Fully Functional Device Controller

Written by:
PROMISE WP R6 Members

DELIVERABLE NO	DR6.4: Test & Evaluation Document
DATE	20. March 2006
WORK PACKAGE NO	WP 6: Fully Functional Device Controller
VERSION NO.	0.1
ELECTRONIC FILE CODE	dr6.4.test & evaluation document.doc
CONTRACT NO	507100 PROMISE A Project of the 6th Framework Programme Information Society Technologies (IST)
ABSTRACT:	

STATUS OF DELIVERABLE		
ACTION	**BY**	**DATE (dd.mm.yyyy)**
SUBMITTED (author(s))	Celal Dikici	02.05.2006
VU (WP Leader)	Gregor Hackenbroich	15.05.2006
APPROVED (QIM)	D. Kiritsis	15.05.2006

Revision History

Date (dd.mm.yyyy)	Version	Author	Comments
10.01.2006	0.1	Celal Dikici	Document setup and initial outline structure
15.02.2006	0.2	Celal Dikici	Put some content
02.05.2006	0.3	Celal Dikici	Finalized Version for Review (QIM)
04.05.2006	0.4	David Potter	Expand Scope and Objectives, sharpen Introduction and initiate Overall Summary. Add table of Abbreviations.
04.05.2006	0.5	Kary Främling	Finalised HUT part
08.05.2006	0.6	Jürgen Anke	Finalized SAP part
09.05.2006	0.7	Bjorn Forss	Finalized Stockway part
12.05.2006	0.9	Celal Dikici	Finalized DR6.4 (send to partners for latest comments)
12.05.2006	0.91	Kary Främling	Made HUT figure captions shorter according to review recommendation
15.05.2006	1.0	Celal Dikici	Filnalized & ready for approval by QIM

Author(s)' contact information

Name	Organisation	E-mail	Tel	Fax
Gregor Hackenbroich	SAP Research	gregor.hackenbroich@sap.com	+49 351 4457-2311	+49 6227 78-43474
Jürgen Anke	SAP Research	juergen.anke@sap.com	+49 351 4457-2304	+49 6227 78-44661
Celal Dikici	BIBA	dik@biba.uni-bremen.de	+49 421 218-5582	+49 421 218-5610
David Potter	INDYON	david.potter@indyon.de	+44 23 9234 5152	+44 23 9259 2327
Kary Främling	HUT	Kary.Framling@hut.fi		
Bjorn Forss	Stockway	bjorn.forss@stockway.com		

Table of Contents

Abbreviations:

DC	Device Controller component of the PROMISE Middleware
DHL	Device Handling Layer
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IOCI	Inter-organizational Communication Infrastructure
JVM	Java Virtual Machine
PEID	Product Embedded Information Device
P2P	Peer to Peer
RHL	Request Handling Layer
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
TTL	Time to Live
UPnP	Universal Plug and Play (UPnP™)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WP	PROMISE Project Work Package
WWAI	World Wide Article Information protocol

Abstract

In this deliverable we present the current state of our testing & evaluation activities for the PROMISE middleware. We present a comprehensive testing strategy as theoretical foundation, followed by the definition of test cases and their results. Identified problems will be subsequently addressed to improve the quality of our implementation.

1 Scope and Objectives of this Document

The purpose of this document is to describe the approach used for testing the results of tasks TR6.2 and TR6.3 which are described in the deliverables DR6.2 and DR6.3. This document is the main deliverable of task TR6.4.

The system test intends to prove that the functionality developed by the technical partners is as specified. This will be achieved by defining, reviewing and executing test cases. The results will be compared against pre-specified expected results to evaluate if the test was successful.

There are a number of differences between what has been specified in the task definitions of TR6.2 through TR6.4 and what has been developed and tested thus far. It is important to explain this situation and how these gaps will be addressed in future tasks in WP R6.

One, there is still work in progress to define the architecture for plug-in modules (buffers, aggregators and filters) foreseen in TR6.3; we propose to complete this by M24 in conjunction with task TR6.5. Consequently there is no implementation of these plug-in modules available for testing at this time.

Two, task TR6.4 envisaged “Deployment and test in demonstrators”. This has not been possible since to date there has been no actual demonstrator development. Therefore this test objective must be deferred until later in line with Application Cluster plans for demonstrator development.

Three, task TR6.4 also envisaged “Performance evaluation and Usability study”. Although the tests reported in this document contain an element of performance testing, it has not been feasible to carry out any usability tests at this stage.

On the other hand, the reported tests do include an element of testing of the emerging IOCI middleware component which redresses the balance to some extent.

The tests described in this deliverable represent Unit Testing. At present there is no defined plan for Integration Testing, but this can be addressed in the future plans for overall testing.

The gaps in architecture, implementation and testing that have been described above will be addressed together with the on-going effort planned for tasks TR6.5 through TR6.7. Since the latter task definitions do not currently include any provision for testing, here will be an opportunity to consolidate the plan.

In summary, the testing reported in this deliverable provides a sound basis for the unit testing and eventually integration testing of the PROMISE Middleware.

2 Introduction

It is well known that in any typical software project nearly 50% of time and over 50% of costs are consumed by testing. It is tempting to assume that because testing is a common requirement that an easy to use template exists that needs only to be filled with the test results. In reality testing is always a challenge. There is no one-fits-all solution; every software project needs a testing process with its own specific test definitions.

Even though the testing can be discussed and considered from various technical points of view, the view from economic and human psychology criteria greater impact. Sometimes the capability and the attitude towards testing has a greater impact on the success of the testing than simple technical points.

The first step is to understand what is meant by the term “testing”. This can be interpreted like “testing is a process that shall show that there are no errors” or “testing is a process that increases the trust in the software” but these definitions have a negative feel to them.

Let us consider that testing, which is a time and cost consuming process, should increase the value of the software, where “value” means quality and/or reliability. Increasing the value involves finding and eliminating errors. Therefore an application should not be tested to show that it works correctly, but we should begin with the assumption that the implementation contains errors (which is a valid assumption for the majority of all software).

A more appropriate definition is “*testing is a process to execute an application to find errors*”. Assuming the opposite may lead to a selection of test criteria which reduce the probability of finding errors.

The current situation in WP R6 is that technology Partners SAP and HUT have begun to develop their solution based on the specification. A PROMISE total system does not yet exist so tests investigating the interdependencies and interaction with other parts of PROMISE solution can not be made at this stage.

Taking both the points in the introduction and the current situation in WP R6 into account, the best method for testing the implementation is the method called Black-Box testing. Black-Box testing is also called data driven testing or Input/Output-testing. The tester treats the application as a Black box which means, the tester is not interested in the internal behaviour or structure of the program. His objective is to find the circumstances where the application works against the interface specification and where it does not. The test values are derived from the specification without knowledge of the internal implementation.

To find all errors with this method it is necessary that a complete input test must be executed. Complete means to give infinite test case values with possible and not possible values. (e.g. specification: integer given value real). This shows clearly that a complete input test is not possible. That results in two conclusions:

1. A program cannot be tested in a way where the freedom from error cannot be guaranteed
2. Fundamental point of view is the economics of testing

Therefore the intention should be to maximize the effect of testing. This requires analysing the implementation and suggesting reasonable assumptions about the programming. This approach is taken from White Box testing. In White Box testing an investigation on the intended structure of the program will be made and the tester defines test data with knowledge of the implementation in mind.

Another important point is that a complete PROMISE system does not yet exist (as mentioned above). This means that testing the interrelations and interactions with other PROMISE components is not yet possible. Where a test of the whole System/Application is not possible, each component can be tested. This is the so called Module test (or Unit testing).

It can be assumed that in WP R6 the testing will be executed in a way that

1. the modules are taken as standalone,
2. the specification will be used for creating test cases, while
3. keeping in mind the internal structure of the implementation for boundaries for test cases.

3 Test Planning

The test plan prescribes the scope, approach, resources, and schedule of the testing activities. It identifies the items to be tested, the features to be tested, the testing tasks to be performed and the personnel responsible for each task.
Previous deliverables in WP R6 are used as a base. The "*IEEE Std. 829-1998 IEEE Standard for Software Test Documentation*" as additional source is also considered.

Approach

The proposed common approach for system testing process is as follows:

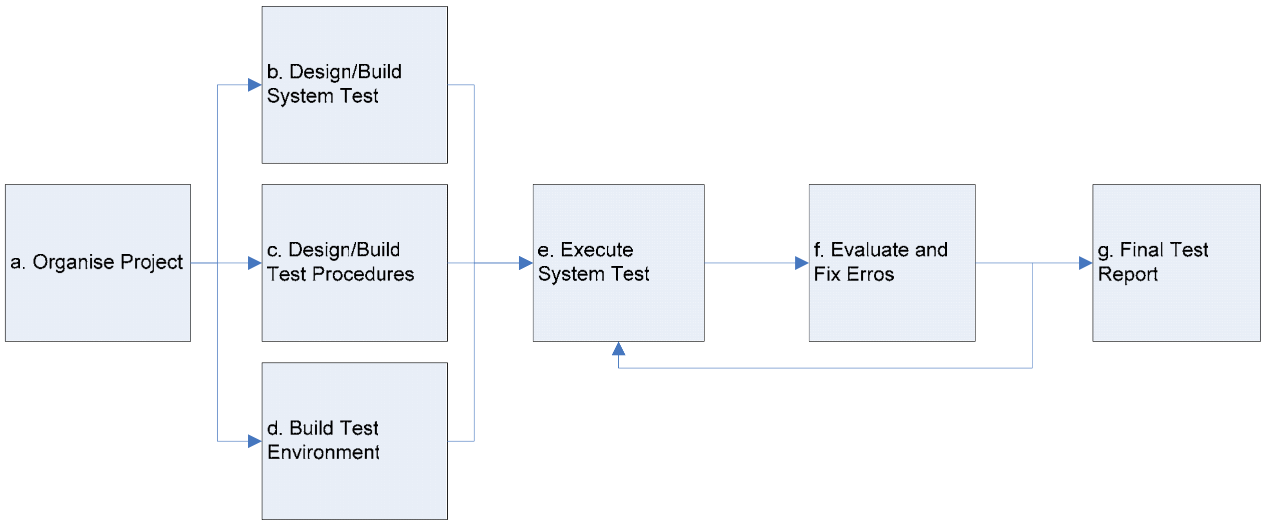

a. Organise Project
b. Design/Build System Test
c. Design/Build Test Procedures
d. Build Test Environment
e. Execute System Test
f. Evaluate and Fix Erros
g. Final Test Report

Figure 1: Testing process[1]

- **a.** **Organise Project** involves creating a System Test Plan, Schedule & Test Approach, and requesting/assigning resources.

- **b.** **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team. The Test Team will then identify Test Cases and the Data required. The Test data are derived from the specification documents DR6.2 and DR6.3.

  The main test types can be summarised as follows. All system test plans and conditions will be developed from the functional specification and the requirements definition.

- **c.** **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

- **d.** **Build Test Environment** includes requesting/building hardware, software and data set-ups.

[1] Adapted from Sample Software system test plan for a new application: http://members.tripod.com/~bazman/

e. **Execute System Test**
The main thrust of the approach is to intensively test the modules thus raising approximately 80% of errors in this period. When all errors (which potentially impact overall processing) are fixed, an additional set of test cases can processed to ensure the system works in an integrated manner. This will be the final test of the system as a single application which will not be handled in R6. An integration test will be required at some stage in the whole project.

f. **Evaluate and Fix Errors**

g. **Final Test Report** summarizes the overall results achieved within these testing tasks.

4 System Test plan for PROMISE R6

This section is based on the former section, in which a generic Testing and Test Types are presented. The goal of this section is to tailor the generic description and make it suitable for the test work package of PROMISE project.

In the previous deliverable DR 6.3 we mentioned a possible testing process as seen in Figure 1 again but revised.

