performed already in GV-TC-60 as one of the first processing tasks. What remains is the verification of results.		

To check availability and size of the output file:
`ls -l ~/eodata/GOME_L3/v1/2007/`
`# expected output size: 4153904`		

To check the content
`cd ~/cache`
`mkdir test-70`
`cd test-70`
`ncdump ~/eodata/GOME_L3/v1/2007/GOME2.nc > output.cdl`
`ncdump ~/eodata/backup/GOME_L3/v1/2007/GOME2*nc \`
`> expected.cdl`
`diff expected.cdl output.cdl`
Compare with expected output. Only a few lines should differ. Example:
`1c1`
`< netcdf GOME2xxxxxx_L3_MC_M02_20070700000000_E_20111229213421 {`
`---`
`> netcdf GOME2xxxxxx_L3_MC_M02_20070700000000_E_20120110102858 {`
`42c42`
`< :filename = "GOME2xxxxxx_L3_MC_M02_20070700000000_E_20111229213421.nc" ;`
`---`
`> :filename = "GOME2xxxxxx_L3_MC_M02_20070700000000_E_20120110102858.nc" ;`
`44,45c44,45`
`< :timestamp = "20111229203507" ;`
`< :history = "2011-12-29 20:35:09 UTC: merge L3 GDP 4.4" ;`
`---`
`> :timestamp = "20120110093000" ;`
`> :history = "2012-01-10 09:30:01 UTC: merge L3 GDP 4.4" ;`		

Delete cache
`cd ~/cache`
`rm -r test-70`		

Enter disposition (passed/to be repeated/failed)		

5.3 Test Procedures for Data Management

GV-TC-110 Input ingestion		
Steps	Notes	Rerun

Look up additional SSM/I input data:
`cd`
`find eodata/additionalinputs -type f | less`
`find eodata/additionalinputs -type f | wc -l`
15 files are expected to be listed.		

Add the input data to the archive:
`rsync -rv eodata/additionalinputs/MER_RR__1P/r03/2007/08 \`
`eodata/MER_RR__1P/r03/2007`		

Verify that the data has been copied:
`find eodata/MER_RR__1P/r03/2007/08 -type f | less`
`find eodata/MER_RR__1P/r03/2007/08 -type f | wc -l`
The number of files is expected to be the same as in the first step.		

Enter disposition (passed/to be repeated/failed)		

GV-TC-120 Result retrieval		
Steps	Notes	Rerun

Look up result data:
`cd`
`find eodata/SSMI_MERIS_L3/v1/2007/07 -type f | less`
`du -s eodata/SSMI_MERIS_L3/v1/2007/07`		

Add the input data to the archive:
`mkdir -p eodata/gv-export/SSMI_MERIS_L3/v1/2007`
`rsync -rv eodata/SSMI_MERIS_L3/v1/2007/07 \`
`eoatat/gv-export/SSMI_MERIS_L3/v1/2007`
Verify that the data has been copied:
`du -s eodata/SSMI_MERIS_L3/v1/2007/07 eodata/gv-export*`
The size of the export directory is expected to be of the same size as the result directory.		

Enter disposition (passed/to be repeated/failed)		

GV-TC-130 Partial cleanup		
Steps	Notes	Rerun

Determine archive space and use:
`cd`
`df eodata/`
`du -s eodata/*`		

Remove the additional input data from the archive:
`cd`
`mkdir -p eodata/trash`
`mv eodata/MER_RR__1P/r03/2007/08 eodata/trash`
`rm -r eodata/trash`
Verify that the data has been removed:
`du -s eodata/*`
The size of the `MER_RR__1P` directory is expected to be smaller as in the first step.		

Enter disposition (passed/to be repeated/failed)		

5.4 Test Procedures for Contingency Cases

GV-TC-90 Interruption and resume		
Steps	Notes	Rerun

To remove some results from the archive directory tree:
`cd`
`mv eodata/backup eodata/backup1`
`test/backuptestoutputs.sh`
`find eodata/backup -type f | wc -l`
 	
The number of files moved away depends on the final status of GV-TC-60.		

To prepare the instance to a state where some outputs are still to be generated:
`cd`
`mv test/gv.report.test inst/gv.report`
`ls -l inst`
 	
The size of the gv.report file is expected to be 375607 bytes.		

