



Data, Information and Process Integration
with Semantic Web Services

DIP

Data, Information and Process Integration with Semantic Web Services

FP6 - 507483

Deliverable

WP 8: Case Study B2B in Telecommunications

D8.4

Case Study implementation prototype V1

Sam Watkins

Alistair Duke

Marc Richardson

Bernhard Schreder

Alexander Wahler

Ruben Verlinden

Thomas Haselwanter

July 10th, 2004



SUMMARY

This deliverable describes the implementation prototype for the Assurance Integration use case as part of the B2B in Telecommunications Case Study, previously discussed in [3]. The document includes detailed installation guidelines and documentation. The prototype has been built using the WSMX core architecture but also includes additional components.

WSMX [5] is an execution environment which enables discovery, selection, mediation, invocation and interoperation of the Semantic Web Services (SWS). WSMX is based on the conceptual model provided by WSMO [4], being at the same time a reference implementation of it. It is the scope of WSMX to provide a test bed for WSMO and to prove its viability as a mean to achieve dynamic interoperability of Semantic Web Services. For this Case Study prototype the open source implementation of the system has been chosen, which is used as a reference implementation of the DIP architecture.

Additional external components have been integrated with the core WSMX architecture to realise the Assurance Integration use case. These components include the Front-end web application and the back-end Semantic Web Services.

An instance of the prototype can also be accessed online at <http://wsmx.deri.org>, while the Front-end for this instance can be found at <http://starpc27.vub.ac.be:8180/frontend>.

This deliverable contributes to the Open Source Semantic Web Service Architecture and is relevant to all components developed in DIP. The target audiences of this deliverable are developers and IT experts who are interested to test and evaluate the prototype.

Disclaimer: The DIP Consortium is proprietary. There is no warranty for the accuracy or completeness of the information, text, graphics, links or other items contained within this material. This document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.

Document Information

IST Project Number	FP6 – 507483	Acronym	DIP
Full title	Data, Information, and Process Integration with Semantic Web Services		
Project URL	http://dip.semanticweb.org		
Document URL			
EU Project officer	Kai Tullius		

Deliverable	Number	8.4	Title	Case Study implementation prototype V1
Work package	Number	8	Title	Case Study B2B in Telecommunications

Date of delivery	Contractual	M 18	Actual	29-Jul-05
Status	version. 0.04		final	<input checked="" type="checkbox"/>
Nature	Prototype <input checked="" type="checkbox"/> Report <input type="checkbox"/> Dissemination <input type="checkbox"/> Ontology <input type="checkbox"/>			
Dissemination Level	Public <input checked="" type="checkbox"/> Consortium <input type="checkbox"/>			





Authors (Partner)	Bernhard Schreder (NIWA), Alexander Wahler (NIWA), Sam Watkins (BT), Alistair Duke (BT), Marc Richardson (BT), Ruben Verlinden (VUB), Thomas Haselwanter (UIBK)			
Responsible Author	Bernhard Schreder		Email	bernhard.schreder@niwa.at
	Partner	NIWA	Phone	+4313195843-13






Abstract (for dissemination)	This deliverables provides an overview of the first version of the B2B platform prototype including detailed installation guidelines and documentation.
Keywords	Execution environment, B2B platform, eCo, architecture, semantic web, semantic web services, prototype




Version Log			
Issue Date	Rev No.	Author	Change
13-jul-05	001	Bernhard Schreder	Initial version
20-jul-05	002	Bernhard Schreder	Updates to all sections
21-jul-05	003	Bernhard	Integrated comments from WP8 partners