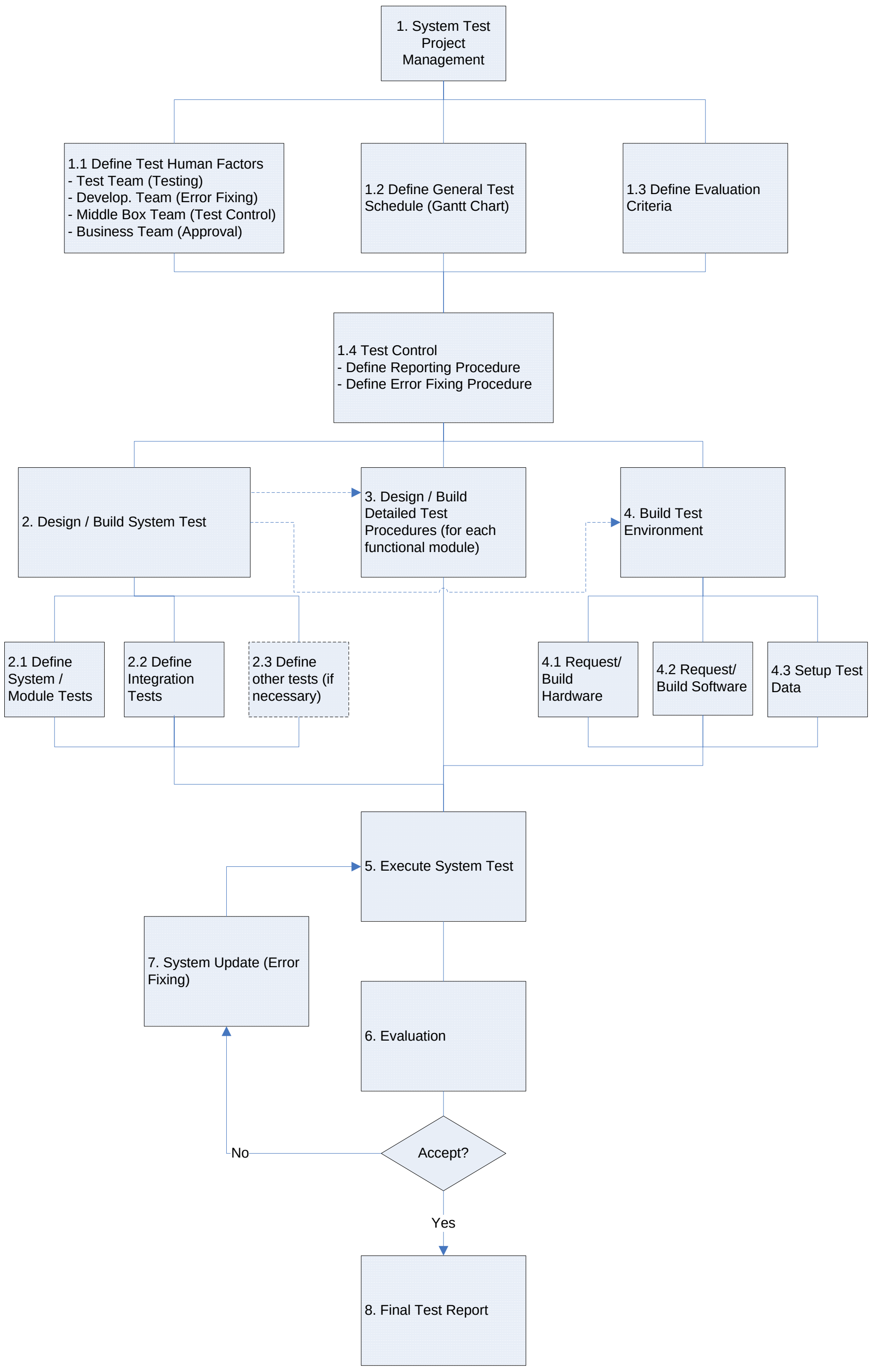

1. System Test Project Management
1.1 Define Test Human Factors
- Test Team (Testing)
- Develop. Team (Error Fixing)
- Middle Box Team (Test Control)
- Business Team (Approval)
1.2 Define General Test Schedule (Gantt Chart)
1.3 Define Evaluation Criteria
1.4 Test Control
- Define Reporting Procedure
- Define Error Fixing Procedure
2. Design / Build System Test
3. Design / Build Detailed Test Procedures (for each functional module)
4. Build Test Environment
2.1 Define System / Module Tests
2.2 Define Integration Tests
2.3 Define other tests (if necessary)
4.1 Request/ Build Hardware
4.2 Request/ Build Software
4.3 Setup Test Data
5. Execute System Test
7. System Update (Error Fixing)
6. Evaluation
Accept?
No
Yes
8. Final Test Report

Figure 2: Testing Process (modified from DR 6.3)

4.1 System Test Project Management

4.1.1 Define Test Human Factors

- Test Team - responsible for testing the system modules according to the predefined tabular *Test Forms* (see Appendix A) and filling the test forms.
  (SAP, HUT, Stockway)
- Development Team - responsible for fixing errors and updating system (and then delivering the next release system for further test).
  (SAP, HUT, Stockway)
- Middle Box Team – made up by Test Team Leaders and Development Team Leaders, responsible for co-ordinating the work between Test Team and Development Team. Their main tasks include:
  - Evaluate test results based on the Test Forms completed by Test Team;
  - Analyse, categorise and prioritise the errors (bugs);
  - Make *mid-term Test Report*;
  - Evaluate bug fix results based on the Test Forms
  - Make *mid-term Bug fix Report*;
  - Control and monitor the work of both Test Team and Development Team.

  (BIBA, Indyon)
- Business Team – test and evaluate the system from their business view, and finally approve.
  (BIBA, Indyon)

4.1.2 Define General Test Schedule

This section contains a table manage the whole user test process, reflecting all of the system function modules which are defined in Design Specification Documents (DR 6.2 & DR 6.3).

Task	**Due date: April 2006,**
Build Test Environment	11.
Define test cases	12.
Generating test data	20.
Executing tests	27.
Review / Evaluation	Concurrently 13. – 27.
Error Fixing	Concurrently, 13. – 28.
Final Report	26. – 30.

4.1.3 Define Evaluation Criteria

The tests are approved when the passed the testing criteria, namely if the real result is the same as expected.

4.2 Design / Build System Test

This step involves identifying tabular Test Forms for each functional module, which include Test Cases, Entrance & Exit Criteria, Expected Results, etc. Required Test Data should also be defined in this phase. (see Appendix A).

4.2.1 Define System / Module Tests

This is the process of testing individual code modules before they are integrated with other modules. The goal of module testing is to identify and fix as many errors as possible before modules are combined into larger software units such as programs, classes, and subsystems.

4.2.2 Define Integration Tests

This test proves that all areas of the system interface with each other correctly and that there are no gaps in the data flow. Final Integration Test proves that system works as integrated unit when all the fixes are complete. These tests can not be handled within the actual phase of the PROMISE project due a missing of a whole PROMISE software product. Therefore after all PROMISE modules of all work packages are implemented a new task for overall PROMISE Testing has to be performed.

4.3 Design / Build Detailed Test Procedures

Design/build detailed test procedures need to be carried out for each functional module. It is based on the test forms defined in the Appendix A – Test case form00 below reflecting the test procedures for each functional module.

4.4 Build Test Environment

The test environment for each Functional Module consists of identifying necessary hardware, software and test data to be used for executing tests, i.e. applying the test forms as described above.

4.4.1 Hardware

The hardware needs to be tailored to meet the requirements of the software systems applied.

Server Hardware
The PC to be used as server should fulfil the requirements listed in the specific section.

Client Hardware
The PC to be used as client should fulfil the requirements listed in the specific section.

4.4.2 Software

The software necessary to run the PROMISE Modules is given here.

4.4.3 Test Data

The test data to be used can be derived from the specification.

4.5 Execute System Test

Based on the outputs of Sections above the tests will be executed. The corresponding Test Forms are to be completed.

4.6 Evaluation

According to the output of Section 4.5 – completed *Test Forms*, activate the Reporting Procedure and Error Fixing Procedure. Figure 3 shows a general approach on testing and evaluation.

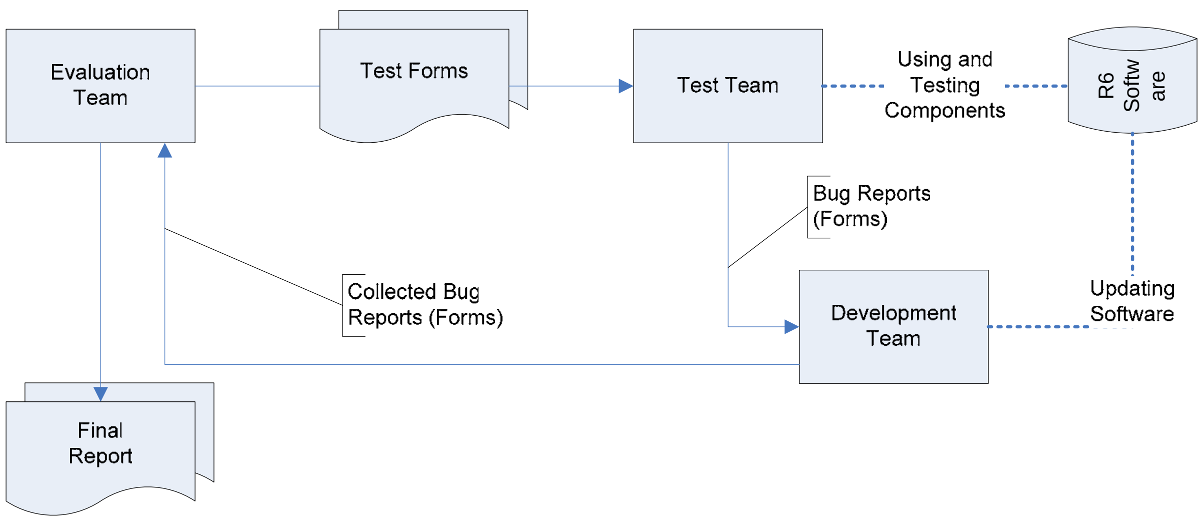


Figure 3: General approach of testing and evaluation

4.7 System Update / Error Fixing

The reported errors are communicated to the developer team. The developer team has the responsibility to fix the reported error and inform the evaluation team. After fixing the problem the components affected by the problem are to be re-tested.

4.8 Final Test Report

The final test report will be initiated when all the test are finished and accepted by the approval team. Synchronous this will normally flow into the pilot release of the software, which can now be deployed to a pilot system. In PROMISE a deployment is not planned yet. The implemented modules will be used in the demonstrators developed in the *Application* clusters (A1 – A11). This deliverable will then be reconsidered for additional tests and evaluation of the whole PROMISE solution.

5 Test execution for each testing Partner

5.1 Test cases for Modules of SAP

The aim of this section is to verify the correct implementation of the WP R6 modules at SAP based on the following test cases. The test cases are based on the specification made in DR 6.2 and DR 6.3.

5.1.1 Guidelines and comments for testing at SAP.

System / Application to be tested	PROMISE R6 Device Controller Implementation of SAP.
System version	N/A
Test risks	Changes on implementation or test system during the test phase
Pre-conditions	Internal pre-test are made
Components and functions / units which are not tested	Service Repository
Components and functions / units which are tested	Existing implementation at the test date installed on internal systems at SAP, specifically the Request Handling Layer, and the Device Handling Layer.
Test data (Test values)	Test data is derived from the specification
Testing user	
Remarks	Without errors, the test case will marked OK With errors, the testcase marked with NOK (also in the section title to see easily with one view which tests are passed or not.

5.1.2 Test environment at SAP

Hardware
The hardware needs to be tailored to meet the requirements of the software systems applied.

Server Hardware
The PC to be used as server should fulfill the following requirements:

RHL
Pentium 4, 3.2GHz, 2GB HD, 2GB RAM

DHL
40MB HD, 64MB RAM

Software
The software necessary to run the PROMISE Modules is given by the following table:

RHL

Operating System	Windows X (any Windows version)
Server	SAP Web AS (J2EE 1.3 compliant application server)

DHL

Operating System	Windows X (any Windows version)
Service Platform	Oscar (OSGi Container)
DB	Derby

5.1.3 Functional Test cases for modules of SAP

5.1.3.1 Testcase001_SAP_CONTENT_I OK

Positive test / Module test			
Core function: READ			
Pre-requisites: none			
	Test ID		**4.1**
	### Test activity ###		
	1.1	Read infoItemY on deviceX where deviceX is connected and infoItemY exists on deviceX	
	1.2	Read infoItemY on deviceX where deviceX is connected and infoItemY **doesn't** exist on deviceX (i.e. not checked against metadata)	
	1.3	Read infoItemY on deviceX where deviceX is not connected	
	### Expected result ###		
	1.1	result String with current value of infoItemY on deviceX	
	1.2	result String with error message that infoItemY does not exist on deviceX	
	1.3	result String with requestId to retrieve result later	
	### Realised Result (Textual) ###		
	1.1	result String with current value of infoItemY on deviceX	
	1.2	result String with error message that Read can't be invoked on infoItemY on deviceX	
	1.3	result String with requestId to retrieve result later	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: 4: Erroneous/imprecise error message
	1.3	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		4 (not critical)
	Remarks		For 1.2 test requests against metadata information (i.e. correct targeted/infoItemId pairs) and only accept valid requests
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		12.04.2006

5.1.3.2 *Testcase002_SAP_CONTENT_II OK / NOK*

Positive test / Module test			
Core function: WRITE			
Pre-requisites: none			
	Test ID		**4.2**
	### Test activity ###		
	1.1	write infoItemY on deviceX where deviceX is connected and infoItemY exists on deviceX	
	1.2	write infoItemY on deviceX where deviceX is connected and infoItemY doesn't exist on deviceX	
	1.3	write infoItemY on deviceX where deviceX is not connected	
	### Expected result ###		
	1.1	result String confirming new value of infoItemY on deviceX	
	1.2	result String with error message that infoItemY does not exist on deviceX	
	1.3	result String with requestId to retrieve result later	
	### Realised Result (Textual) ###		
	1.1	result String confirming new value of infoItemY on deviceX	
	1.2	new infoItem key is added on device, and value is written	
	1.3	result String with requestId to retrieve result later	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: 2-3 new infoItemId created on write
	1.3	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		NOK
	Error Class		3 (strong)
	Remarks		TBD in DHL: Return error message if infoItem is not available on device
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		12.04.2006

5.1.3.3 Testcase003_SAP_CONTENT_III OK

Positive test / Module test		
Core function: PROCESS REQUESTS		
Pre-requisites: devi ceX not connected, requests are buffered for deviceX		
Test ID	**4.3**	
### Test activity ###		
1.1	Connect deviceX	
1.2	RHL: retrieve(requestId)	
1.3		
### Expected result ###		
1.1	Buffered requests for deviceX are sent to DHL and processed	
1.2	Resultstring for request with requestId	
1.3		
### Realised Result (Textual) ###		
1.1	Requests are processed at DHL, results are sent to RHL	
1.2	Resultstring for request with requestId	
1.3		
### Realised Result ###		
	ok?	**Error description**
1.1	OK	
1.2	OK	
1.3		
### Overall assessment of the Test ###		
Overall Test Result	OK	
Error Class		
Remarks		
Validation	Succeeded	
### Organisational Data ###		
Tester	Katrin Eisenreich	
Test date	12.04.2006	

