




---

## ADMIRE – Development & deployment report for USMT V0 and V1: Capabilities of USMT V1

---

<b>Project Title</b>	ADMIRE
<b>Document Title</b>	ADMIRE – Development & deployment report for USMT V0 and V1: Capabilities of USMT V1
<b>Deliverable Number</b>	D4.1
<b>Authorship</b>	S. van den Berghe and work package partners
<b>Document Filename</b>	ADMIRE-D4.1-DevelopmentandDeploymentReport.doc
<b>Document Version</b>	1.0
<b>Distribution Classification</b>	Internal
<b>Distribution List</b>	<i>ADMIRE Project Team</i>
<b>Approval List</b>	<i>Jano van Hemert, Project Manager, Executive Board</i>

### Document History

<i>Personnel</i>	<i>Date</i>	<i>Comment</i>	<i>Version</i>
D. Snelling	2008/07/08	First draft	0.1
SvdB	2008/08/08	Added initial input from UEDIN and UPM	0.2
SvdB	2008/08/13	Input from MD and VL	0.3
D. Snelling	2008/08/14	Input from DS	0.4
SvdB	2008/08/19	Version for review	0.5
D. Snelling	2008/08/27	Reviewed	0.6
R.Baxter	2008/08/28	Approved	1.0

## Contents

<b>Contents .....</b>	<b>1</b>
<b>Executive Summary.....</b>	<b>2</b>
<b>A note on work package numbering .....</b>	<b>2</b>
<b>Related documents.....</b>	<b>2</b>
<b>1 WP 4: Summary: Project Month 6 .....</b>	<b>4</b>
<b>1.1 Products delivered to other work packages .....</b>	<b>4</b>
<b>2 Progress against Planned Activities .....</b>	<b>5</b>
<b>2.1 Activity 4.1: OGSA-DAI Integration.....</b>	<b>5</b>
<b>2.1.1 Planned activity.....</b>	<b>5</b>
<b>2.1.2 Actual activity .....</b>	<b>5</b>
<b>2.1.3 Deviations from plan .....</b>	<b>7</b>
<b>2.2 Activity 4.7: General Enhancement to USMT .....</b>	<b>7</b>
<b>2.2.1 Planned activity.....</b>	<b>7</b>
<b>2.2.2 Actual activity .....</b>	<b>8</b>
<b>2.2.3 Deviations from plan .....</b>	<b>10</b>
<b>2.3 Other Activities.....</b>	<b>10</b>
<b>3 Risks and Issues .....</b>	<b>10</b>
<b>3.1 Project issues.....</b>	<b>10</b>
<b>3.2 Significant problem reports.....</b>	<b>10</b>
<b>3.3 New risks .....</b>	<b>11</b>
<b>4 Plans for next Period .....</b>	<b>11</b>
<b>4.1 Activity 4.1: OGSA-DAI Integration.....</b>	<b>11</b>
<b>4.2 Activity 4.2: Enhanced Monitoring.....</b>	<b>12</b>
<b>4.3 Activity 4.3: Semantic Registry Integration.....</b>	<b>12</b>
<b>4.4 Activity 4.4: Semantic Provisioning Services Integration.....</b>	<b>13</b>
<b>4.5 Activity 4.6: OGSA-DQP Integration.....</b>	<b>13</b>
<b>4.6 Activity 4.7: General Enhancement to USMT .....</b>	<b>13</b>
<b>4.7 Activity 4.8: Standards Alignment (FLE, UVIE, UEDIN, UPM) .....</b>	<b>13</b>
<b>Appendix A WP 4 overall timeline.....</b>	<b>15</b>
<b>Appendix B WP 4 deliverable schedule .....</b>	<b>16</b>

## Executive Summary

This document is ADMIRE deliverable D4.1 and describes progress on Work Package 4 during months 1 to 6 of the ADMIRE project.

At project month 6 ADMIRE WP4 is essentially on target against original plans, having successfully deployed the initial V0 version of USMT (WP4.1 complete) and being almost ready to deploy the first (V1) ADMIRE-enhanced version of USMT (WP4.3) including a port of OGSA-DAI middleware to USMT (WP4.2). WP4.2 and WP4.3 are slightly delayed as familiarisation with the new technologies introduced by USMT took longer than anticipated. As this delay is not yet on the critical path of other work packages is not expected to have a significant impact on the progress of the project.