		Schreder	
28-jul-05	004	Bernhard Schreder	Changes from reviewer's comments

Project Consortium Information

Partner	Acronym	Contact
National University of Ireland Galway	NUIG 	Prof. Dr. Christoph Bussler Digital Enterprise Research Institute (DERI) National University of Ireland, Galway Galway Ireland Email: chris.bussler@deri.org Tel: +353 91 512460
Fundacion De La Innovacion.Bankinter	Bankinter 	Monica Martinez Montes Fundacion de la Innovacion. BankInter Paseo Castellana, 29 28046 Madrid, Spain Email: mmtnez@bankinter.es Tel: 916234238
Berlecon Research GmbH	Berlecon 	Dr. Thorsten Wichmann Berlecon Research GmbH Oranienburger Str. 32 10117 Berlin, Germany Email: tw@berlecon.de Tel: +49 30 2852960
British Telecommunications Plc.	BT 	Dr John Davies BT Exact (Orion Floor 5 pp12) Adastral Park Martlesham Ipswich IP5 3RE, United Kingdom Email: john.nj.davies@bt.com Tel: +44 1473 609583
Swiss Federal Institute of Technology, Lausanne	EPFL 	Prof. Karl Aberer Distributed Information Systems Laboratory École Polytechnique Fédérale de Lausanne Bât. PSE-A 1015 Lausanne, Switzerland Email : Karl.Aberer@epfl.ch Tel: +41 21 693 4679
Essex County Council	Essex 	Mary Rowlett, Essex County Council PO Box 11, County Hall, Duke Street Chelmsford, Essex, CM1 1LX United Kingdom. Email: maryr@essexcc.gov.uk Tel: +44 (0)1245 436524
Forschungszentrum Informatik	FZI 	Andreas Abecker Forschungszentrum Informatik Haid-und-Neu Strasse 10-14 76131 Karlsruhe Germany Email: abecker@fzi.de Tel: +49 721 9654 0
Partner	Acronym	Contact

<p>Institut für Informatik, Leopold-Franzens Universität Innsbruck</p>	<p>UIBK </p>	<p>Prof. Dieter Fensel Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck, Austria Email: dieter.fensel@deri.org Tel: +43 512 5076485</p>
<p>ILOG SA</p>	<p>ILOG  Changing the rules of business</p>	<p>Christian de Sainte Marie 9 Rue de Verdun, 94253 Gentilly, France Email: csma@ilog.fr Tel: +33 1 49082981</p>
<p>inubit AG</p>	<p>Inubit  the integration experts</p>	<p>Torsten Schmale inubit AG Lützowstraße 105-106 D-10785 Berlin Germany Email: ts@inubit.com Tel: +49 30726112 0</p>
<p>Intelligent Software Components, S.A.</p>	<p>iSOCO </p>	<p>Dr. V. Richard Benjamins, Director R&D Intelligent Software Components, S.A. Pedro de Valdivia 10 28006 Madrid, Spain Email: rbenjamins@isoco.com Tel. +34 913 349 797</p>
<p>NIWA WEB Solutions</p>	<p>NIWA </p>	<p>Alexander Wahler NIWA WEB Solutions Niederacher & Wahler OEG Kirchengasse 13/1a A-1070 Wien Email: wahler@niwa.at Tel: +43(0)1 3195843-11 </p>
<p>The Open University</p>	<p>OU  The Open University</p>	<p>Dr. John Domingue Knowledge Media Institute The Open University, Walton Hall Milton Keynes, MK7 6AA United Kingdom Email: j.b.domingue@open.ac.uk Tel.: +44 1908 655014</p>
<p>SAP AG</p>	<p>SAP </p>	<p>Dr. Elmar Dörner SAP Research, CEC Karlsruhe SAP AG Vincenz-Priessnitz-Str. 1 76131 Karlsruhe, Germany Email: elmar.dorner@sap.com Tel: +49 721 6902 31</p>

<p>Sirma AI Ltd.</p>	<p>Sirma</p>  <p>Ontotext Knowledge and Language Engineering Lab of Sirma</p>	<p>Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD Office Express IT Centre, 3rd Floor 135 Tzarigradsko Chausse Sofia 1784, Bulgaria Email: atanas.kiryakov@sirma.bg Tel.: +359 2 9768 303</p>
<p>Unicorn Solution Ltd.</p>	<p>Unicorn</p> 	<p>Jeff Eisenberg Unicorn Solutions Ltd, Malcha Technology Park 1 Jerusalem 96951 Israel Email: Jeff.Eisenberg@unicorn.com Tel.: +972 2 6491111</p>
<p>Vrije Universiteit Brussel</p>	<p>VUB</p>  <p>Vrije Universiteit Brussel</p>	<p>Pieter De Leenheer Starlab- VUB Vrije Universiteit Brussel Pleinlaan 2, G-10 1050 Brussel ,Belgium Email: Pieter.De.Leenheer@vub.ac.be Tel.: +32 (0) 2 629 3749</p>

LIST OF KEY WORDS/ABBREVIATIONS

API	Application Programming Interface
B2B	Business to Business
eCo	BT Wholesale B2B OSS Gateway, run by BT Wholesale [9]
GUI	Graphical User Interface
OSS	Operational Support System
SOPHIE	Semantic choreography engine
SWS	Semantic Web Services
UI	User Interface
WSML	Web Service Modelling Language
WSMO	Web Services Modelling Ontology
WSMX	Web Services Execution Environment
XML	eXtensible Mark-up Language