5.1.3.4 *Testcase004_SAP_CONTENT_IV OK /NOK*

Positive test / Module test			
Core function: RETRIEVE RESULTS			
Pre-requisites: none			
	Test ID		**4.4.**
	### Test activity ###		
	1.1	Retrieve result for request to deviceX where deviceX has connected after request has been issued	
	1.2	Retrieve result for request to deviceX where deviceX has not connected after request has been issued	
	1.3	Retrieve result for invalid requestId (i.e. requestId that has not been assigned to a request)	
	### Expected result ###		
	1.1	Result String with result or error message if request couldn't be processed	
	1.2	Message stating that request has not yet been processed	
	1.3	Message informing about invalid requestId	
	### Realised Result (Textual) ###		
	1.1	Result String with result or error message if request couldn't be processed	
	1.2	Message stating that request has not yet been processed	
	1.3	Message stating that request has not yet been processed	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	NOK	Error Class 4: not distinguishing between pending requests and non-existing requests
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		4 (not critical)
	Remarks		For 1.3 Check against buffered requests in order to inform about invalid requestIds
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		18.04.2006

5.1.3.5 Testcase005_SAP_METADATA_I OK

Positive test / Module test			
Core function: REGISTRATION			
Pre-requisites: none			
	Test ID		**4.5.**
	### Test activity ###		
	1.1	Start a device that has already been registered with the Device Manager	
	1.2	Start a device that has already been registered with the Device Manager and allow to connect	
	1.3	Start a device that has already been registered with the Device Manager and refuse connection	
	### Expected result ###		
	1.1	Device gets connected if connectable==true in registry	
	1.2	Device gets connected; registry entry with connectable==true; metadata (available infoItems) is set	
	1.3	Connection refused; registry entry with connectable==false;	
	### Realised Result (Textual) ###		
	1.1	Device gets connected if connectable==true in registry;	
	1.2	Device gets connected; registry entry with connectable==true; metadata (available infoItems) is set	
	1.3	Connection refused; registry entry with connectable==false;	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		12.04.2006

5.1.3.6 Testcase006_SAP_METADATA_II NOK

Positive test / Module test			
Core function: METADATA UPDATE			
Pre-requisites: none			
	Test ID	**4.6**	
	### Test activity ###		
	1.1	Add an infoItem to a device	
	1.2		
	1.3		
	### Expected result ###		
	1.1	InfoItem item becomes available on device and infoItemId is added to metadata storage	
	1.2		
	1.3		
	### Realised Result (Textual) ###		
	1.1	InfoItem item becomes available on device	
	1.2		
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	NOK	New Metadata information is not yet propagated to RHL
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result	NOK	
	Error Class	2 (strong)	
	Remarks	Propagation of metadata update to RHL can not be done neatly with the current DHL implementation, but a provisional solution will be relatively easy.	
	Validation	Failed	
	### Organisational Data ###		
	Tester	Katrin Eisenreich	
	Test date	18.04.2006	

5.1.3.7 Testcase007_SAP_METADATA_II NOK

Positive test / Module test			
Core function: METADATA RETRIEVAL			
Pre-requisites: none			
	Test ID		**4.7**
	### Test activity ###		
	1.1	Get list of devices	
	1.2	Get list of infoItems for a deviceX	
	1.3		
	### Expected result ###		
	1.1	List of devices registered with device registry	
	1.2	List of infoItems available on deviceX	
	1.3		
	### Realised Result (Textual) ###		
	1.1	List of devices registered with device registry	
	1.2	List of infoItems available on deviceX	
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	Error Class:
	1.2	OK	
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		18.04.2006

5.1.4 Non-functional test cases at SAP

5.1.4.1 Testcase009_SAP_Scalability_Pretest OK / NOK

Positive test / Module test			
Core function: Scalability Pretest			
Pre-requisites: none			
	Test ID		**5.0**
	### Test activity ###		
	1.1	Raise quantity of UPnP devices. Find maximum number of detected devices.	
	1.2		
	1.3		
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	-	
	1.3	-	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2	-	
	1.3	-	
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006

5.1.4.2 Testcase010_SAP_Scalability_I OK / NOK

Positive test / Module test			
Core function: Scalability I			
Pre-requisites: none			
	Test ID	**5.1**	
	### Test activity ###		
	1.1	Create 10 UPnP devices. Send one request to every device. Log performance.	
	1.2	Create 100 UPnP devices. Send one request to every device. Log performance.	
	1.3	Raise number of UPnP devices. Send one request to every device. Log performance. Find limitation.	
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	TBD	
	1.3	TBD	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks		
	Validation	Succeeded / Succeeded with limitation / Not succeeded	
	### Organisational Data ###		
	Tester	Bernhard Wolf	
	Test date	18.04.2006	

5.1.4.3 *Testcase011_SAP_Scalability_II OK / NOK*

Positive test / Module test			
Core function: Scalability II			
Pre-requisites: none			
	Test ID		**5.2**
	### Test activity ###		
	1.1	Create 10 requests for one UPnP device. Log performance.	
	1.2	Create 100 requests for one UPnP device. Log performance.	
	1.3	Create 1000 requests for one UPnP device. Log performance.	
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	TBD	
	1.3	TBD	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006

5.1.4.4 Testcase012_SAP_Scalability_III OK / NOK

Positive test / Module test			
Core function: Scalability III			
Pre-requisites: none			
	Test ID		**5.3**
	### Test activity ###		
	1.1	Create multiple requests for multiple UPnP devices. Log performance.	
	1.2	Variation possible.	
	1.3		
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	-	
	1.3	-	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2	-	
	1.3	-	
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006

5.1.5 Summary for parts of SAP

With the tests listed above (and their update in “Appendix B – Functional Test cases for modules of SAP (Updated)”) we have evaluated the conformance of the developed components and their integration with the specification. A number of errors and missing features have been identified, part of which could already be fixed before a second test was run. As indicated in the test document, there is still a number of shortcomings which must be corrected, and exception handling must be improved. Also, the subscription feature still lacks a valid implementation, which will be provided in the next weeks. As a next step a set of scalability tests will be carried out to evaluate the components' performance depending on the quantity of devices and requests to be handled.

5.2 Functional Test cases for modules of HUT

The aim of this section is to verify the correct implementation of the WP R6 modules based on the following test cases. The test cases are based on the interface specification made in DR 6.3, section 5. The UPnP-based interfaces defined in section 4 are not included in the HUT implementation.

5.2.1 Guidelines and comments for testing

System / Application to be tested	PROMISE R6 Device Controller Implementation of HUT.
System version	N/A
Test risks	Changes on implementation or test system during the test phase
Pre-conditions	Internal pre-tests are made
Components and functions / units which are not tested	
Components and functions / units which are tested	Existing implementation at the test date.
Test data (Test values)	Test data is derived from the specification
Testing user	Kary Främling, Lorenzo Marra
Remarks	Without errors, the test case will marked OK With errors, the testcase marked with NOK (also in the section title to see easily with one view which tests were successful and which ones were not.

5.2.2 Test environment

This section explains the requirements for the test environment. The first subsection explains hardware-related requirements and the second explains software-related requirements. In the third subsection, the DIALOG system developed at HUT is described, together with a description of how it has been used as a base for implementing the Web Service interface defined in PROMISE DR6.3, section 5. The fourth subsection describes what tests are performed, what test data has been used and how the test environment has been implemented.

The purpose of this section is to describe the test environment and the tests performed in a way that makes it possible to repeat and verify the tests for any third-party organisation.

Hardware

The hardware requirements are:

1. Availability of Java Virtual Machine (JVM) for the hardware platform used and
2. IP network connectivity.

Server Hardware

Because the implemented PROMISE interface is a Web Service interface, it puts some requirements on the available memory and hard disk space. In addition to the space needed to run a Java Servlet container (e.g. Apache Tomcat), there also needs to be space available for the Web Service libraries (e.g. about 15 Mbytes for the Apache Axis libraries).

Client Hardware
For the moment there is no need to run a Java Servlet container on the "client" side. However, the Web Service libraries are needed also on the client side for performing necessary Java to Web Service conversions.

Software
The software necessary to run the PROMISE Modules is given by the following table:

Operating System	Any operating system that supports Java Virtual Machines
Java Virtual Machine	Version 1.5.0 has been used in the tests. It is recommended to use this version or more recent ones, even though older Java versions might also be possible.
Java Servlet Container	Jakarta Tomcat version 5.5 from the Apache Software Foundation has been used in the tests reported here. In principle, any compatible Java Servlet Container should be usable.
Web Service (SOAP, WSDL) implementation	The Axis implementation, version 1.3, has been used in the tests reported here. In principle, any Web Service implementation could be used, except for one class that is Axis-specific and is used for retrieving the installation directory of the Web Service.
Application software	The DIALOG software plus corresponding implementation of the Web Service interface defined in PROMISE DR6.3, section 5.
Database	No database is needed for the tests performed here. On a more general level, any JDBC-compliant database can be used, even though only some of the most common ones are tested and explicitly supported.

5.2.3 Description of DIALOG software architecture

The background and the main design principles of the DIALOG architecture are described in many scientific articles and various PROMISE deliverables, e.g. section 6.2 of DR6.3. From the communication point of view, DIALOG is essentially a peer-to-peer (P2P) architecture. As for other P2P systems, all DIALOG nodes are equal, i.e. they can both send and receive messages. Therefore any node in the network can for instance both ask for information and provide information for the items that it has information about, as shown in Figure 4. Other major design principles adopted from P2P are low installation overhead, equality between parties and scalability.

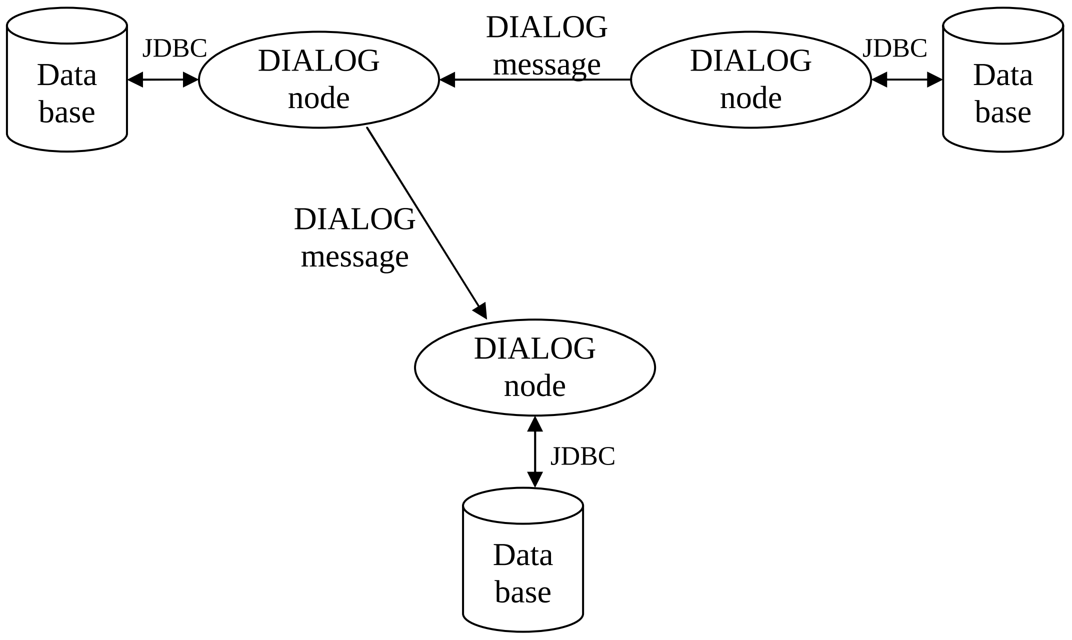


Figure 4. P2P communication between DIALOG nodes. Any node can both send and receive messages. JDBC: Java Database Connectivity.

The main components or *objects* of a DIALOG node and their functionality are illustrated in Figure 5. The only external function (or method) of a DIALOG node is the "receive" function that takes a message object of type DialogMessage as its only parameter and returns an object of the same type. Three "receiver" types that contain this function exist for the moment, which support different messaging protocols, i.e. SOAP (*SoapReceiveImpl*), RMI (*RMIReceiveImpl*) and HTML <FORM> messaging (*DialogHTMLProductAgent*) used by standard CGI- or servlet-type web applications. Which messaging protocol is used depends on how the DIALOG node is started. Any number of messaging protocols can be running concurrently as long as they can be separated by port number, path names or other standard URL components.

The "ReceiveImpl" object is created at startup. It also has a "receive" function that is called by the "receiver" object when a message is received. If the received message is synchronous (i.e. time-to-live, TTL, is zero), then the message is passed to all "agent" objects that have registered for receiving messages indicated by the "type" field of the message. If the received message is asynchronous, it is buffered in the "receive" buffer and given to the appropriate agent(s) by a thread. Agents can also be described by the notion of "plug-in", i.e. a software component that can be used for extending the functionality of the "core" system at any time. Agent objects are therefore a way of implementing the "plug-in" functionality mentioned in the R6 section of the PROMISE Description of Work. For simplicity, we will use the "agent" concept for the rest of this text.

Every agent has a reference to the "SendImpl" object, whose function "send" is called with a DialogMessage object as parameter. The "send" method also returns a DialogMessage object that either contains the requested information (e.g. for a successful synchronous "read" request) or some other status information about the sending of the message. If the message to send is synchronous, it is sent directly to the appropriate "sender" object for delivery and the result is returned if successful. If the message to send is asynchronous, it is buffered and sent by a sending thread using the appropriate "sender" object. The "sender" object to use is determined either from the type of the message (configurable) or the protocol part of the destination URI/URL.