To reduce the concurrency of the system in order to simplify the test case edit `inst/gv.py`. Change
`do_ssmi = True`
`do_meris = True`
`do_gome = True`
to
`do_ssmi = False`
`do_meris = True`
`do_gome = False`
and change
`hosts = [('localhost',19)]`
to
`hosts = [('localhost',5)]`		

To start the processing system:
`cd inst`
`. mygv`
`gvstartup`
 	
This command is expected to start the processing system and write out one summary status line and one line for a process being started:
`332 created, 1 running, 287 backlog, 44 processed, 0 failed`
`r /home/gvop/inst/bin/beam-pconvert.sh /home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 /home/gvop/eodata/MER_RR__1P/dim/2007/07/31`		

To interrupt processing, stop the monitor and if necessary computing processes:
`gvshutdown`
`ps auxww|egrep 'java|gv.py|cdo|collocate.x| SSMIS_SAFProg.out|l2l3_ssmi'`
`kill <pid>`
`# where <pid> is the listed process of the processor`
`ps auxww|egrep 'java|gv.py|cdo|collocate.x| SSMIS_SAFProg.out|l2l3_ssmi'`
The verification is expected to list no process of a processor nore the python process executing gv.py.		

Resume processing at that state:
`gvstartup`
`cat gv.status`		

The expected output is that the interrupted task is started again:
`312 created, 1 running, 268 backlog, 43 processed, 0 failed`
`r /home/gvop/inst/bin/beam-pconvert.sh`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 /home/gvop/eodata/MER_RR__1P/dim/2007/07/31`		

Interrupt processing again to use the same system status for the next test case.		
Enter disposition (passed/to be repeated/failed)		

GV-TC-100 Processing failure handling		
Steps	Notes	Rerun
Ensure that the initial status is the same as for GV-TC-90.		

Replace the input of the next step by a corrupted file:
`cd`
`mkdir eodata/backup2`
`mv \`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 \`
`eodata/backup2`
`touch \`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1`
`mv \`
`/home/gvop/eodata/MER_RR__1P/dim/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000* \`
`~/eodata/backup2/`
`ls -l /home/gvop/eodata/MER_RR__1P/r03/2007/07/31`
The size of the corrupted input file is expected to be 0 bytes.		

To start the processing system:
`cd inst`
`. mygv`
`gvstartup`
This command is expected to start the processing system and write out one summary status line and one line for a process being started:
`332 created, 1 running, 287 backlog, 44 processed, 0 failed`
`r /home/gvop/inst/bin/beam-pconvert.sh`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 /home/gvop/eodata/MER_RR__1P/dim/2007/07/31`		

Verify the status after some seconds:
`head -n3 gv.status`
The expected output is that the task with the corrupted input has failed:
`326 created, 0 running, 281 backlog, 44 processed, 1 failed`
`f /home/gvop/inst/bin/beam-pconvert.sh`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 /home/gvop/eodata/MER_RR__1P/dim/2007/07/31`
`...`		

Interrupt processing again to use the same system status for the next test case.		

Replace the input of the next step by the correct input file again:
`cd`
`mv -f \`
`eodata/backup2/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 \`
`/home/gvop/eodata/MER_RR__1P/r03/2007/07/31`
`ls -l /home/gvop/eodata/MER_RR__1P/r03/2007/07/31`
The size of the first input file is expected to be larger than 0 bytes.		

Resume processing at that state:
`cd inst`
`gvstartup`		

sleep 3 cat gv.status The expected output is that the interrupted task is started again: 312 created, 1 running, 268 backlog, 43 processed, 0 failed r /home/gvop/inst/bin/beam-pconvert.sh /home/gvop/eodata/MER_RR__1P/r03/2007/07/31/MER_RR__1PRACR20070731_010141_000026342060_00203_28313_0000.N1 /home/gvop/eodata/MER_RR__1P/dim/2007/07/31		
Enter disposition (passed/to be repeated/failed)		

5.5 Test Summary

Test result	
Steps	Notes
Number of tests (finally) passed:	
Number of tests failed:	
Enter disposition (passed, or updates required, tests to be repeated)	

6 Requirements Traceability

This chapter traces the system requirements from the design document [AD 4] to test cases defined in the previous chapter.