TABLE OF CONTENTS

SUMMARY	I
LIST OF KEY WORDS/ABBREVIATIONS	VII
TABLE OF CONTENTS	VIII
1 INTRODUCTION	1
2 ARCHITECTURE PROTOTYPE FACT SHEET	1
2.1 Description of purpose, scope and functionality	1
2.1.1 Purpose and Scope.....	1
2.1.2 Functionality.....	2
2.2 Installation Guidelines.....	3
2.2.1 Software Prerequisites	3
2.2.2 WSMX Installation Instructions.....	4
2.2.3 Front-end Installation Instructions	4
2.2.4 Web Services Installation Instructions	5
2.3 Type of API.....	5
2.4 Licence information.....	6
2.5 How to use the prototype.....	6
2.6 Roadmap for future plans	7
3 ARCHITECTURE PROTOTYPE DOCUMENTATION	7
3.1 Front-end GUI	7
3.2 Additional documentation	11
4 CONCLUSION	11
REFERENCES	12

LIST OF FIGURES

Figure 1: WSMX for the Telecommunications B2B Integration Platform.....	3
Figure 2: The Service Provider’s Test Request UI.....	8
Figure 3: Follow up an existing request	9
Figure 4: Send Request Information.....	10

LIST OF TABLES

Table 1: used WSMX components.....	2
Table 2: Software Prerequisites.....	3



1 INTRODUCTION

This deliverable describes the implementation prototype for the Assurance Integration use case as part of the B2B in Telecommunications Case Study, previously discussed in [3]. The document includes detailed installation guidelines and documentation. The prototype has been built using the WSMX core architecture but also includes additional components.

As a prototype deliverable, this document is structured as follows: Section 2 consists of the Architecture Prototype Fact Sheet, which contains all the relevant information concerning the prototype, i.e. installation guidelines. Section 3 contains a description of the Front-end GUI, as well as the links to additional documentation relevant for the prototype implementation. Finally, the conclusion summarises the status of the prototype architecture and briefly explains how the prototype can be demonstrated.

2 ARCHITECTURE PROTOTYPE FACT SHEET

The contact person for the architecture prototype is Bernhard Schreder (bernhard.schreder@niwa.at)

2.1 Description of purpose, scope and functionality

2.1.1 Purpose and Scope

Assurance Integration is a use case of the Telecommunications B2B Integration Platform which is discussed in detail in DIP D8.3 [3].

The prototype illustrates how the heterogeneous systems and messages of a Trading Partner can be more easily integrated through the use of Semantic Web Services. The prototype demonstrates Test Requests being generated by a Trading Partner OSS and communicated to BT Wholesale's eCo platform which then carries out the test and responds with the result.

The prototype works by showing the progress of such messages from the Trading Partner, through the WSMX components and hence through adapters as well as mediators, and finally the response of the back-end web services to that test request.

As the Trading Partner does not use the same process flow as the Eco platform, SOPHIE [8] is used to handle multiple message choreographies and mediate between the message exchange formats of the Trading Partners and BT Wholesale's Eco platform.

Hence, the scope of this prototype includes:

- Adaptation of native messages, data mediation and semantic descriptions of choreography.
- Process and component management
- Implementation of a DIP architecture as a Semantically Enhanced Service Orientated Architecture.
- Process mediation, carried out by the choreography engine, in order to integrate the differing message exchange patterns of the two business partners: BT Wholesale and its Trading Partner

The open architecture allows additional DIP components to be added as they become available. In particular, the process mediation component will replace SOPHIE once it becomes available.

The user of the prototype is taken through the process of raising a Test Request on a Trading Partner OSS which is then sent, via WSMX, to the eCo platform. Status messages indicate the progress of the request (and response) as it passes through the various adaptation and mediation steps.

2.1.2 Functionality

The prototype implementation of the B2B Integration Platform for the Assurance Integration use case requires a subset of the currently available WSMX Components. The components used for the prototype can be seen in Table 1.

