Horus
IMSETY
Software Transfer Document
Version 0.2 21st June 2007

Project team:  Jeroen Keiren  0569081
                Frank Koenders  0575629
                Thijs Nugteren  0574426
                Joeri de Ruiter  0578312
                Stijn Stiefelhagen  0579816
                Carst Tankink  0569954
                Pim Vullers  0575766
                Freek van Walderveen  0566348

Project manager:  Egbert Teeselink

Senior management:  L. Somers  TU/e (HG 7.83)
                    M. v.d. Brand  TU/e (HG 7.44)

Adviser:  R.J. Bril  TU/e (HG 5.09)

Customer:  E. v. Breukelen  ISIS

Computer Science, Eindhoven University of Technology, Eindhoven
Abstract

This document describes the status of the project at the transfer moment from the developers to the customer. This involves a list of delivered items and the development state of these items compared to the requirements from the URD [15] as tested during the acceptance test.
6.2.5 Request high resolution data ........................................ 13
6.2.6 Request deletion of data on the satellite .......................... 13

7 Software Change Requests ........................................... 14

8 Software Modification Reports ...................................... 15

A Stubs ........................................................................ 16
A.1 MCC ...................................................................... 16
A.1.1 Stub ................................................................... 16
A.1.2 Backends ............................................................... 16
A.1.3 Use ..................................................................... 16
A.2 MCS ...................................................................... 17
A.2.1 Stub ................................................................... 17
A.2.2 Backends ............................................................... 17
A.2.3 Use ..................................................................... 17
A.3 Installation ................................................................. 17
# Document status sheet

<table>
<thead>
<tr>
<th>Document title</th>
<th>Software Transfer Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document identifier</td>
<td>IMSETY/doc/STD</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Joeri de Ruiter, Pim Vullers</td>
</tr>
<tr>
<td>Version</td>
<td>0.2</td>
</tr>
<tr>
<td>Document status</td>
<td>Internally approved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 (Revision 1974)</td>
<td>20-06-2007</td>
<td>Joeri de Ruiter, Pim Vullers</td>
<td>Initial document, up for review.</td>
</tr>
<tr>
<td>0.2 (Revision 1994)</td>
<td>21-06-2007</td>
<td>Joeri de Ruiter, Pim Vullers</td>
<td>Fixed review remarks.</td>
</tr>
</tbody>
</table>
## Document change report

<table>
<thead>
<tr>
<th>Document title</th>
<th>Software Transfer Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document identifier</td>
<td>IMSETY/doc/STD</td>
</tr>
<tr>
<td>Date of changes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section number</th>
<th>Reason for change</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Fixed review remarks.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Purpose

This document gives all information concerning the transfer of the product from the developers to the customer. It describes the items to be transferred, how these have been tested prior to the actual transfer. And finally the state of these items compared to the requirements from the URD [15].

1.2 Scope

IMSETY will provide scientists and observers with an easy to use program interface to conduct experiments on space based samples and possible reference samples on Earth.

1.3 List of definitions

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Architectural Design Document</td>
</tr>
<tr>
<td>AT</td>
<td>Acceptance Test</td>
</tr>
<tr>
<td>ATP</td>
<td>Acceptance Test Plan</td>
</tr>
<tr>
<td>DDD</td>
<td>Detailed Design Document</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>GENSO</td>
<td>Global Educational Network for Satellite Operations</td>
</tr>
<tr>
<td>IPD</td>
<td>Interface and Protocol Definition</td>
</tr>
<tr>
<td>ITP</td>
<td>Integration Test Plan</td>
</tr>
<tr>
<td>MCC</td>
<td>Mission Control Client, see the URD [15] for more information.</td>
</tr>
<tr>
<td>MCS</td>
<td>Mission Control Software, see the URD [15] for more information.</td>
</tr>
<tr>
<td>SCMP</td>
<td>Software Configuration Management Plan</td>
</tr>
<tr>
<td>SPMP</td>
<td>Software Project Management Plan</td>
</tr>
<tr>
<td>SQAP</td>
<td>Software Quality Assurance Plan</td>
</tr>
<tr>
<td>SRD</td>
<td>Software Requirements Document</td>
</tr>
<tr>
<td>STD</td>
<td>Software Transfer Document</td>
</tr>
<tr>
<td>STP</td>
<td>System Test Plan</td>
</tr>
<tr>
<td>SUM</td>
<td>Software User Manual</td>
</tr>
<tr>
<td>SVVP</td>
<td>Software Verification and Validation Plan</td>
</tr>
<tr>
<td>URD</td>
<td>User Requirements document</td>
</tr>
<tr>
<td>UTP</td>
<td>Unit Test Plan</td>
</tr>
</tbody>
</table>
1.4 List of references

Chapter 2

Build procedure

2.1 Client

The IMSETY client was built on Stijn Stiefelhagen’s laptop running Windows, which already contained a Qt development installation. This was needed as the customer did not have a system available with this environment. This took place at the ISIS office in Delft. The IMSETY client version 0.2 was built according to the instructions from the DDD [3] section 4.3. No errors occurred during this procedure.