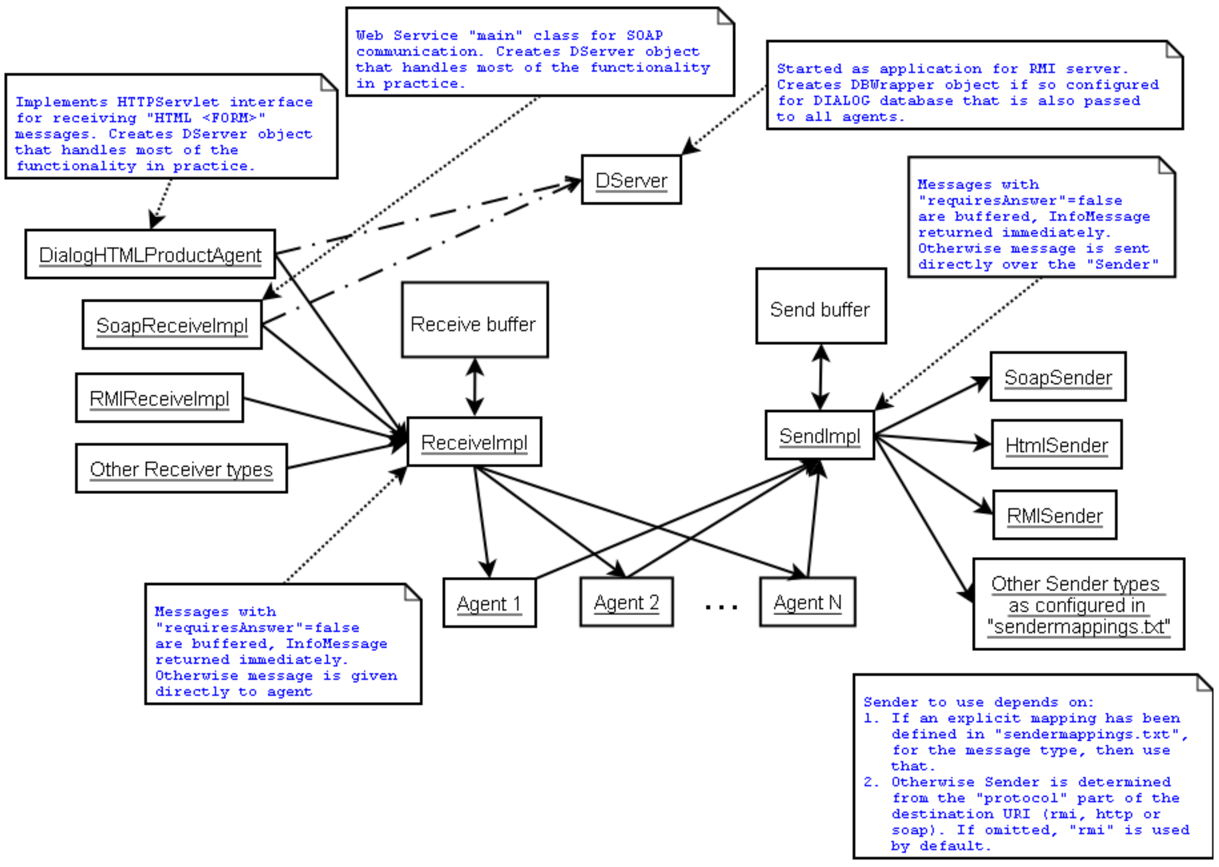

Web Service "main" class for SOAP communication. Creates DServer object that handles most of the functionality in practice.
Started as application for RMI server. Creates DBWrapper object if so configured for DIALOG database that is also passed to all agents.
Implements HTTPServlet interface for receiving "HTML <FORM>" messages. Creates DServer object that handles most of the functionality in practice.
DServer
Messages with "requiresAnswer"=false are buffered, InfoMessage returned immediately. Otherwise message is sent directly over the "Sender"
DialogHTMLProductAgent
SoapReceiveImpl
Receive buffer
Send buffer
RMIReceiveImpl
SoapSender
Other Receiver types
ReceiveImpl
SendImpl
HtmlSender
RMISender
Other Sender types as configured in "sendermappings.txt"
Agent 1
Agent 2
. . .
Agent N
Messages with "requiresAnswer"=false are buffered, InfoMessage returned immediately. Otherwise message is given directly to agent
Sender to use depends on:
1. If an explicit mapping has been defined in "sendermappings.txt", for the message type, then use that.
2. Otherwise Sender is determined from the "protocol" part of the destination URI (rmi, http or soap). If omitted, "rmi" is used by default.

Figure 5. Illustration as UML object diagram of main internal components of a DIALOG node and their functionality.

The implementation of the PROMISE DR6.3 interface has been done by adding a new "receiver" type called "PromiseDCinterface" that converts incoming messages into appropriate DialogMessage objects and forwards them to the "ReceiveImpl" object, where the actual treatment is taken care of by the appropriate agent depending on the message type. A corresponding "sender" called "PromiseDCSender" takes care of sending messages using the PROMISE DR6.3 interface when the message type indicates it (as configured in the file "sendermappings.txt").

These receiver and sender mappings make it possible to use the basic DIALOG messaging, buffering and other functions but they do no processing of the information. All processing (or "business logic") is performed by the agent objects. By default, agents are initialized with references to the DServer, ReceiveImpl and SendImpl objects and a reference to a JDBC database object if a database is configured. This allows them to easily implement basic functionality, but it does not prevent them from implementing any kind of advanced functionality, such as accessing the file system, using own databases or calling web applications.

A DIALOG agent consists of one or a set of Java classes. In practice, most agents consist of one single class that is a subclass of the abstract class "fi.hut.dialog.agents.DialogAgent". Agent classes usually implement both the server-side logic and a GUI that can be used in "client" applications. This approach has the advantage of combining all the application-specific processing into one single component (i.e. the DIALOG agent) that is independent of the rest of the system. It is also possible to separate "client" and "server" logic if that is desired but the size of a typical DIALOG agent is only 10-20 kbytes (this is the case for the test agents used here) so a separation is usually not interesting in practice. The "client" functionality could also be something different than a GUI, e.g. a component that is UPnP-enabled and implements the CorePAC interface defined in PROMISE DR4.2.

5.2.4 Test data

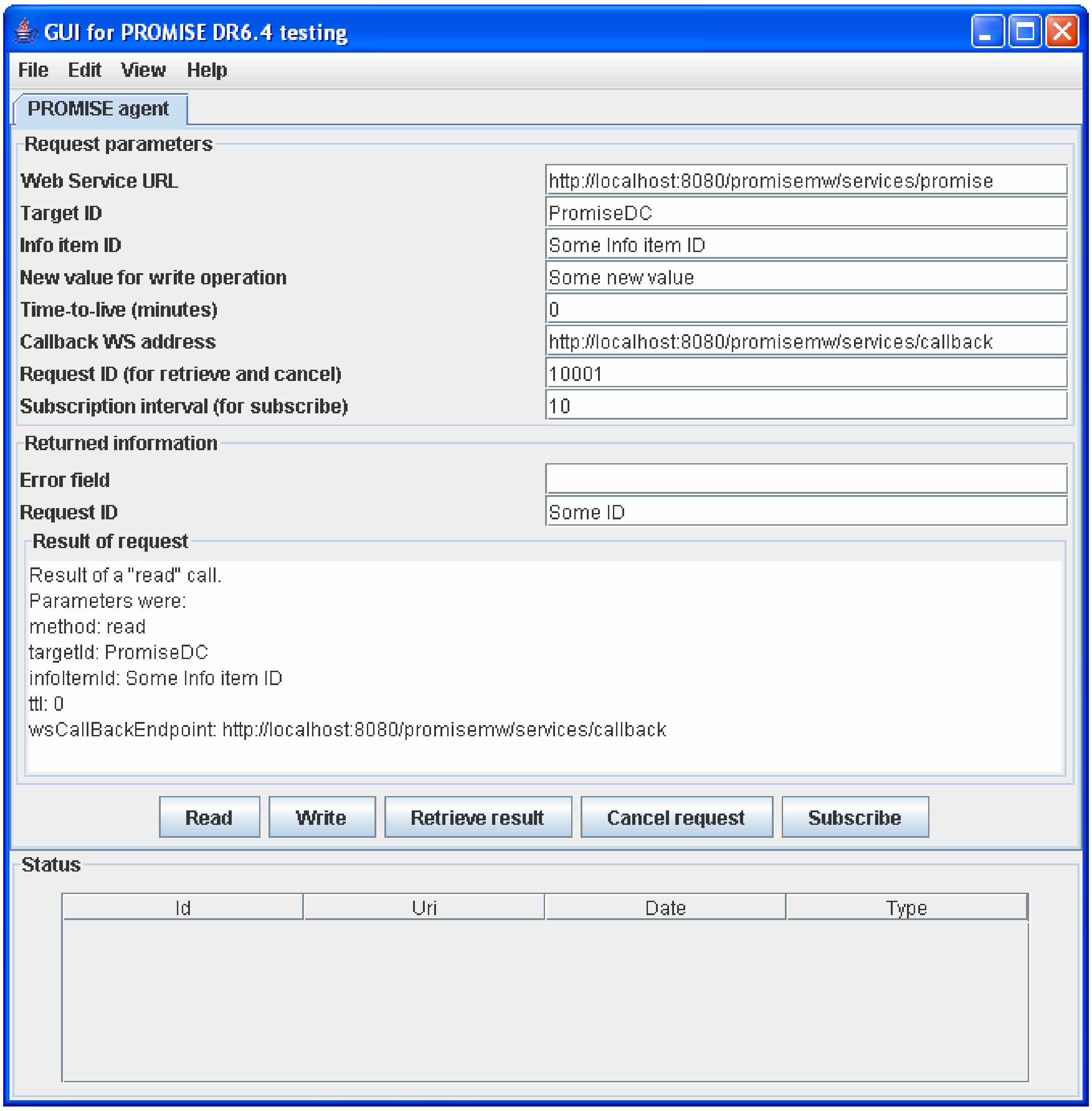
Three different types of tests are conducted here:

1. **Basic messaging**: messages with zero TTL are sent by a "test GUI" to the Web Service, which transforms the function parameters into a DIALOG message that is forwarded to the appropriate agent. The agent responds by a new DialogMessage object that is transformed into a PromiseMWresult object and given back to the caller as a return value.
2. **Time-to-live functionality and message buffering**: messages are sent from "test GUI" with zero, greater than zero and -1 TTL values and check they are buffered for the requested duration.
3. **Inter-organisational communication with "real-life" agent implementation**: developed for and executed with MTS (for PROMISE demonstrator A7). The agent functionality consists of a GUI part (installed at MTS) that allows the user to select a text file to transfer and send it to the receiving Web Service (installed at HUT) through the "write" function.

For the "basic messaging" tests, the following functions in section 5.1 "Request Handling" of DR6.3 have been implemented for the moment:

- "read"
- "write"
- "retrieveResult"
- "cancelRequest"
- "subscribe"

As there is no "business logic" associated with these functions, they return a PromiseMWresult object with a "result" value that is a text string containing the parameters and parameter values of the call. This is sufficient for test purposes because it shows that the message has reached the receiving agent, been processed by it and that an appropriate return value has been returned. Figure 6 shows a screenshot of the GUI developed for "basic messaging" and "time-to-live" tests. In Figure 6, "http://localhost:8080/promisemw/services/promise" is the URL where the Web Service is installed, in this case on the local computer, i.e. same computer as the "client" application. The "Result of request" field shows the return value from a "read" operation. The field "Request ID" shows the request ID returned, which for this test is set to the constant value "Some ID". Finally, the "Status" area at the bottom is for showing messages that are in the "send" buffer and that have not been sent yet. In TTL testing, this status area is also used for verifying the correct functionality.



Request parameters

Web Service URL	http://localhost:8080/promisemw/services/promise
Target ID	PromiseDC
Info item ID	Some Info item ID
New value for write operation	Some new value
Time-to-live (minutes)	0
Callback WS address	http://localhost:8080/promisemw/services/callback
Request ID (for retrieve and cancel)	10001
Subscription interval (for subscribe)	10

Returned information

Error field	
Request ID	Some ID

Result of request

```
Result of a "read" call.
Parameters were:
method: read
targetId: PromiseDC
infoItemId: Some Info item ID
ttl: 0
wsCallBackEndpoint: http://localhost:8080/promisemw/services/callback
```

Status

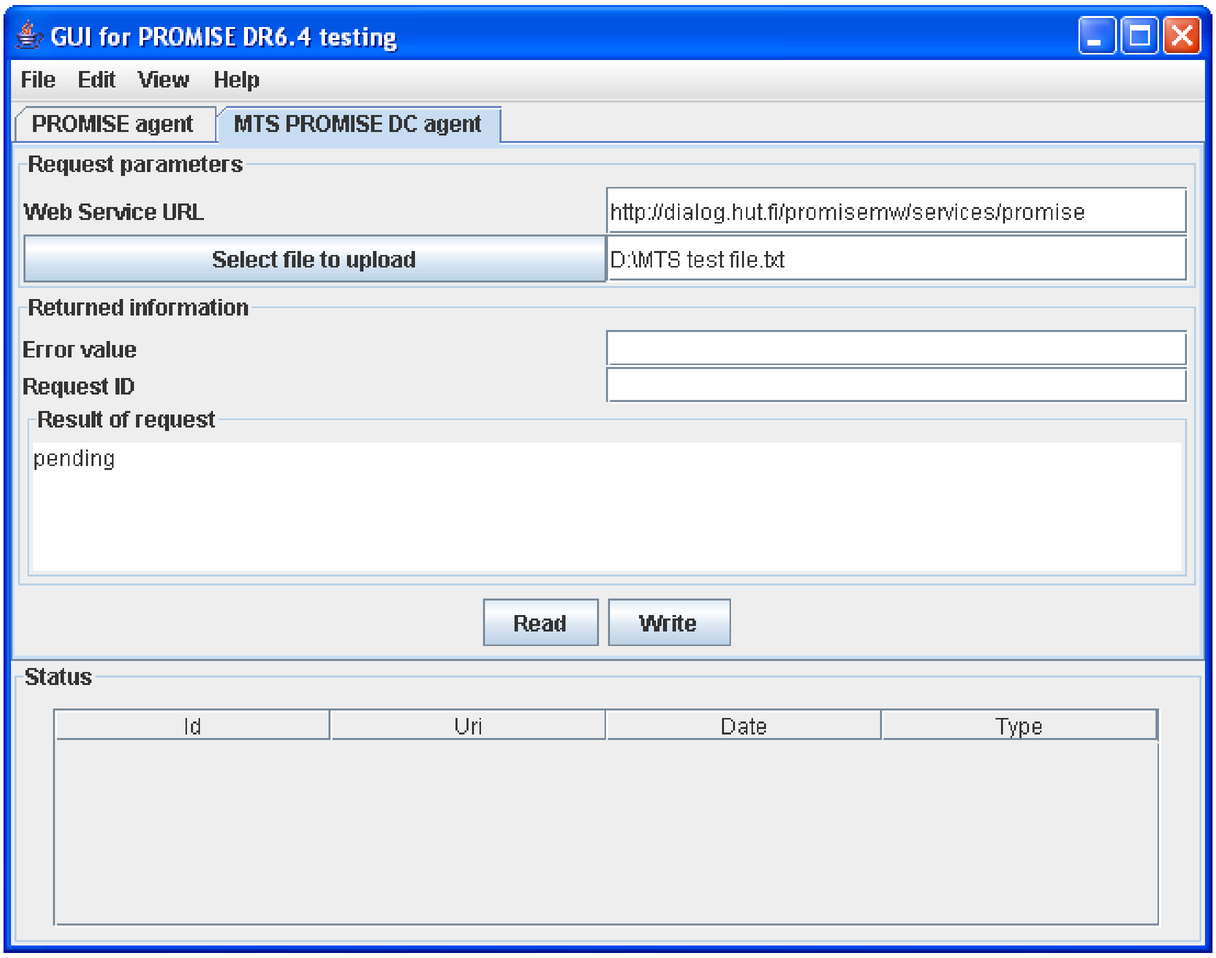
Id	Uri	Date	Type

Figure 6. GUI developed for "basic messaging" and "time-to-live" tests.