Table 1: used WSMX components

Component name
Adapter
CommunicationManager
DataMediator
Parser
ResourceManager
Invoker

These components are used for a modified version of the execution semantics accessed by the “achieveGoal” entry point into WSMX. Execution semantics, or operational semantics, is the formal definition of the operational behavior of a system. It describes in a formal, unambiguous language how the system behaves. Because the meaning of the system (to the outside world) consists of its behavior, this formal definition is called ‘execution semantics’. Functionality provided by the WSMX system as a whole can be described in terms of its entry points - the standardized interfaces of the system enabling communication with any external entities requesting services from the system. Entry points and corresponding execution semantics are further explained in detail in [1], respectively in [6]. Figure 1 shows the version of WSMX being used for the prototype:

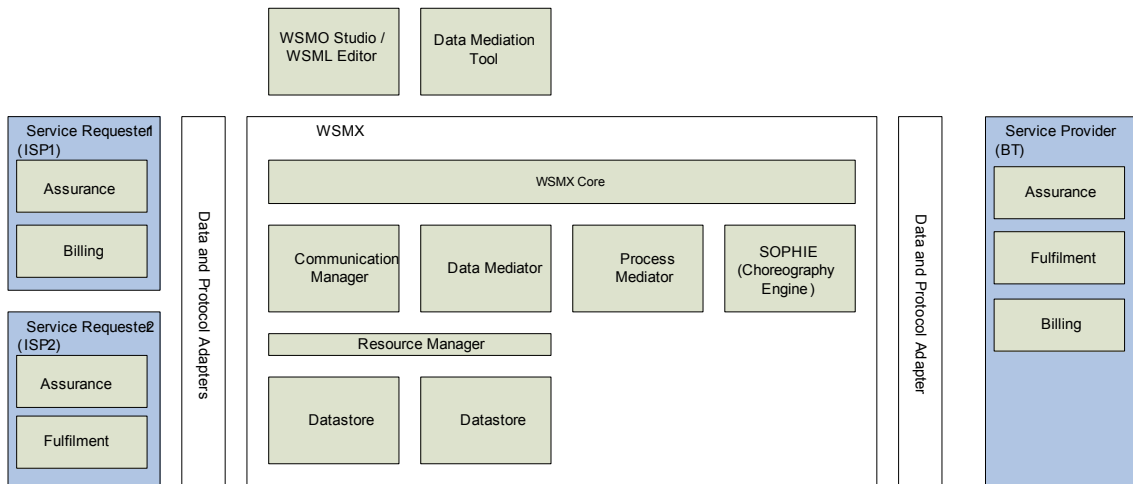


Figure 1: WSMX for the Telecommunications B2B Integration Platform

The complete description of the functionalities for the prototype are explained in [3].

2.2 Installation Guidelines

The following sections explain the software dependencies as well as step-by-step guidelines to set up the platform prototype, respectively its diverse components. The prototype itself uses several software packages which are available for download on <http://starp27.vub.ac.be:8180/frontend>. The WSMX server and the components used for the prototype can be downloaded from <http://sourceforge.net/projects/wsmx>. The third-party software needed for the prototype has to be downloaded and installed separately, as detailed in the next section.

2.2.1 Software Prerequisites

Table 2 shows the necessary third-party software and libraries to set up the platform prototype. While the prototype has been tested with the versions of the third-party products mentioned below, other versions will certainly work as well. (e.g. it is possible to use any servlet engine which implements the version 2.2 or greater of the servlet API.)

Table 2: Software Prerequisites

Third-party software and libraries	Download from
Tomcat servlet engine 5.5.9	http://jakarta.apache.org/tomcat/index.html
Axis 1.2.1	http://ws.apache.org/axis/index.html
Apache Commons	http://jakarta.apache.org/commons
JavaBeans Activation Framework 1.0.2	http://java.sun.com/products/javabeans/glasgow/jaf.html
Java Mail API 1.3.2	http://java.sun.com/products/javamail/

JDK 5.0	http://java.sun.com/j2se/1.5.0/download.jsp
JINI Technology Starter Kit (v2.0.x)	http://starterkit.jini.org/downloads
Inca X & JavaSpaces Starter Kit v3.5	http://www.incax.com
MySQL Database Server 4.1	http://dev.mysql.com/downloads/mysql/4.1.html

2.2.2 WSMX Installation Instructions

A WSMX server (version 0.2) is included with the release of the platform prototype. This version of WSMX includes all the components, as needed by the B2B Integration platform. (see Table 1 for a list of the included components)

Detailed installation guidelines for WSMX are described in [2]. In addition to the steps mentioned in that deliverable, a new ant build target has been added to the build.xml in the root directory of the WSMX server. This target (called “deploy_b2b_components”) deploys all components to the systemcodebase. (The notion of a WSMX server’s systemcodebase and the associated kernel configuration are explained in section 2.3.3 of the DIP Architecture Prototype v1 deliverable [2].)

2.2.3 Front-end Installation Instructions

The Front-end web application is used to notify the prototype user about the status of the diverse components of the B2B Integration Platform. It consists of the Service Provider GUI (specified in [3]), and visualises the current status of the WSMX server and the back-end web services.