2.2 Server

The IMSETY server was built on isis-1.starters.tudelft.nl running Linux, which already had the dependencies installed. This took place using SSH access with the ‘horus’ account. This machine has two gcc versions installed, from which only one was working correctly. The malfunctioning version was selected automatically as /usr/local/bin occurred in the PATH environment variable before /usr/bin which contains the correct functioning gcc install. To overcome this problem we removed the /usr/local/bin directory from the PATH environment variable.

After this problem was solved the IMSETY server version 0.2 was built according to the instructions from the DDD [3] section 4.2. No errors occurred during this procedure.
Chapter 3

Installation procedure

3.1 Client

The client was installed to a Windows machine of ISIS. This was done by copying the following files from the build system:

- client.exe
- mingwm10.dll
- QtCore4.dll
- QtGui4.dll
- QtNetwork4.dll
- QtXml4.dll

This confirms to the procedure as described in the DDD [3] section 4.3.

3.2 Server

The IMSETY server was used directly from the source directory. Before the server was started the configuration file server.icf was edited to reflect the configuration from table 3.1 and the at-data.sql file was loaded into the database to initialize the MySQL database.

This confirms to the procedure as described in the DDD [3] section 4.3. The only step that has been altered is that at-data.sql was loaded instead of imsety.sql, as this file contained some predefined experiments.

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>imsety</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>localhost</td>
</tr>
<tr>
<td>USER</td>
<td>horus</td>
</tr>
<tr>
<td>PASSWORD</td>
<td>jahooreens</td>
</tr>
<tr>
<td>RETURNADDR</td>
<td>localhost</td>
</tr>
</tbody>
</table>

Table 3.1: Server configuration
Chapter 4

Configuration item list

This chapter gives an overview of the configuration items transferred to the customer. All product documentation and project plans are delivered in PDF format as well as \TeX{} format, other process documentation is only available in \TeX{} format. The tags/doc directory contains all versions of the documentation in PDF format whereas the trunk/doc directory contains the latest version of all documents in \TeX{} format. All software products are delivered as source files. These can be found in the trunk/src directory.

4.1 Product items

4.1.1 Requirement and design documents

- URD [15]
- SRD [9]
- IPD [5]
- ADD [2]
- DDD [3]

4.1.2 Test plans

- UTP [14]
- ITP [4]
- STP [13]
- ATP [1]

4.1.3 Other documentation

- SUM [11]
- STD [10]
CHAPTER 4. CONFIGURATION ITEM LIST

4.1.4 Software products

- IMSETY server
- IMSETY client
- MCC stub
- MCS stub

4.2 Process items

4.2.1 Project plans

- SCMP [6]
- SPMP [7]
- SQAP [8]
- SVVP [12]

4.2.2 Other documentation

- Agendas
- Minutes
- Progress reports
- Review reports
- Presentations
Chapter 5

Acceptance test report summary

The system has been subject to two acceptance tests. The statistics from these tests can be found in section 5.1 and 5.2. During these test stubs have been used to simulate the MCC and MCS behavior. Documentation concerning these stubs can be found in appendix A.

5.1 Monday June 11th 2007

<table>
<thead>
<tr>
<th>Test report identifier</th>
<th>ATRPT1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>11-06-2007</td>
</tr>
<tr>
<td>Time</td>
<td>10:00 - 16:00</td>
</tr>
<tr>
<td>Location</td>
<td>Eindhoven, TU/e Main building, room 10.38</td>
</tr>
<tr>
<td>Participants</td>
<td>Eddie van Breukelen (customer), Thijs Nugteren (Horus), Pim Vullers (Horus) and Freek van Walderveen (Horus).</td>
</tr>
<tr>
<td>Test document</td>
<td>ATP 0.4</td>
</tr>
</tbody>
</table>

Description: This report concerns the first AT of IMSETY. Subject to this test are the IMSETY server version 0.1 and the IMSETY client version 0.1. These subjects have been tested against the tests described in ATP chapter 3.