6.1 Verification Methods

Table 6-1 lists the verification methods for requirements (as requested by [RD 5], section 7.2.1.3). They are referred to in the requirements traceability below.

Table 6-1: Verification methods for system requirements

Code	Verification Method
D	Design, the requirement shall be fulfilled by features of the design
R	Review, the requirement shall be verified by a review
I	Inspection, the requirement shall be verified by inspecting documentation or configuration
T	Test, the requirement shall be verified by a test case
M	Monitoring, the requirement can be verified by observation during operations

Not all requirements can adequately be verified by a formal test. The main other method to be used is by inspection and by features of the design.

6.2 System Requirements Traceability

Table 6-2 lists the system requirements for the GlobVapour stand-alone processing system and traces them to test cases as far as it is planned to verify them by test.

Table 6-2: System requirements and their test cases

Req. ID	Requirement Title	Verif. Method	Test Case / Verif. Item
High level functional requirements			
GV-SR 10	The processing system shall generate GV products.	R	AR
GV-SR 20	The processing system shall integrate the GV processors.	R	AR
GV-SR 30	The processing system shall run the GV processing chains.	R	AR
GV-SR 40	The processing system shall manage the GV data.	R	AR
Functional requirements			
GV-SR 50	The processing system shall integrate the SSM/I processors.	T	GV-TC-10, GV-TC-20, GV-TC-60
GV-SR 60	The processing system shall integrate the MERIS processors.	T	GV-TC-30, GV-TC-40, GV-TC-50, GV-TC-60
GV-SR 70	The processing system shall integrate the GOME/SCIA processors.	T	GV-TC-70
GV-SR 80	The processing system shall integrate the (A)ATSR processors if they are available.	T	GV-TC-80

Req. ID	Requirement Title	Verif. Method	Test Case / Verif. Item
GV-SR 90	The processing system shall provide inputs to processors.	T	GV-TC-60, GV-TC-70, GV-TC-80
GV-SR 100	The processing system shall provide processing parameters to processors.	T	GV-TC-10, GV-TC-20
GV-SR 110	The processing system shall handle results from processors..	T	GV-TC-60, GV-TC-70, GV-TC-80
GV-SR 120	The processing system shall execute the SSM/I-MERIS chain.	T	GV-TC-60
GV-SR 130	The processing system shall execute the GOME-SCIA chain.	T	GV-TC-70
GV-SR 140	The processing system shall execute the ATSR chain if it is available.	T	GV-TC-80
GV-SR 150	The processing system shall execute the processing tasks according to their dependencies.	T	GV-TC-60
GV-SR 160	The processing system shall hold input product files in its file system archive.	I	file system
GV-SR 170	The processing system shall hold output product files in its file system archive.	I	file system
GV-SR 180	The processing system shall provide working directories to tasks.	I	file system
GV-SR 190	The processing system shall provide an operational procedure to ingest input products into its archive	T	GV-TC-110
GV-SR 200	The processing system shall provide an operational procedure to retrieve output products from its archive.	T	GV-TC-120
GV-SR 210	The processing system shall provide an operational procedure to cleanup parts of its archive.	T	GV-TC-130
Interface requirements			
GV-SR 220	The processing system shall not have dependencies to other systems on the same level (e.g. an external archive).	D	SDD
GV-SR 230	The processing system shall be designed for a computer with a Linux operating system.	D	SDD
Performance and sizing requirements			
GV-SR 240	The processing system shall have a storage capacity for one month of inputs, intermediates and outputs.	I	file system
GV-SR 250	The processing system design shall allow scaling to process 20 years of data in 3 months.	D	SDD
GV-SR 260	The processing system shall support concurrent processing and distributed processing on different hosts.	T (conc)/D(dist)	GV-TC-60 / SDD
Operational requirements			
GV-SR 270	The processing system shall process automatically a specified amount of input data through a processing chain.	T	GV-TC-60
GV-SR 280	The processing system shall handle processing failures.	T	GV-TC-100
GV-SR 290	The processing system shall allow to interrupt production.	T	GV-TC-90
GV-SR 300	The processing system shall allow to resume production after failure or interrupt, repeating the failed or interrupted task(s).	T	GV-TC-90, GV-TC-100