To set up the web application, follow the steps detailed below:

- Set up the Tomcat servlet engine, following the online instructions for your platform at <http://jakarta.apache.org/tomcat/tomcat-5.5-doc/setup.html>
- Set up Axis to deploy the web services, following the online instructions for your platform and chosen application server or servlet engine at <http://ws.apache.org/axis/java/install.html>. Additional dependencies for Axis include the JavaBeans Activation Framework and the Java Mail API.
- Set up the MySQL server, following the online instructions at <http://dev.mysql.com/doc/>. In addition a database and a database user have to be created, using the following SQL statements:

```
CREATE DATABASE dip;
GRANT ALL ON dip.* TO dip@% IDENTIFIED BY 'dip';
```

- Create the tables needed by the Front-end. For this purpose a sql script has been provided with the release. The script – called “prototype.sql” – can be found in the root directory of the visualisation package. Execute the script by using the following command (this can vary according to your platform, and your installation of the MySQL server):

```
mysql -u dip -p -h localhost dip < prototype.sql
```

- Finally extract the web application itself from the archive supplied with the deliverable software on cd, and copy the “frontend” directory to the “webapps” directory of your Tomcat installation. Restart the Tomcat server for the changes to take effect.

Following these steps, point your browser to <http://localhost:8080/frontend> to start up the Front-end GUI.

2.2.4 Web Services Installation Instructions

Several web services are supplied with the first version of the prototype platform. These web services can be deployed on the same Tomcat instance set up during the previous step – or they can run on another servlet engine or application server of your choice.

Some of the web services which are used for the platform prototype need to access the same database which was set up in the previous sections. Therefore the steps to set up a MySQL server and the corresponding database and tables have to be completed, before the relevant web services can be accessed.

To deploy the web services, follow the steps below for every single web service:

- Copy the web service’s compiled class files to `webapps/axis/WEB-INF/classes` directory, and the corresponding deployment descriptor (`deploy.wsdd`) to the same directory
- Change to the directory containing the class file and use the Axis AdminService to deploy the service using one of the commands shown in Listing 1, depending on your platform.

On Windows

```
java -cp %AXISCLASSPATH% org.apache.axis.client.AdminClient  
-lhttp://localhost:8080/axis/services/AdminService deploy.wsdd
```

On UNIX

```
java -cp $AXISCLASSPATH org.apache.axis.client.AdminClient  
-lhttp://localhost:8080/axis/services/AdminService deploy.wsdd
```

Listing 1: Deploying the web services with Axis

Additional information on how to install and deploy new web services can be found in the Axis installation guidelines on <http://ws.apache.org/axis/java/install.html>

Finally the web services have to be registered with the running WSMX server, so the server actually knows how to invoke the requested services. (There is no discovery component in this version of the Case Study prototype, so the CommunicationManager of the WSMX server always invokes the same Semantic Web Service which is able to fulfil a request.). Details on the registration of a web service with WSMX using the Resource Manager interface can be found in [1].

2.3 Type of API

The WSMX integration API has been used for the component development and their integration with the WSMX core architecture (binaries and documentation are available at <http://sourceforge.net/projects/wsmx>).

The platform prototype itself, respectively the Service Provider front-end, uses the “achieveGoal” entry point for WSMX (entry points and the corresponding branches of execution semantics are explained in [1]).

2.4 Licence information

This Case Study prototype uses the same licensing as the underlying DIP architecture components. WSMX itself uses the GNU General Public Licence¹.

The third party software components and libraries included in the current WSMX release are using diverse licences, all of which are explained in the current release of WSMX, as well as in [2].

More detailed information about the licensing of DIP components are provided in [7].

2.5 How to use the prototype

In section 2.1 the scope and functionalities of the platform prototype have been described. The following sample scenario can be followed through to learn about the use of the prototype itself.

All steps of the scenario have been visualised, using the Front-end web application. The Front-end itself is documented in section 3.1. The steps below detail the scenario for the “Assurance Test” use case:

- Create a new Test Request: The Service Provider’s customer enters details via a form on the ticketing system. The values correspond to the Service Provider’s XML schema for request messages. After creating a new trouble ticket, the current request status is displayed in a new window.

The displayed status is updated when the message is received. Generated trouble tickets are stored in a database for future reference and editing.

- The new test request message is received by the platform, invoking the “achieveGoal” entry point.

The passage of the message through the different components of the WSMX server is shown, i.e. adaptation, data mediation etc.

- Finally the message arrives at the back-end web services, visualised through a separate status window, which is notified of the arrival.