The functions defined in section 5.2 "Metadata" and 5.3 "Group Management" of DR6.3 exist in the WSDL interface implementation but no functionality has been implemented for them yet. Therefore only a messaging-level test is possible for the moment. Still, successful results from the tests performed on the functions defined in section 5.1 "Request Handling" of DR6.3 is sufficient to show that the basic messaging-level is operational also for the functions in section 5.2 because all functions are implemented in a similar way.

For the moment, the value of the "Target ID" parameter is used as the "message type" in the DIALOG implementation. DIALOG agents can register to listen for different types of messages. The test agent for cases one and two registers for listening to messages of type "PromiseDC", which is an arbitrarily decided text string. It is therefore essential to use "PromiseDC" as the value of "Target ID" in order to route it to the right agent. This mapping from PROMISE "Target ID" to DIALOG message type may be specific only for the test cases presented here. The actual semantic interpretation of the PROMISE "Target ID" may be application-specific in the future.

In addition to these "internal" tests performed at HUT, an inter-organisational test was set up for the PROMISE A7 demonstrator with the company MTS. The tested functionality was asynchronous sending (i.e. TTL>0) of a text file containing boiler field data by using the "write" function. The goals of this test were to implement some real functionality by an agent and verify that communication works in a multi-organisational setting. This agent registers as listening to messages of type "PromiseDC_MTS". In Figure 7, "http://dialog.hut.fi/promisemw/services/promise" is the URL where the Web Service is installed. This URL is on a remote computer behind a firewall that limits the network traffic, so the server has been configured to use the standard HTTP port 80 instead of port 8080 used in Figure 6.



Web Service URL: http://dialog.hut.fi/promisemw/services/promise

D:\MTS test file.txt

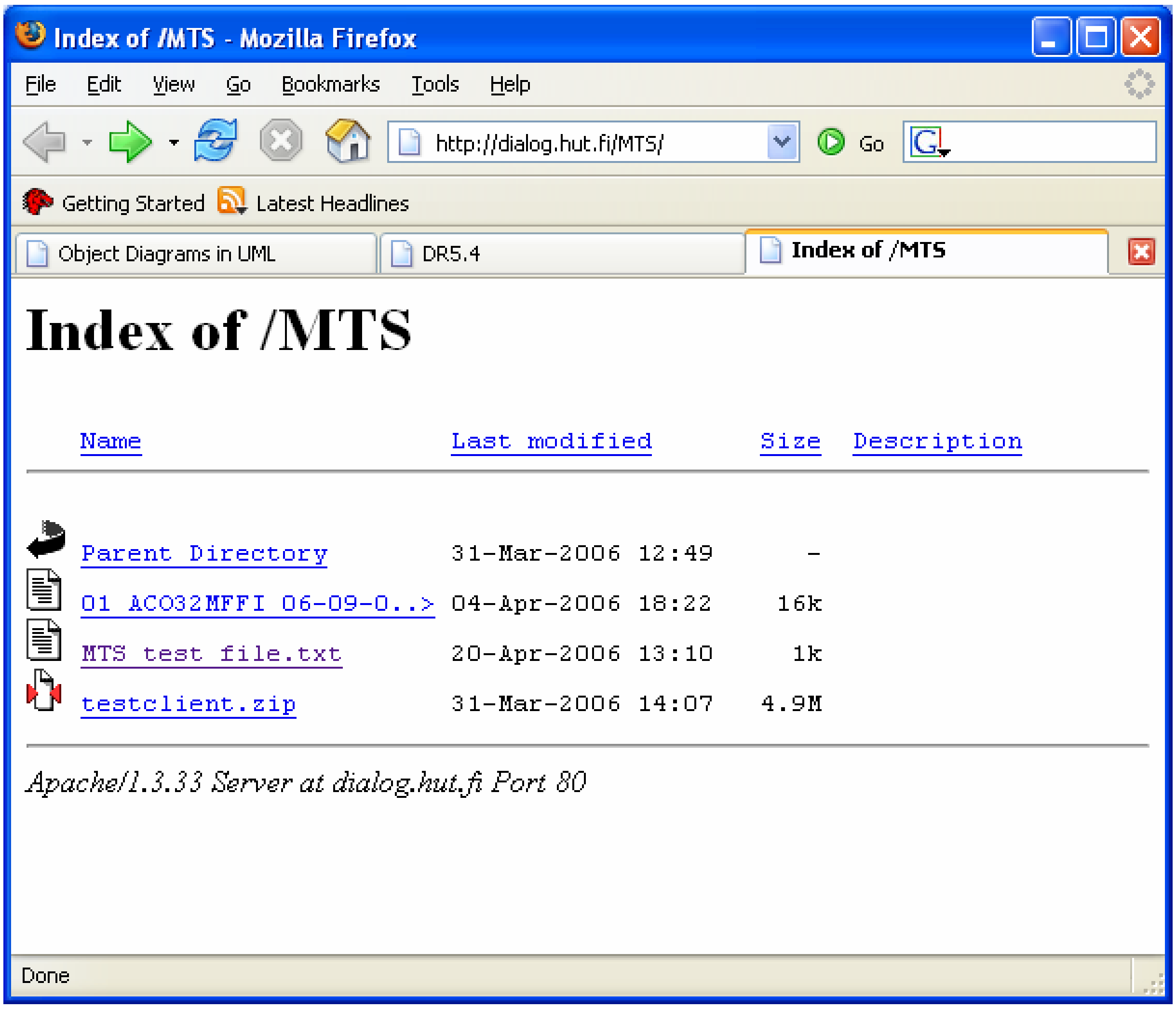
Error value:

Request ID:

pending

Figure 7. GUI developed for inter-organisational test with MTS.

The MTS agent at "dialog.hut.fi" is configured to store the received file into a browser-accessible directory so that the success of the file transfer can be verified directly. Figure 8 shows the contents of this directory after clicking on the "Write" button of the window shown in Figure 7. The file "MTS_test_file.txt" is the one that was transmitted in Figure 7. The other two files are 1) a Zip file that contains the "client" application downloaded and used by MTS and 2) the file transmitted from MTS. This directory was also used for testing with MTS, who downloaded the "client" application shown in Figure 7 from this directory. The file transmitted from MTS is also stored into the same directory. The "pending" result of the request is because this is an asynchronous request with TTL set to one minute. This means that the message is buffered until it can be sent successfully or until TTL expires, but the "client" application regains control immediately.


Index of /MTS - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://dialog.hut.fi/MTS/
Go
Getting Started
Latest Headlines
Object Diagrams in UML
DR5.4
Index of /MTS
Done

Index of /MTS

```
    Name                      Last modified       Size  Description

    Parent Directory          31-Mar-2006 12:49      -
    01 ACO32MFFI 06-09-0..>   04-Apr-2006 18:22    16k
    MTS test file.txt         20-Apr-2006 13:10     1k
    testclient.zip            31-Mar-2006 14:07   4.9M

Apache/1.3.33 Server at dialog.hut.fi Port 80
```

Figure 8. Uploaded files are directly stored into a browser-accessible location, where the the transmitted file can be accessed.

5.2.5 Test cases

5.2.5.1 *Testcase001_HUT_BasicMessaging* ***OK***

Positive test / Module test			
Core function: Basic Messaging			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	Call "read" function, return text that shows the parameters and their values	
	1.2	Call "write" function, return text that shows the parameters and their values	
	1.3	Call "retrieveResult" function, return text that shows the parameters and their values	
	1.4	Call "cancelRequest" function, return text that shows the parameters and their values	
	1.5	Call "subscribe" function, return text that shows the parameters and their values	
	### Expected result ###		
	1.1	The "result" field of the returned "PromiseMWResult" should contain text that shows the call parameters and their values	
	1.2	The "result" field of the returned "PromiseMWResult" should contain text that shows the call parameters and their values	
	1.3	The "result" field of the returned "PromiseMWResult" should contain text that shows the call parameters and their values	
	1.4	"true", shown by a text displayed in the test GUI	
	1.5	The "result" field of the returned "PromiseMWResult" should contain text that shows the call parameters and their values	
	### Realised Result (Textual) ###		
	1.1	Test GUI "result" field shows correct list of parameters and parameter values as defined in DR6.3, section 5	
	1.2	Test GUI "result" field shows correct list of parameters and parameter values as defined in DR6.3, section 5	
	1.3	Test GUI "result" field shows correct list of parameters and parameter values as defined in DR6.3, section 5	
	1.4	Test GUI "result" field shows correct list of parameters and parameter values as defined in DR6.3, section 5	
	1.5	Test GUI "result" field shows correct list of parameters and parameter values as defined in DR6.3, section 5	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	OK	
	1.4	OK	
	1.5	OK	
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class		
	Remarks		
	Validation	Succeeded	
	### Organisational Data ###		
	Tester	Kary Främling, HUT	
	Test date	20.04.2006	

5.2.5.2 *Testcase002_HUT_TTL_Buffering OK*

Positive test / Module test			
Core function: Time-to-live functionality (buffering)			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	Call one of the implemented functions with TTL=0, using an inexistent Web Service address or without having started the Web Service. Check that message is not buffered into "send" buffer, which means that the command window will not show any send attempts and that the size of the buffer file does not change	
	1.2	Call one of the implemented functions with TTL > 0, using an inexistent Web Service address or without having started the Web Service. Check that message is buffered into "send" buffer, which means that the command window shows send attempts and that the size of the buffer file changes. Check that send attempts stop after TTL time has elapsed and that buffer file size decreases.	
	1.3	Call one of the implemented functions with TTL = -1 ("keep forever") , using an inexistent Web Service address or without having started the Web Service. Check that message is buffered into "send" buffer, which means that the command window will show send attempts and that the size of the buffer file changes. The "forever" functionality is obviously not possible to test but a "long" time can be considered sufficient – in this case 10 minutes was used. This can be considered "sufficient" because there is no reason why the message would disappear from the buffer once it has been inserted there due to how the buffering functionality is implemented.	
	1.4	Call one of the implemented functions with 0 TTL, but without registering the PromiseAgent agent at the receiver node, which means that the message cannot be received. Check that message is not buffered into "receive" buffer, which means that the command window will not show any send attempts and that the size of the buffer file does not change.	
	1.5	Call one of the implemented functions with TTL > 0, but without registering the PromiseAgent agent at the receiver node, which means that the message cannot be received. Check that message is buffered into "receive" buffer, which means that the command window will show send attempts and that the size of the buffer file changes. Check that send attempts stop after TTL time has elapsed and that buffer file size decreases.	
	1.6	Call one of the implemented functions with TTL = -1 ("keep forever"), but without registering the PromiseAgent agent at the receiver node, which means that the message cannot be received. Check that message is buffered into "receive" buffer, which means that the command window will show send attempts and that the size of the buffer file changes. The "forever" functionality is obviously not possible to test but a "long" time can be considered sufficient – in this case 10 minutes was used. This can be considered "sufficient" because there is no reason why the message would disappear from the buffer once it has been inserted there due to how the buffering functionality is implemented.	
	### Expected result ###		
	1.1	Size of "send" buffer does not change, no send attempts show in command window.	
	1.2	Command window shows attempts to send for given TTL, size of "send" buffer increases during sending attempts.	
	1.3	Command window shows attempts to send "forever" (10 minutes will be considered sufficient), , size of "send" buffer increases.	
	1.4	Size of "receive" buffer does not change, no send attempts show in command window	
	1.5	Command window shows attempts to send for given TTL, size of "receive" buffer increases during sending attempts.	
	1.6	Command window shows attempts to send "forever" (10 minutes will be considered sufficient), size of "receive" buffer increases.	
	### Realised Result (Textual) ###		
	1.1	Works OK, result is shown directly both in case of success and in case of error	
	1.2	Status "pending" is shown and send attempts are shown for the time indicated by TTL. For some reason the buffer sizes do not become smaller after TTL has expired but the output shown in the buffer status area and in the command window are sufficient to show that this functionality works.	
	1.3	Status "pending" is shown and send attempts go on forever. If the "server" is started at the indicated URL, then the sending succeeds and the message is removed from the buffer.	
	1.4	Works OK, error is returned immediately and message is not buffered	
	1.5	The command window shows that the "receive" thread tries to hand to message to the agent/plug-in for the indicated time and then the message disappears	
	1.6	The command window shows that the "receive" thread tries to hand to message to the agent/plug-in "forever". When stopping the Web Service, adding the agent/plug-in to the list of agents and then re-starting, the message disappears from the buffer.	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	OK	

	1.4	OK	
	1.5	OK	
	1.6	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Kary Främling, HUT
	Test date		20.04.2006

5.2.5.3 Testcase003_HUT_FieldData OK

Positive test / Module test			
Core function: *Sending boiler field data as text file from MTS to HUT web* server by calling “write” function			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	Installing Web Service on “dialog.hut.fi”, configuring it to port 80, configuring the directory where files should be stored and testing the installation	
	1.2	Downloading test “client” application to MTS, installing it and transmitting text file with field data to “dialog.hut.fi”	
	### Expected result ###		
	1.1	Web Service responds to port 80, stores files into configured location	
	1.2	Transmitted text file is stored into configured directory	
	### Realised Result (Textual) ###		
	1.1	As expected	
	1.2	As expected.	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class		
	Remarks		
	Validation	Succeeded	
	### Organisational Data ###		
	Tester	Kary Främling, Lorenzo Marra	
	Test date	04.04.2006	

5.2.6 Summary for parts of HUT

The main purpose of the tests performed by HUT was to study how easily PROMISE Web Service interfaces can be implemented by using HUT's DIALOG system as the base platform. This was done with relatively little effort and with good test results, including tests performed in a multi-organisational setting. Because the PROMISE-specific part of the implementation counts for less than 1% of the entire source code, these results illustrate that PROMISE Web Service interfaces can be successfully integrated into existing implementations without too much effort.