### A note on work package numbering

Sub-work packages within WP4 are numbered according to the list of steps described on page 51 of the Description of Work [1] and reproduced (in slightly corrected form) in Appendix A. Thus, Step 4:

4.4	M5-7	Start integrating available DMI platform services into the USMT.	Ongoing
-----	------	--	---------

is numbered in these reports as ‘WP4.4’.

The Activity numbers used in the Description of Work unfortunately differ from the work breakdown structure defined by this scheme. The Activity descriptions are used in this report but the numbering scheme has been dropped.

### Related documents

- [1] ADMIRE Annex 1: Description of Work, <http://www.admire-project.eu/trac/browser/projman/plans/ADMIRE-Annex1-DoW.doc>
- [2] ADMIRE Risks and Issues Log, July 2008,
- [3] <http://www.admire-project.eu/trac/browser/projman/plans/ADMIRE-risksIssues.doc>
- [4] The OGSA-DAI Project <http://www.ogsadai.org.uk/>
- [5] OGSA-DAI Component Registry: <http://www.ogsadai.org.uk/contribs/registry.php>
- [6] JAX-WS reference Implementation: <https://jax-ws.dev.java.net/>
- [7] Apache Ant: <http://ant.apache.org/>
- [8] WS-ResourceProperties (WSRF-RP) specification: [http://docs.oasis-open.org/wsrp/wsrp-ws\\_resource\\_properties-1.2-spec-os.pdf](http://docs.oasis-open.org/wsrp/wsrp-ws_resource_properties-1.2-spec-os.pdf)
- [9] WS-ResourceLifetime (WSRF-RL) specification: [http://docs.oasis-open.org/wsrp/wsrp-ws\\_resource\\_lifetime-1.2-spec-os.pdf](http://docs.oasis-open.org/wsrp/wsrp-ws_resource_lifetime-1.2-spec-os.pdf)
- [10] Web Services Notification: [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsn](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn)
- [11] Web Services Addressing: <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>
- [12] Resource Namespace Service: [http://www.ggf.org/GGF17/materials/272/Resource\\_Namespace\\_Service\\_Refactored.pdf](http://www.ggf.org/GGF17/materials/272/Resource_Namespace_Service_Refactored.pdf)
- [13] Security Assertion Markup Language: [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security)
- [14] XACML: [http://www.oasis-open.org/committees/download.php/2713/Brief\\_Introduction\\_to\\_XACML.html](http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html)
- [15] Usage Record - Format Recommendation: <http://www.ogf.org/documents/GFD.98.pdf>

- [16] SOAP Message Transmission Optimization Mechanism:  
<http://www.w3.org/TR/soap12-mtom/>
- [17] Resource Description Framework (RDF):  
<http://www.w3.org/RDF/>
- [18] RDF Schema (RDFS): <http://www.w3.org/TR/rdf-schema/>
- [19] OWL Web Ontology Language: <http://www.w3.org/TR/owl-features/>

## 1 WP 4: Summary: Project Month 6

The objective of this Work Package 4 is to develop and enhance the core service infrastructure that will form the foundations of the DMI platform, which will be deployed to this infrastructure and delivered to the consortium and beyond by WP3. The architecture effort of WP2 determines the components to be developed and included as part of the DMI platform and the modelling and language research undertaken in WP1 will form the basis of the DMI model for the platform. Where requirements are identified for enhancing the capabilities of the infrastructure for wider applicability across a number of tools from WP5 and services from this work package, respectively, the infrastructure will be enhanced accordingly.

The core service infrastructure is known as USMT (Unified Systems Management Technologies). USMT will provide a set of common service management capabilities that are required for monitoring and management of services hosted by the DMI platform. USMT, provided by Fujitsu Labs of Europe at the start of the project (as V0 in task WP4.1), will be enhanced and expanded as required by the work of the project to support the needs of Data Mining and Integration.

Data Mining and Integration processes are implemented as a set of individual processing steps using Grid services, which are combined to form the end to end process. These processes can be complex workflows involving several sub-processes such as data selection, data filtering, data transformation, data integration, data modelling (applying a data mining algorithm), and post-processing the mining results (e.g. visualisation). The Data Mining and Integration processes are often large in terms of the number of sub-processes in the workflow; in terms of the total execution time of the process; and in terms of the data volume involved.

In this work package, USMT will be modified to facilitate the management and monitoring of complex DMI oriented workflows, by integrating a number of technologies.

Current and future activities include the OGSA-DAI platform, an enhanced monitoring framework for DMI oriented processes, the WEEP enactment engine, the OGSA-DQP platform, a semantic registry and semantic provisioning services for providing advanced service and resource publishing and discovery mechanisms.