After automatically determining whether a request is valid, the Operator opens the request and either chooses to accept it or to reject it.

- If accepted, the request is persistently stored by the web service, and a test is commenced. If rejected, a suitable response is sent back to the Service Provider’s Front-end.
- The Front-end status window is updated to display the result of the test request (either accepted or rejected)

¹ <http://www.opensource.org/licenses/gpl-license.php>

- After a short time delay to simulate the actual occurrence of a network test, the result of the test arrives back at the web service. The test result message is returned to the Service Provider.
- The acceptance message is combined with the result message (as the Service Provider expects to receive only one reply from BT wholesale) and sent on to the Service Provider. This merging of messages is done by SOPHIE.
- The Service Provider's Front-end updates its status window to display the result of the test.

2.6 Roadmap for future plans

The prototype delivered in D8.4 is intended to be a proof of concept for the use of Semantic Web Services and specifically WSMO in carryout out integration between the Operational Support Systems of BT and its Trading Partners.

The scope of the use case for the prototype deliverable D8.5 will be widened. This will include:

- The representation of a much larger number of services and capabilities from a variety of sources including those hosted by BT and its partners but also those of third parties
- The consolidation of services as a service/contract catalogue representation for BT's customer. Customers will be able to order and maintain bundled offerings.
- The consolidation of business rules expressed by the service providers allowing the service/contract catalogue representation

The prototype will place wider requirements upon the DIP technical workpackages and in particular upon the Composition and Discovery related tasks.

The detailed requirements and plans for D8.5 will be presented in an internal deliverable due at M24.

3 ARCHITECTURE PROTOTYPE DOCUMENTATION

3.1 Front-end GUI

The Service Provider's Front-end enables a Customer to submit a Trouble Ticket, requesting the test of a product bought from the Provider. The front-end also provides facilities to follow up on the status of the Trouble Ticket and to learn about the underlying technologies involved.

Figure 2 shows the Test Request Submission page, with the Customer details to be provided on the left and the status window of the test progress on the right hand side.

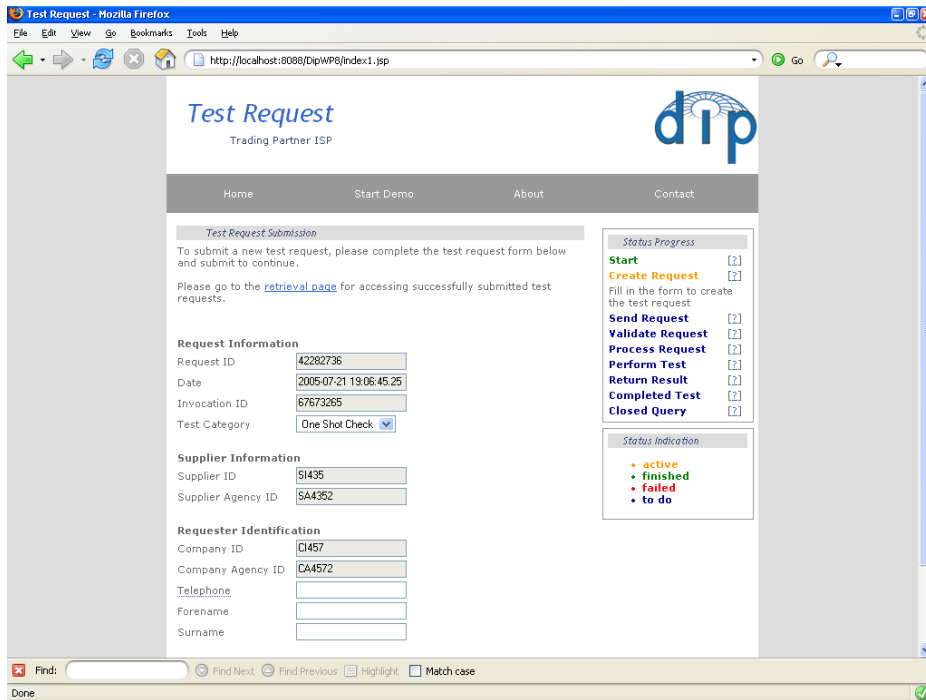


Figure 2: The Service Provider’s Test Request UI

Figure 3 shows how to select an existing Trouble Ticket and follow its current status. On the right it also shows how the user is informed of the different steps and technologies involved. First there is a very short description always visible in the status box. Second, when hovering over a certain status a more detailed description is provided in a separate box.