Results: 16 tests passed and 5 tests failed. Failures concern user rights and the scheduling of experiments. Extra tests must be written w.r.t unscheduling experiments and getting an overview of logs and of observation data (thumbnails).
## 5.2 Friday June 15th 2007

<table>
<thead>
<tr>
<th>Test report identifier</th>
<th>ATRPT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>15-06-2007</td>
</tr>
<tr>
<td>Time</td>
<td>14:00 - 16:00</td>
</tr>
<tr>
<td>Location</td>
<td>Delft, ISIS Office</td>
</tr>
<tr>
<td>Participants</td>
<td>Eddie van Breukelen (customer), Joeri de Ruiter (Horus), Stijn Stiefelhagen (Horus).</td>
</tr>
<tr>
<td>Test document</td>
<td>ATP 0.5</td>
</tr>
<tr>
<td>Description</td>
<td>This report concerns the second AT of IMSETY. Subject to this test are the IMSETY server version 0.2 and the IMSETY client version 0.2. These subjects have been tested against: All testcases described in ATP chapter 3 except the ones concerning administration, these have been tested successfully and approved by the customer during the previous acceptance test.</td>
</tr>
<tr>
<td>Results</td>
<td>All tests passed. Customer has approved our software.</td>
</tr>
</tbody>
</table>
Chapter 6

Software Problem Reports

The IMSETY software package does not meet all requirements stated in the URD [15]. The following sections give an overview of the requirements which have not been implemented. For each group of requirements the available functionality is given as well as the work to be done to meet the requirements. More information on how and where to solve the missing functionality can be found in the DDD [3] section 3.1.1 and 3.2.1.

6.1 Client non conformance

6.1.1 Logs and user notification

<table>
<thead>
<tr>
<th>Requirement(s):</th>
<th>URCAR10, URCAR61, URCAR62, URCAR63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality:</td>
<td>Showing logs in the user interface according to user rights, notifying the user about important messages.</td>
</tr>
<tr>
<td>Available:</td>
<td>• Logs are created by all server components.</td>
</tr>
<tr>
<td></td>
<td>• Logs are available for administrators through the administration interface.</td>
</tr>
<tr>
<td>Todo:</td>
<td>• Implement log retrieval in the server connectivity.</td>
</tr>
<tr>
<td></td>
<td>• Implement log viewers in the user interface.</td>
</tr>
<tr>
<td></td>
<td>• Implement user notification for important messages.</td>
</tr>
</tbody>
</table>
6.1.2 Payload locking

Requirement(s): URCAR19, URCAR38

Functionality: Locking of payloads to prevent multiple access to payloads and experiments.

Available:
- Nothing.

Todo:
- Implement lock request whenever a user wants to modify a payload or experiment.
- Implement lock release whenever a user finishes payload or experiment modification.

6.1.3 Plug-in system

Requirement(s): URCAR29, URCAR45, URCAR46, URCAR52, URCAR53, URCAR54

Functionality: Plug-in systems for data representations and pre- and postprocessing of data representations. As well as example plug-ins for these systems.

Available:
- Used the Qt framework for easy modification of the client, such that it can be easily adapted to support other data representations.

Todo:
- Implement the plug-in systems using the Qt plug-in functionality.
- Adapt the user interface to facilitate plug-in management.

6.1.4 Experiment intervention

Requirement(s): URCAR37

Functionality: Sending commands to an already executing command.

Available:
- GUI component for intervention in the observation window.

Todo:
- Implement sending commands to the server.
- Implement receiving feedback about the sent command (intervention log).
### 6.1.5 Marking of interesting parts

Requirement(s): URCAR50

Functionality: Marking of interesting parts during real time observation.

Available:
- GUI component for marking periods during real time observation.

Todo:
- Implement storage of this marking on the local filesystem.

### 6.1.6 Request high resolution data

Requirement(s): URCAR51, URCAR56

Functionality: Request the download of high resolution data.

Available:
- GUI component for requesting download of high resolution data.

Todo:
- Implement sending the request to the server.

### 6.1.7 Request deletion of data on the satellite

Requirement(s): URCAR57

Functionality: Request the deletion of high resolution data from the satellite.

Available:
- GUI component for requesting deletion of data from the satellite.
- Marking for deletion is available through the administration interface.

Todo:
- Implement sending the request to the server.
- Implement receiving feedback about the deletion from the server.

### 6.2 Server non conformance

#### 6.2.1 Logs and user notification

Requirement(s): URCAR10, URCAR61, URCAR62, URCAR63

Functionality: Sending the logs to the client.

Available:
- Logs are made by all server components.

Todo:
- Implement sending logs to the client.
CHAPTER 6. SOFTWARE PROBLEM REPORTS

6.2.2 Payload locking

Requirement(s): URCAR19, URCAR38

Functionality: Locking of payloads to prevent multiple access to payloads and experiments.

Available:

• Nothing.

Todo:

• Implement receiving lock request and release from the client.

• Implement lock management in the controller.

• Implement sending responses to the client.