5.3 Functional Test cases for modules of Stockway

The aim of this section is to verify the correct implementation of the WP R6 modules based on the following test cases. The test cases are based on the specification made in DR 6.2 and DR 6.3.

5.3.1 Guidelines and comments for testing

System / Application to be tested	PROMISE R6 Device Controller Interfacing Module Implementation of Stockway.
System version	N/A
Test risks	Changes on implementation or test system during the test phase. Data security rules which complicates and makes the Stockway / SAP joint tests impossible to execute as planned.
Pre-conditions	Internal pre-test are made
Components and functions / units which are <u>not</u> tested	Device Controller • Stockway has not implemented the actual DC functionality so the DC is not tested. IOCI against specifications • As IOCI specification work starts M18.
Components and functions / units which are tested	The following is tested by Stockway: • Trackway system and WWAI concept as communication infrastructure of PEID data in Promise context. • Trackway to DC integration module
Test data (Test values)	Test data is derived from the specification
Testing user	
Remarks	Without errors, the test case will marked OK With errors, the testcase marked with NOK (also in the section title to see easily with one view which tests are passed or not.

5.3.2 Test Environment - Stockway

The test environment set up by Stockway is for performing two basic types of tests. The first type of tests is for testing the WWAI concept and the Trackway software as communication infrastructure for Promise PEID data. The second is for testing interoperability between the Promise module made for Stockway´s Trackway software and the SAP Device Controller implementation. The goal with the second test is to prove the correctness of both SAP and Stockway implementations.

<u>Hardware</u>

Different PCs are involved in the tests. All software components tested in these tests are server software components. Here we describe the PCs that will function as servers in the tests. Server

PC 1 will perform most testing activity. Server PC 2 will function as a remote WWAI node for tests of WWAI communication over the Internet. Server PC 3 will be provided by SAP and function as Device Controller implementation. Trackway Promise modules will request the Web Services provided by the DC.

Requirements
Any modern PC can run the Trackway based parts of the Promise system. Recommended minimum is:

Processor	Intel Pentium – 1 GHz
System Memory	512 MB

Server PC 1
Server PC 1 will be the mostly used PC in the tests. It is the main PC for running test applications on. Server PC 1 will physically be located in Stockway's office in Reading, UK.

Processor	Intel Pentium Mobile - 2 GHz
System Memory	2 GB

Server PC 2
Server PC 2 operates a WWAI node that Server PC 1 will do simulated PEID data requests to. Server PC 2 is physically located in Stockway's office in Espoo, Finland.

Processor	Intel Xeon - 3 GHz
System Memory	1 GB

Server PC 3
To be defined by SAP.

<u>Software</u>
The software necessary to run the PROMISE Modules for Trackway is given by the following table:

Requirements

Operating System*	Any with Java 1.5 support – Windows XP recommended*
Java SDK	Version 1.5
Trackway Software	Version 4
Database System	Any SQL database system - Trackway embedded MySQL recommended**

** Installation of Promise modules for Trackway is simplest in Windows XP environment.*
*** Installation of the Trackway system is simplest if using the embedded database.*

Server PC 1
Software environment for Server PC 1:

Operating System	Windows XP
Database System	MySQL (4.0.16)

Java Environment	Java SDK 1.5.0_05

Server PC 2
Software environment for Server PC 2:

Operating System	Linux
Database System	MySQL (4.1.12)
Java Environment	Java SDK 1.5.0_04

<u>Network</u>
Server PC 1 and Server PC 2 are both connected with broadband Internet connections. Transfer speed between the two systems varies, but is efficient enough for the tests. Ping times between the two systems have been monitored on different occasions and are quite stable around 50 ms.

5.3.3 Test Data

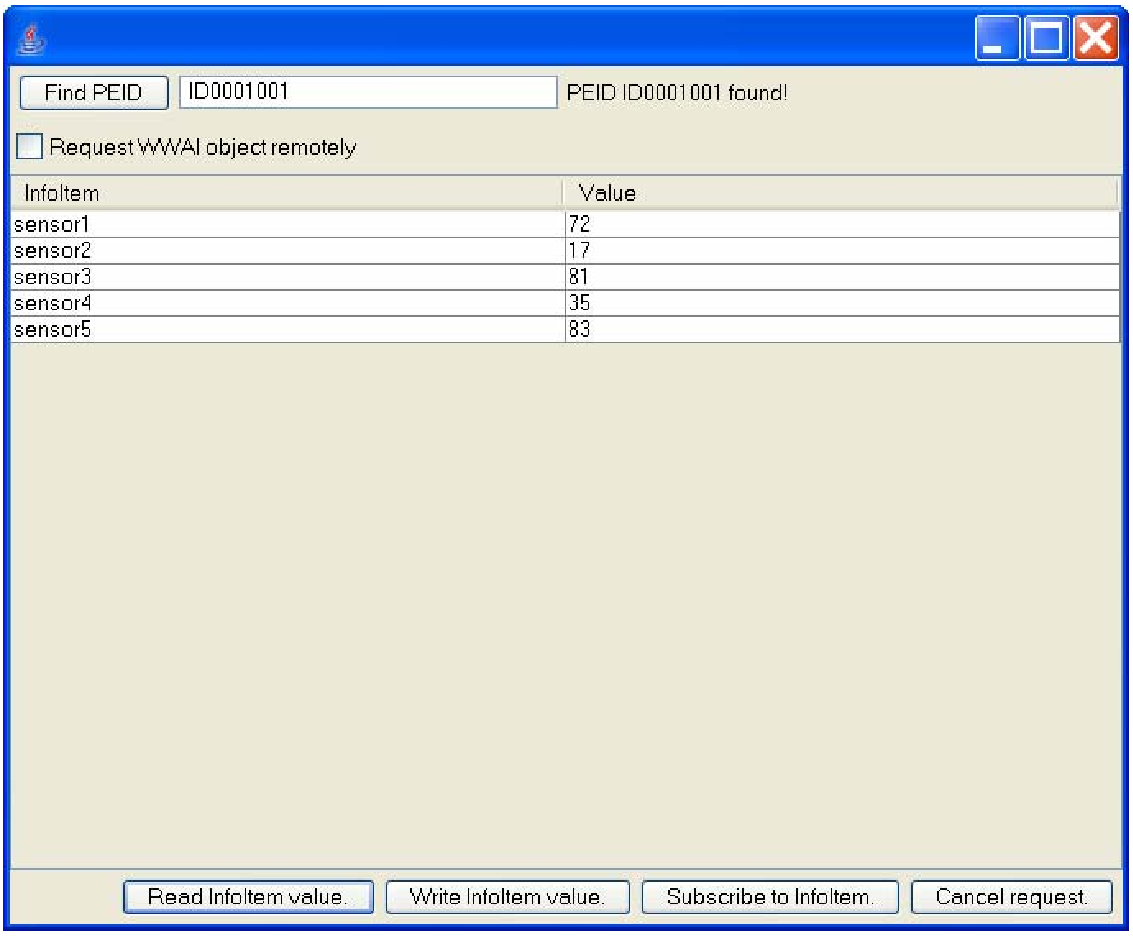
For WWAI and Trackway performance tests, the test data is WWAI object data with some attributes. The WWAI object is a representation of an identifiable item in the WWAI network. The WWAI object represents in the Promise context the PEID or other identifiable object. The object will have 10 randomly created attributes with randomly created data. These attributes represents info items on the PEID.

WWAI Object (PEID data)	
ID	ID number
Attribute 1	Data 1
Attribute 2	Data 2
Attribute 3	Data 3
Attribute 4	Data 4
Attribute 5	Data 5
Attribute 6	Data 6
Attribute 7	Data 7
Attribute 8	Data 8
Attribute 9	Data 9
Attribute 10	Data 10

For the tests of the web service interface, different test data is used. The id used in these tests is defined by the user during the tests. We do not define any coding schemas that are used for id generation for PEIDs so any binary string should be accepted as id for a PEID.

Using the dummy DC implementation from Stockway, requests for info items and info item values are answered randomly. Also, it is randomly decided if a request is answered right away or in a random amount of time (between 0 to 30 seconds). All info item values are regarded as text strings. No data validation is done on the correctness of the data values. We do not need to show that the data make sense as PEID data, as all data is randomly generated.

The format of the xml messages is according to specifications in DR6.3, chapter 5.

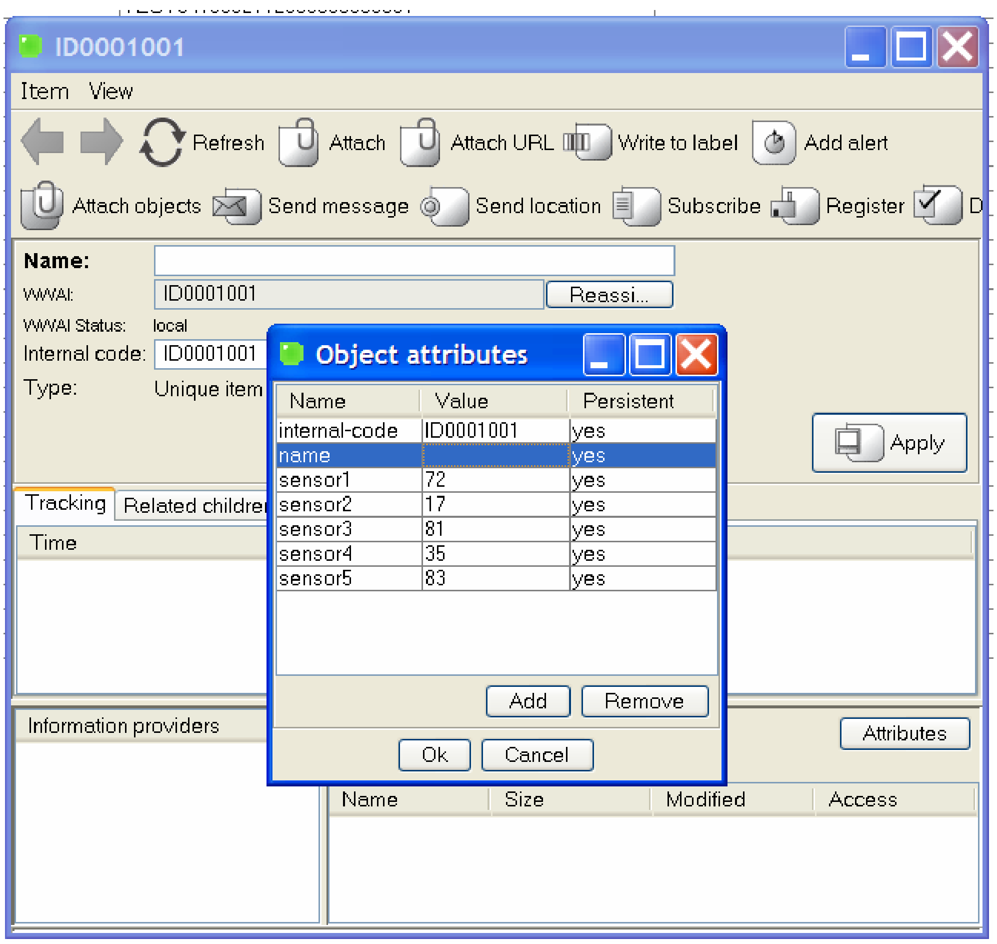


Find PEID | ID0001001 | PEID ID0001001 found!

Request WWAI object remotely

InfoItem	Value
sensor1	72
sensor2	17
sensor3	81
sensor4	35
sensor5	83

Figure 9 ***Test GUI for DC requests. Values for Info Items are updated based on values read from the DC. Corresponding WWAI data is created in the Trackway system.***



Name:

WWAI: ID0001001

WWAI Status: local

Internal code: ID0001001

Type: Unique item

Object attributes

Name	Value	Persistent
internal-code	ID0001001	yes
name		yes
sensor1	72	yes
sensor2	17	yes
sensor3	81	yes
sensor4	35	yes
sensor5	83	yes

Name	Size	Modified	Access

Figure 10 PEID WWAI data shown in the Trackway system using a Trackway development GUI.