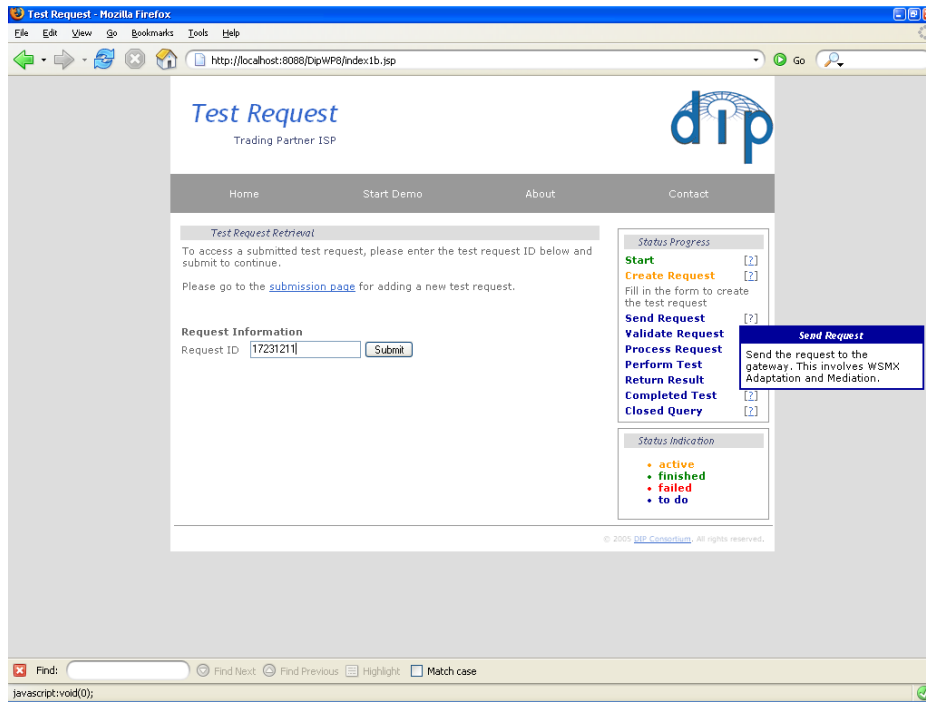


Figure 3: Follow up an existing request

Figure 4 shows the information the user sees when the test request is being sent. The status information on the right is updated to the current status. The main content panel shows the user extra information on the steps and technologies involved. In this particular case the user can see more information on WSMX Adaptation and Mediation.

Other screens in the Front-end provide a similar layout to inform the user of the current status of the request, the technologies involved and the details of these technologies.