6.2.3 Experiment modification after upload

Requirement(s): URCAR35, URCAR36

Functionality: Modify an experiment before it is executed.

Available:

• Modified experiments can be received from the client.

• Experiments can be sent to the satellite.

Todo:

• Implement notification if another upload is not possible before the scheduled execution.

• Implement replacing the original experiment.

6.2.4 Experiment intervention

Requirement(s): URCAR37

Functionality: Sending commands to an already executing command.

Available:

• Communication with the satellite in the MCS controller.

Todo:

• Implement receiving commands from the client.

• Implement forwarding commands from the client to the MCS controller.

• Implement feedback from the MCS controller to the client.
### 6.2.5 Request high resolution data

**Requirement(s):** URCAR40, URCAR51, URCAR56  
**Functionality:** Request the download of high resolution data.  
**Available:**  
- Communication with the satellite in the MCS controller.  
**Todo:**  
- Implement receiving a download request from the client.  
- Implement the request to the satellite.

### 6.2.6 Request deletion of data on the satellite

**Requirement(s):** URCAR57  
**Functionality:** Request the deletion of high resolution data from the satellite.  
**Available:**  
- Communication with the satellite in the MCS controller.  
**Todo:**  
- Implement receiving a deletion request from the client.  
- Implement the request to the satellite.
Chapter 7

Software Change Requests

Not applicable.
Chapter 8

Software Modification Reports

Not applicable.
Appendix A

Stubs

This chapter describes the stubs built to simulate the behaviour of the MCC and the MCS, since both are not yet available to us. Both stubs have a manual backend as well as an automatic backend. The stubs are written in the programming language PHP. Apache is used to host the stubs so we can use the built-in SSL and authentication options. All requests are logged in a MySQL database.

A.1 MCC

A.1.1 Stub

The MCC stub implements the methods specified in the IPD [5]. When a request is received by the MCC stub, the request is saved in a database and the identifier is returned. Then, if it is enabled, the automatic backend is called.

A.1.2 Backends

Automatic

The automatic backend is called by the MCC stub. The backend will wait for a few seconds and then an acknowledgement will be sent back. When a list of opportunities is requested a random list of opportunities is created. There is also an automatic backend that always sends the same reply. This backend can be used for debugging.

Manual

The manual backend has to be operated by a user. The user will be presented a list with received messages and whether a message has already been answered. It is then possible to answer a particular message. The user will then get a form which he can use to answer the message.

A.1.3 Use

The MCC stub can be used by calling the file\texttt{mcc.php} on the server. By passing arguments via the URL a different backend can be chosen. By default the normal automatic backend is selected. The manual backend can be selected by calling\texttt{mcc.php?manual=1}. The predictable automatic backend is selected by calling\texttt{mcc.php?test=ok}. The stubs have to be used as the URL of the MCC. If you want to operate the manual backend you can go to\texttt{mcc.backend.man.php}.
APPENDIX A. STUBS

A.2 MCS

A.2.1 Stub

The MCS stub implements the methods specified in the IPD [5]. When a request is received by the MCS stub, the request is saved in a database and the identifier is returned. Then, if it is enabled, the automatic backend is called. The MCS stubs also contain a streaming stub that can be used for observations.

A.2.2 Backends

Automatic

The automatic backend is called by the MCS stub. The backend will wait for a few seconds and then an acknowledgement will be sent back. There is also a part that can be used to send a stream of ‘received’ data for a given time. This received data consists of webcam images.

Manual

The manual backend has to be called by the user. The user will be presented a list with received messages and if a message has already been answered. It is then possible to answer a particular message. The user will then see a form which he can use to answer the message. When answering to a fetch request it is possible to upload a file that has to be send, but it is also possible to send a webcam image.

Streaming

The streaming stub sends new data to a given host for a given payload and time. The stub sends the images from a webcam.

A.2.3 Use

The stub can be used by calling the file mcs.php on the server. By passing arguments via the URL a different backend can be chosen. By default the normal automatic backend is selected. The manual backend can be selected by calling mcs.php?manual=1. These stubs can be used by using them as the URL of the MCS. If you want to you operate the manual backend you can go to mcs_backend_man.php.

The streaming stub can be used by calling mcs_backend_stream.php. Via the URL the complete URL of the MCS listener (e.g. http://localhost:10001/RPC2), the execution time in seconds and the payload identifier have to be passed, so you have to call the stub using:

mcs_backend_stream.php?time={time}&address={address}&payload={payload id}

This stub has to be called via the webbrowser to start sending data to the given address.

A.3 Installation

The stubs can be installed by copying the stubs folder to a folder that is reachable by your webserver. You then add the structure of the database you can find in stubs.sql. The database host, username, password and database have to be filled in in the file connect.php. You can now use the stubs.