5.3.4 Test cases for Modules of Stockway

The integration tests planned between Stockway's system and the SAP device controller could not be performed as planned before this deliverable. We have problems with configuring firewalls to let the Web Service calls through on both Stockway and SAP side.

Also time restrictions made it impossible to perform the tests from Stockway's side. (Only 0.4 man months was planned for Stockway to do tests at this stage and the work with the IOCI (DR6.6) is scheduled to start in June. All tests between the Trackway IOCI implementation and the DC would have been carried out ahead of schedule.)

These tests are planned to be carried out in later during R6 work and before the actual IOCI work will start.

5.3.4.1 *Testcase001_STW_DC_read OK / NOK*

Positive test / Module test			
Core function: Reads an info item value from the DC though the DC Web Service interface and updates the value for the info item in the Trackway System. The DC gives immediate response with the value.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User selects the PEID info item using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service, implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User selects the PEID info item using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	Updated value for the PEID info item in WWAI representation in the Trackway system.	
	1.2	Updated value for the PEID info item in WWAI representation in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The info item values are read correctly from the DC and present in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.2 *Testcase002_STW_DC_read_callBack OK / NOK*

Positive test / Module test			
Core function: Reads an info item value from the DC though the DC Web Service interface. The DC will return with the result later using the Web Service call-back functionality.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User selects the PEID info item using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User selects the PEID info item using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The DC accepts the request and will return with the result when available.	
	1.2	The DC accepts the request and will return with the result when available.	
	### Realised Result (Textual) ###		
	1.1	The info item values are update correctly from the DC using the call-back service and present in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.3 *Testcase003_STW_DC_write OK / NOK*

Positive test / Module test			
Core function: Writes an info item value to the DC though the DC Web Service interface and updates the value for the info item in the Trackway System.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User selects the PEID info item to write and gives the new value using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User selects the PEID info item to write and gives the new value using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	Updated value for the PEID info item WWAI representation in the Trackway system and updated value for the info item in the DC implementation.	
	1.2	Updated value for the PEID info item WWAI representation in the Trackway system and updated value for the info item in the DC implementation.	
	### Realised Result (Textual) ###		
	1.1	The info item value is written correctly to the DC and new value present in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.4 Testcase004_STW_DC_retrieveResult OK / NOK

Positive test / Module test			
Core function: Retrieves a result for a request from the DC and updates the values correspondingly in the Trackway System.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User gives a request id using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User gives a request id using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	If the request can be answered, the values are updated accordingly for the PEID info item WWAI representation in the Trackway system. If it cannot be answered, no values are changed.	
	1.2	If the request can be answered, the values are updated accordingly for the PEID info item WWAI representation in the Trackway system. If it cannot be answered, no values are changed.	
	### Realised Result (Textual) ###		
	1.1	The info item values are read correctly from the DC and present in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.5 *Testcase005_STW_DC_cancelRequest OK / NOK*

Positive test / Module test			
Core function: Cancels a previously made request. Requests a cancellation from the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User gives the request id of the request to cancel using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User gives the request id of the request to cancel using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The local dummy DC accepts the cancellation of the request.	
	1.2	The SAP DC accepts the cancellation of the request.	
	### Realised Result (Textual) ###		
	1.1	The request cancellation was delivered to DC.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.6 Testcase006_STW_DC_subsrcibe OK / NOK

Positive test / Module test			
Core function: Does a subscription to an info item on a PEID using the DC. Requests a subscription from the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User selects the PEID info item to request a subscription for using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User selects the PEID info item to request a subscription for using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The local dummy DC accepts the subscription request.	
	1.2	The SAP DC accepts the subscription request.	
	### Realised Result (Textual) ###		
	1.1	The subscription request was delivered to the DC.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.7 *Testcase007_STW_DC_getInfoItemList OK / NOK*

Positive test / Module test			
Core function: List all info items for a PEID. Creates the corresponding PEID information in the Trackway system.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User gives a PEID id to request info items for using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User gives a PEID id to request info items for using the test GUI developed by Stockway The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	Trackway DC integration module creates a WWAI representation in the Trackway system of the PEID. The info items are listed in a table in the test GUI.	
	1.2	Trackway DC integration module creates a WWAI representation in the Trackway system of the PEID. The info items are listed in a table in the test GUI.	
	### Realised Result (Textual) ###		
	1.1	The Info Items for the object where read from the DC and the corresponding Trackway representation was created.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.8 Testcase008_STW_DC_getInfoItemListExt OK / NOK

Positive test / Module test		
Core function: List all info items for a PEID. Creates the corresponding PEID information in the Trackway system.		
Pre-requisites: none		
Test ID		
### Test activity ###		
1.1	User gives a PEID id to request info items for using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
1.2	User gives a PEID id to request info items for using the test GUI developed by Stockway The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
### Expected result ###		
1.1	Trackway DC integration module creates a WWAI representation in the Trackway system of the PEID. The info items are listed in a table in the test GUI.	
1.2	Trackway DC integration module creates a WWAI representation in the Trackway system of the PEID. The info items are listed in a table in the test GUI.	
### Realised Result (Textual) ###		
1.1	The Info Items for the object where read from the DC and the corresponding Trackway representation was created.	
1.2	*Could not be performed (see reason above)*	
### Realised Result ###		
	ok?	**Error description**
1.1	OK	
1.2	NOK	Error Class: *Not performed (see reason above)*
### Overall assessment of the Test ###		
Overall Test Result	OK	
Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
Validation	Succeeded with limitation	
### Organisational Data ###		
Tester	Björn Forss, Stockway	
Test date	20.04.2006	

5.3.4.9 *Testcase009_STW_DC_getDeviceList OK / NOK*

Positive test / Module test			
Core function: Requests the DC to list all PEIDs known to the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	User selects to view PEID ids using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	User selects to view PEID ids using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	A list of all PEID ids is shown to the user.	
	1.2	A list of all PEID ids is shown to the user.	
	### Realised Result (Textual) ###		
	1.1	The request was delivered to the DC and the list of devices registered in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.10 Testcase010_STW_DC_addEventListener OK / NOK

Positive test / Module test			
Core function: Create an event listener for an event type. Does an event listener request to the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	The user gives the event type identifier using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	The user gives the event type identifier using the test GUI developed by Stockway. The GUI calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The local dummy DC accepts the listener request.	
	1.2	The SAP DC accepts the listener request.	
	### Realised Result (Textual) ###		
	1.1	The event listener request was delivered to the DC.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.11 Testcase011_STW_DC_createGroup OK / NOK

Positive test / Module test			
Core function: Request the DC to create a group.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The request is accepted by the DC. A representation of the group is created in the Trackway system.	
	1.2	The request is accepted by the DC. A representation of the group is created in the DC and in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The group creation request was delivered to the DC and the corresponding group was created in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.12 *Testcase012_STW_DC_deleteGroup OK / NOK*

Positive test / Module test			
Core function: Request the DC to delete a group.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The request is accepted by the DC. The representation of the group is removed in the Trackway system.	
	1.2	The request is accepted by the DC. The representation of the group is removed in the DC and in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The group delete request was delivered to the DC and the corresponding group was removed in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.13 *Testcase013_STW_DC_addObjectToGroup OK / NOK*

Positive test / Module test			
Core function: Adds and object to a group.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The request is accepted by the DC. The object is added to the group in the Trackway system.	
	1.2	The request is accepted by the DC. The object is added to the group in the DC and in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The addition request was delivered to the DC and the corresponding object was put in the group in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.14 *Testcase014_STW_DC_removeObject OK / NOK*

Positive test / Module test			
Core function: Remove an object from a group.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	The request is accepted by the DC. The object is removed from the group in the Trackway system.	
	1.2	The request is accepted by the DC. The object is removed from the group in the DC and in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The removal request was delivered to the DC and the corresponding object was removed from the group in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Björn Forss, Stockway
	Test date		20.04.2006

5.3.4.15 *Testcase015_STW_DC_listGroups OK / NOK*

Positive test / Module test			
Core function: List all groups in the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	A list of all groups of the DC is shown to the user.	
	1.2	A list of all groups of the DC is shown to the user.	
	### Realised Result (Textual) ###		
	1.1	The request was delivered to the DC and the list of groups registered in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.16 *Testcase016_STW_DC_listObjects OK / NOK*

Positive test / Module test			
Core function: Requests a list of all objects in a group in the DC.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	A test component calls the Trackway DC integration module which calls a local dummy DC Web Service implemented by Stockway. Trackway DC integration module handles the response from the DC.	
	1.2	A test component calls the Trackway DC integration module which calls the DC Web Service implemented by SAP. Trackway DC integration module handles the response from the DC.	
	### Expected result ###		
	1.1	A list of all objects in the group in the DC is shown to the user.	
	1.2	A list of all objects in the group in the DC is shown to the user.	
	### Realised Result (Textual) ###		
	1.1	The request was delivered to the DC and the list of objects in the group where registered in the Trackway system.	
	1.2	*Could not be performed (see reason above)*	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	NOK	Error Class: *Not performed (see reason above)*
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks	The integration part of the test could not be performed (see reason above). The Trackway representation of the test was successful and therefore the tests are successful.	
	Validation	Succeeded with limitation	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.17 *Testcase017_STW_resultCallback_WS OK / NOK*

Positive test / Module test			
Core function: Inform the result of a request using the call-back Web Service. The DC calls the call-back service provided for it.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	The DC returns with the results for a request that could not be performed directly. The DC calls the call-back web service on Trackway DC integration module.	
	### Expected result ###		
	1.1	The Trackway DC integration module updates the values for the info item the request concerned in the Trackway system.	
	### Realised Result (Textual) ###		
	1.1	The Trackway call-back service reacted correctly to the request and made the corresponding retrieve result request to the DC. The retrieved Info Item data w	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks		
	Validation	Succeeded	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.4.18 Testcase018_STW_WWAI_PEID_dataRequest OK / NOK

Positive test / Module test			
Core function: Test that WWAI communication is efficient enough for PEID data exchange.			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1	Two Trackway servers run on the same PC. One of the servers runs a test component that does constant requests to the other server. The time for the requests is measured.	
	1.2	Two Trackway servers run two PCs connected to the Internet. One of the servers runs a test component that does constant requests to the other server. The time for the requests is measured.	
	### Expected result ###		
	1.1	All request could be fulfilled and in a feasible time.	
	1.2	All request could be fulfilled and in a feasible time.	
	### Realised Result (Textual) ###		
	1.1	DEBUG: [*** TEST REUSLTS ***] Tests repeated 50000 Fastest time:40ms Slowest time:691ms Total time:3363626ms Average time:67ms The request could be performed in a reasonably amount of time. The slowest request is likely due to system swapping operations.	
	1.2	DEBUG: [*** TEST REUSLTS ***] Tests repeated 50000 Fastest time:83ms Slowest time:842ms Total time:6150402ms Average time:123ms The request could be performed in a reasonably amount of time. The slowest request is likely due to network delays.	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result	OK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks		
	Validation	Succeeded	
	### Organisational Data ###		
	Tester	Björn Forss, Stockway	
	Test date	20.04.2006	

5.3.5 Summary for parts of HUT

Even though not all test could be performed as planned, the result from the tests are considered positive. The tests performed by Stockway shows that Promise PEID data can be represented in the Trackway software and communicated using the WWAI protocol. This is valuable input to the coming IOCI related work as the Trackway system will be used as one IOCI implementation.

6 Overall summary

This deliverable reports on the progress made in establishing a foundation for testing within the PROMISE project and the initial unit testing of the current state of the art of technology that has been developed by the partners SAP, HUT and Stockway. In the next periods, this technology will be developed further, and it is probable that INDYON also will commence some middleware technology development.

The results of the tests demonstrate that the developed technologies mainly comply with the current PROMISE specifications. The diversity of the testing which results from the differences in the scope of implementation between the three partners, is in itself useful as it has already established a broader base of tests.

These tests will be useful in the future, not only as the foundation for additional tests, but also as regression tests when extended functionality will be developed according to the plans for tasks TR6.5 and TR6.6.

The recognition that there are some gaps between what was initially defined in the DoW task descriptions and what has so far been possible to develop and test, means that the partners in WP R6 can focus on how to address those omissions in conjunction with the development and further testing that is also required to fulfil tasks TR6.5 through TR6.7.

The WP R6 partners will take the responsibility of ensuring that the testing foundation that has been established in this task can be used as the basis for integrated testing of the entire PROMISE technology infrastructure, involving technology developed in WP R4, R8 and R9. This also means that it will be necessary to define additional, wider project testing tasks to meet this objective during the next project planning cycle.