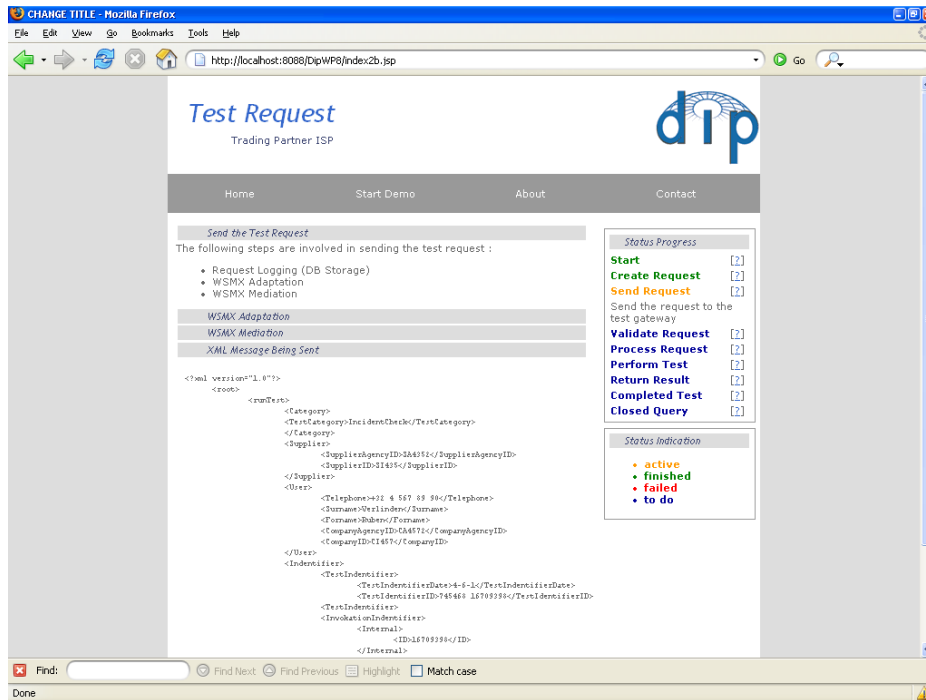


Figure 4: Send Request Information

Listing 2 shows a sample Test Request message in XML produced by the Front-end, which is based on the XML Schema used by one specific Service Provider. This message is adapted by the WSMX adapter framework and thus converted to WSML, before being handed over to the diverse WSMX components.

```
<?xml version="1.0"?>
<root><runTest>
  <Category>
    <TestCategory>IncidentCheck</TestCategory>
  </Category>
  <Supplier>
    <SupplierAgencyID>SI4352</SupplierAgencyID>
    <SupplierID>SI435</SupplierID>
  </Supplier>
  <User>
    <Telephone>0183739334</Telephone>
    <Surname>Richardson</Surname>
    <Forname>Marc</Forname>
    <CompanyAgencyID>CI4572</CompanyAgencyID>
    <CompanyID>CI457</CompanyID>
  </User>
  <Indentifier>
    <TestIndentifier>
      <TestIndentifierDate>05-05-
2004</TestIndentifierDate>
      <TestIndentifierID>000124</TestIndentifierID>
    </TestIndentifier>
    <InvokationIndentifier>
      <Internal>
        <ID>000456</ID>
      </Internal>
    </InvokationIndentifier>
  </Indentifier>
</runTest>
</root>
```

```
        </Internal>
      </InvokationIdentifier>
    </Identifier>
  </runTest>
</root>
```

Listing 2: example for a Service Provider's Test Request

3.2 Additional documentation

The current documentation for the WSMX Integration API is included in the relevant download on the WSMX project at SourceForge².

A detailed description of the prototype architecture, the diverse entry points and the correlating execution semantics is given in deliverable D6.5 [1]. The basis for this prototype implementation is the current release of the WSMX server, as detailed in deliverable D6.7 [2].

4 CONCLUSION

This deliverable provides an overview of the first version of the implementation prototype for the B2B integration platform, using the DIP architecture. The prototype has demonstrated how to design and produce data adapters and data mediators for effective use in the Telecommunications B2B Integration scenario. An instance of the prototype can be accessed online at <http://wsmx.deri.org>.

For demonstration purposes a local installation - set up according to the installation guidelines described in this deliverable - should be used, following the sample scenario discussed in section 2.5. The Front-end visualisation supplied with this version of the prototype illustrates the different steps and status changes of the prototype's components and demonstrates how a real world Service Provider could interact with the B2B Integration Platform.

² <http://sourceforge.net/projects/wsmx>

REFERENCES

- [1] Zaremba, M et al. (2005): *D6.5: DIP Revised architecture*, DIP Project, WP6 Interoperability and Architecture. <http://dip.semanticweb.org>
- [2] Haselwanter, T.; Schreder, B.; Wahler, A.; Zaremba, M. (2005): *D6.7: Architecture Prototype VI*, DIP Project, WP6 Interoperability and Architecture. <http://dip.semanticweb.org>
- [3] Watkins, S. et al. (2005): *D8.3: Prototype Platform Design*, DIP Project, WP8 Case Study B2B in Telecommunications. <http://dip.semanticweb.org>
- [4] Roman, D.; Lausen, H.; Keller, U. (2005): “*D2v1.2 Web Service Modelling Ontology (WSMO)*”, WSMO Working Draft 13 April 2005. <http://www.wsmo.org/TR/d2/v1.2/>
- [5] Cimpian, E.; Vitvar, T.; Zaremba, M. (2005): “*D13.0v0.2 Overview and Scope of WSMX*”, WSMX Working Draft 23 February 2005. <http://www.wsmo.org/TR/d13/d13.0/v0.2/>
- [6] Zaremba, M.; Oren, E. (2005): “*D13.2v0.2 WSMX Execution Semantics*”, WSMX Working Draft 14 July 2005. <http://www.wsmo.org/TR/d13/d13.2/v0.2/>
- [7] De Saint Marie, C. (2005): *D13.2 “Analysis of the appropriate open-source licensing schemata, including those used for related WS and B2B standards*, DIP Project, WP6 Interoperability and Architecture. <http://dip.semanticweb.org>
- [8] Arroyo, S. (2005): “*SOPHIE - Modeling Choreography: Prospects for Applications to Learning Objects*”, http://sigrlo.org/newsletter/Arroyo_1_2_Draft_Version.pdf/, April 2005
- [9] Duke, A. et al. (2004): “*D8.2: Platform Requirements Specification*”, DIP Project, WP8 Case Study B2B in Telecommunications. <http://dip.semanticweb.org>