Appendix A – Test case form Template

Testcase001_Fnct_Name OK / NOK

Positive test / Module test			
Core function:			
Pre-requisites: none			
	Test ID		
	### Test activity ###		
	1.1		
	1.2		
	1.3		
	### Expected result ###		
	1.1		
	1.2		
	1.3		
	### Realised Result (Textual) ###		
	1.1		
	1.2		
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	NOK	Error Class:
	1.2	OK	
	1.3	OK	
	### Overall assessment of the Test ###		
	Overall Test Result	NOK	
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks		
	Validation	Succeeded / Succeeded with limitation / Not succeeded	
	### Organisational Data ###		
	Tester	Name	
	Test date	01.04.2006	

Appendix B – Functional Test cases for modules of SAP (Updated)

B.1 Testcase111Testcase001_CONTENT_I OK

Positive test / Module test			
Core function: VALID_READ_REQUEST			
Pre-requisites: none			
	Test ID		**1114.1**
	### Test activity ###		
	1.1	Read infoItemY on a registered deviceX where deviceX is connected and infoItemY exists on deviceX	
	1.2	Read infoItemY on a registered deviceX where deviceX is not connected and infoItemY exists on deviceX	
	### Expected result ###		
	1.1	result String with result=<current value of infoItemY on deviceX>	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result (Textual) ###		
	1.1	result String with result=<current value of infoItemY on deviceX>	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.2 Testcase111Testca2_CONTENT_II OK

Positive test / Module test			
Core function: INVALID_READ_REQUEST			
Pre-requisites: none			
	Test ID		**1124.1**
	### Test activity ###		
	1.1	Read infoItemY on deviceX where deviceX is not registered (is not listed in metadata)	
	1.2	Read infoItemY on deviceX where deviceX is registered but infoItemY **doesn’t** exist on deviceX (is not listed in metadata)	
	### Expected result ###		
	1.1	result String with result=<empty>, error=<appropriate error message>	
	1.2	result String with result=<empty>, error=<appropriate error message>	
	### Realised Result (Textual) ###		
	1.1	result String with result=<empty>, error=’The targetId 'deviceX' is not associated with a registered device.’	
	1.2	result String with result=<empty>, error=’The infoItem 'infoItemY' is not available at device ‘deviceX’’	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		Requests are now checked against metadata before being sent out, so that there is no risk of invalid requests being handled incorrectly at DHL
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.3 Testcase113Testcase001_CONTENT_III OK

Positive test / Module test			
Core function: VALID_WRITE_REQUEST			
Pre-requisites: none			
	Test ID		**1134.2**
	### Test activity ###		
	1.1	Write infoItemY on a registered deviceX where deviceX is connected and infoItemY exists on deviceX	
	1.2	Write infoItemY on a registered deviceX where deviceX is not connected and infoItemY exists in deviceX	
	### Expected result ###		
	1.1	result String with result=<current value of infoItemY on deviceX>	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result (Textual) ###		
	1.1	result String with result=<current value of infoItemY on deviceX>	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.4 Testcase114Testcase001_CONTENT_IV OK

Positive test / Module test			
Core function: INVALID_WRITE_REQUEST			
Pre-requisites: none			
	Test ID		**1144.2**
	### Test activity ###		
	1.1	Write infoItemY on deviceX where deviceX is not registered (is not listed in metadata)	
	1.2	Write infoItemY on deviceX where deviceX is registered but infoItemY **doesn't** exist on deviceX (is not listed in metadata)	
	### Expected result ###		
	1.1	result String with result=<empty>, error=<appropriate error message>	
	1.2	result String with result=<empty>, error=<appropriate error message>	
	### Realised Result (Textual) ###		
	1.1	result String with result=<empty>, error='The targetId 'deviceX' is not associated with a registered device.'	
	1.2	result String with result=<empty>, error='The infoItem 'infoItemY' is not available at device 'deviceX''	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		Requests are now checked against metadata before being sent out, so that there is no risk of invalid requests being handled incorrectly at DHL
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.5 Testcase115Testcase001_CONTENT_V OK

Positive test / Module test			
Core function: PROCESS REQUESTS			
Pre-requisites: devi ceX not connected, requests are buffered for deviceX			
	Test ID		**1154.3**
	### Test activity ###		
	1.1	Connect deviceX	
	1.2	RHL: retrieve(requestId)	
	1.3		
	### Expected result ###		
	1.1	Buffered requests for deviceX are sent to DHL and processed	
	1.2	Resultstring for request with requestId	
	1.3		
	### Realised Result (Textual) ###		
	1.1	Requests are processed at DHL, results are sent to RHL	
	1.2	Resultstring for request with requestId	
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		12.04.2006

B.6 Testcase116Testcase001_CONTENT_VI OK /NOK

Positive test / Module test			
Core function: RETRIEVE RESULTS			
Pre-requisites: none			
	Test ID		**1164.4.**
	### Test activity ###		
	1.1	Retrieve result for request to deviceX where deviceX has connected after request has been issued	
	1.2	Retrieve result for request to deviceX where deviceX has not connected after request has been issued	
	1.3	Retrieve result for invalid requestId (i.e. requestId that has not been assigned to a request)	
	### Expected result ###		
	1.1	Result String with result or error message if request couldn't be processed	
	1.2	Message stating that request has not yet been processed	
	1.3	Message saying that requested is erroneous, i.e. that there has not been a request issued with this id	
	### Realised Result (Textual) ###		
	1.1	Result String with result or error message if request couldn't be processed	
	1.2	Message stating that request has not yet been processed	
	1.3	Message stating that request has not yet been processed	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	NOK	Error Class 4: not distinguishing between pending requests and non-existing requests
	### Overall assessment of the Test ###		
	Overall Test Result		NOK
	Error Class		4 (not critical)
	Remarks		For 1.3 Though this situation shouldn't occur with automatic request processing, requestIds should be checked against buffered requests in order to inform about invalid requestIds
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.7 Testcase117Testcase001_CONTENT_VII NOK

Positive test / Module test			
Core function: VALID_SUBSCRIBE_REQUEST			
Pre-requisites: none			
	Test ID		**1174.1**
	### Test activity ###		
	1.1	Subscribe to infoItemY on a registered deviceX with subscription interval= 1min, where deviceX is connected and infoItemY exists on deviceX	
	1.2	Subscribe to infoItemY on a registered deviceX with subscription interval= 1min, where deviceX is not connected and infoItemY exists on deviceX	
	### Expected result ###		
	1.1	Result String with result=<current value of infoItemY on deviceX>, new results are sent every minute	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result (Textual) ###		
	1.1	result String with result=<current value of infoItemY on deviceX>, no new results after subscription interval	
	1.2	result String with result=<empty> and requestId to retrieve result later	
	### Realised Result ###		
		ok?	**Error description**
	1.1	NOK	The request processing based on the subscription interval has not yet been properly implemented as the timing (thread) which is necessary for this task could not be implemented with EJB. A separate timing/scheduling service (Quartz, licensed under Apache 2.0 license) will be used for this purpose.
	1.2	NOK	
	### Overall assessment of the Test ###		
	Overall Test Result		NOK
	Error Class		1 (Very Strong)
	Remarks		
	Validation		Failed
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.8 Testcase111Testca8_CONTENT_VIII OK

Positive test / Module test			
Core function: INVALID_SUBSCRIBE_REQUEST			
Pre-requisites: none			
	Test ID		**1184.1**
	### Test activity ###		
	1.1	Subscribe to infoItemY on deviceX with subscription interval=1min where deviceX is not registered (is not listed in metadata)	
	1.2	Subscribe to infoItemY on deviceX with subscription interval=1min where deviceX is registered but infoItemY **doesn't** exist on deviceX (is not listed in metadata)	
	### Expected result ###		
	1.1	result String with result=<empty>, error=<appropriate error message>	
	1.2	result String with result=<empty>, error=<appropriate error message>	
	### Realised Result (Textual) ###		
	1.1	result String with result=<empty>, error='The targetId 'deviceX' is not associated with a registered device.'	
	1.2	result String with result=<empty>, error='The infoItem 'infoItemY' is not available at device 'deviceX''	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		Requests are now checked against metadata before being sent out, so that there is no risk of invalid requests being handled incorrectly at DHL
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenreich
	Test date		05.05.2006

B.9 Testcase121Testcase001_METADATA_I OK

Positive test / Module test			
Core function: REGISTRATION			
Pre-requisites: none			
	Test ID		**1294.5.**
	### Test activity ###		
	1.1	Start a device that has already been registered with the Device Manager	
	1.2	Start a device that has not yet been registered with the Device Manager and allow to connect	
	1.3	Start a device that has not yet been registered with the Device Manager and refuse connection	
	### Expected result ###		
	1.1	Device gets connected iff connectable==true in registry	
	1.2	Device gets connected; registry entry with connectable=true; metadata is set in DHL	
	1.3	Connection refused; registry entry with connectable=false; metadata is not set in DHL	
	### Realised Result (Textual) ###		
	1.1	Device gets connected iff connectable==true in registry	
	1.2	Device gets connected; registry entry with connectable=true; metadata is set in DHL	
	1.3	Connection refused; registry entry with connectable=false; metadata is not set in DHL	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3	OK	
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenereich
	Test date		05.05.2006

B.10 Testcase122Testcase001_METADATA_II OK/NOK

Positive test / Module test			
Core function: METADATA RETRIEVAL			
Pre-requisites: none			
	Test ID		**1224.7**
	### Test activity ###		
	1.1	Get list of devices registered with DHL	
	1.2	Get list of infoItems for a deviceX which is registered with DHL	
	1.3	Get list of infoItems for a deviceX which is not registered with DHL	
	### Expected result ###		
	1.1	List of devices registered with device registry	
	1.2	List of infoItems available on deviceX	
	1.3	Appropriate error message stating that there is no such device	
	### Realised Result (Textual) ###		
	1.1	List of devices registered with device registry	
	1.2	List of infoItems available on deviceX	
	1.3	Throws unhandled Exception	
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	Error Class:
	1.2	OK	
	1.3	NOK	Exception is not handled appropriately
	### Overall assessment of the Test ###		
	Overall Test Result		NOK
	Error Class		
	Remarks		Though 1.3 should not occur since only registered devices should be queried for infoItems, there should be an appropriate error message at this point.
	Validation		Succeeded with limitation
	### Organisational Data ###		
	Tester		Katrin Eisenereich
	Test date		05.05.2006

B.11 Testcase123Testcase001_METADATA_III OK

Positive test / Module test			
Core function: METADATA UPDATE			
Pre-requisites: none			
	Test ID		**1234.6**
	### Test activity ###		
	1.1	Add an infoItem key to a device	
	1.2	Remove an infoItem key from a device	
	1.3		
	### Expected result ###		
	1.1	InfoItem item gets available on device and infoItemId is added to metadata storage in DHL	
	1.2	InfoItem item gets unavailable on device and infoItemId is removed from metadata storage in DHL	
	1.3		
	### Realised Result (Textual) ###		
	1.1	InfoItem item gets available on device and infoItemId is added to metadata storage in DHL	
	1.2	InfoItem item gets unavailable on device and infoItemId is removed from metadata storage in DHL	
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	OK	
	1.2	OK	
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		OK
	Error Class		
	Remarks		Propagation of metadata updates to DHL is now taken care of. But should be reengineered do to poor separation of concerns in DHL.
	Validation		Succeeded
	### Organisational Data ###		
	Tester		Katrin Eisenereich
	Test date		05.05.2006

B.12 Testcase131Testcase001_MISCELLANEOUS_I NOK

Positive test / Module test			
Core function: METADATA RETRIEVAL			
Pre-requisites: none			
	Test ID		**1314.7**
	### Test activity ###		
	1.1	Device is available, DHL is started	
	1.2		
	1.3		
	### Expected result ###		
	1.1	Device gets connected to DHL (if it is allowed to connect) and available in the GUI	
	1.2		
	1.3		
	### Realised Result (Textual) ###		
	1.1	Device gets connected, but exception is thrown and device is not displayed in GUI	
	1.2		
	1.3		
	### Realised Result ###		
		ok?	**Error description**
	1.1	NOK	Exception at RootDeviceListenerError Class:
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		NOK
	Error Class		1 (very strong)
	Remarks		DHL needs be checked for error.
	Validation		Failed
	### Organisational Data ###		
	Tester		Katrin Eisenereich
	Test date		05.05.2006

Non-functional test cases

B.13 Testcase201Testcase001_Scalability_Pretest

Positive test / Module test			
Core function: Scalability Pretest			
Pre-requisites: none			
	Test ID		**2.15.0**
	### Test activity ###		
	1.1	Raise quantity of UPnP devices. Find maximum number of detected devices.	
	1.2		
	1.3		
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	-	
	1.3	-	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2	-	
	1.3	-	
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006

B.14 Testcase202Testcase001_Scalability_I

Positive test / Module test			
Core function: Scalability I			
Pre-requisites: none			
	Test ID	**2.25.1**	
	### Test activity ###		
	1.1	Create 10 UPnP devices. Send one request to every device. Log performance.	
	1.2	Create 100 UPnP devices. Send one request to every device. Log performance.	
	1.3	Raise number of UPnP devices. Send one request to every device. Log performance. Find limitation.	
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	TBD	
	1.3	TBD	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class	1 (very strong), 2 (strong), 3 (medium), 4 (not critical)	
	Remarks		
	Validation	Succeeded / Succeeded with limitation / Not succeeded	
	### Organisational Data ###		
	Tester	Bernhard Wolf	
	Test date	18.04.2006	

B.15 Testcase203Testcase001_Scalability_II

Positive test / Module test			
Core function: Scalability II			
Pre-requisites: none			
	Test ID		**5.2.3**
	### Test activity ###		
	1.1	Create 10 requests for one UPnP device. Log performance.	
	1.2	Create 100 requests for one UPnP device. Log performance.	
	1.3	Create 1000 requests for one UPnP device. Log performance.	
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	TBD	
	1.3	TBD	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2		
	1.3		
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006

B.16 Testcase204Testcase001_Scalability_III

Positive test / Module test			
Core function: Scalability III			
Pre-requisites: none			
	Test ID		**2.45.3**
	### Test activity ###		
	1.1	Create multiple requests for multiple UPnP devices. Log performance.	
	1.2	Variation possible.	
	1.3		
	### Expected result ###		
	1.1	-	
	1.2	-	
	1.3	-	
	### Realised Result (Textual) ###		
	1.1	TBD	
	1.2	-	
	1.3	-	
	### Realised Result ###		
		ok?	**Error description**
	1.1		
	1.2	-	
	1.3	-	
	### Overall assessment of the Test ###		
	Overall Test Result		
	Error Class		1 (very strong), 2 (strong), 3 (medium), 4 (not critical)
	Remarks		
	Validation		Succeeded / Succeeded with limitation / Not succeeded
	### Organisational Data ###		
	Tester		Bernhard Wolf
	Test date		18.04.2006