USMT will give rise to a layered Data Mining and Integration environment that will allow DMI services to make use of encapsulated DMI service with USMT as the foundation layer of each.

In months 1 to 6 of ADMIRE WP4's main focus has been to deploy USMT to the project and to port OGSA-DAI to USMT. Two versions of USMT have been distributed to the project, enabling the project to gain experience of USMT from an early stage and for the USMT developers to gain feedback for future enhancements. The port of OGSA-DAI to USMT is proceeding well. The architectures of the two parts have been found to match well. However, there have been some problems with adjusting to the JAX-WS technology that underlies USMT, which has slowed progress.

### 1.1 Products delivered to other work packages

FLE released two versions of USMT to the ADMIRE project, primarily to enable UEDIN to integrate OGSA-DAI with USMT:

- USMT for Admire V0 was released to ADMIRE on 9 April 2008.
- USMT for Admire V0.5 was released to ADMIRE on 25 June 2008.

These releases were the work of WP4.1.

## 2 Progress against Planned Activities

### 2.1 Activity: OGSA-DAI Integration (WP 4.2, 4.3)

#### 2.1.1 Planned activity

The OGSA-DAI [4] platform technology provides extensive data access, integration and management functions for a large variety of database operations and data handling scenarios. An objective of this work package is to help deploy the OGSA-DAI platform to the USMT, leveraging all of the generic capabilities offered by the USMT, in particular those to monitor workflows and report faults, and extending the infrastructure with any data oriented management functionalities. The University of Edinburgh will work with FLE in this activity.

#### 2.1.2 Actual activity

##### Architectural Match

At the start of the ADMIRE project both USMT and OGSA-DAI were mature software products with a history of independent development and so an initial concern was that their assumptions about the creation and control of resources were compatible or could be made to be compatible. OGSA-DAI presentation layers for several other grid platforms already exist, including those provided by the Globus Toolkit, by Unicore [5] and by gLite [5]. OGSA-DAI has a very clean architectural (and implementation) separation between its core functionality and presentation layers and so it was expected that the implementation of a presentation layer for USMT would not require any changes to the OGSA-DAI core. This has indeed proved to be the case, which has been reassuring for the OGSA-DAI development team. The USMT presentation layer for OGSA-DAI follows similar lines to the Globus Toolkit presentation layer. Where the USMT functionality differed significantly from the Globus Toolkit functionality, USMT was altered to provide hooks similar to those provided by the Globus Toolkit. While not strictly required, these hooks simplified the task for the OGSA-DAI developers.

Architecturally, the challenge of designing a USMT presentation layer for OGSA-DAI is that USMT has been designed to be in control of the creation of services, resources and resource properties, whereas the model used by OGSA-DAI is that these are created inside the OGSA-DAI core. The OGSA-DAI presentation layer (which is the interface to USMT) is simply told that they have been created. This was not the model initially in the mind of the USMT developers so that has required some changes to the USMT API. We are currently investigating what the implications of maintaining the approach of the OGSA-DAI core being in control are and whether this will limit us in the future of the ADMIRE project - as it may restrict how much of the USMT resource management functionality can be used.

##### OGSA-DAI Changes

A new USMT presentation layer for OGSA-DAI was being developed. OGSA-DAI provides six services:

- Data Request Execution Service (DRES)
- Data Resource Information Service (DRIS)
- Data Sink Service (DSiS)
- Data Source Service (DSoS)
- Session Management Service (SMS)
- Request Management Service (RMS).

On the server side, the DRES, DRIS, SMS and RMS are almost complete. The DSiS and DSoS require more work to be done on them than the others but are close to completion. OGSA-DAI also provides a Java client toolkit to ease the development of clients that make use of OGSA-DAI services. This is

currently being adapted to the USMT environment but a significant amount of work remains to be done on the client.

The developers of USMT envisaged that USMT would create all of the resources that it was involved in managing. However, OGSA-DAI creates (and destroys) resources that USMT would like to know about, so a mechanism must be put in place to allow OGSA-DAI to notify USMT of such events and to communicate the properties of these resources. A partial implementation of this (registering newly created resources but not deregistering them) has been developed with relatively little effort.

From an implementation point of view many of the issues that arose related to the JAX-WS technology [6] that underpins USMT. These issues are probably due to the developer of the OGSA-DAI USMT presentation layer being unfamiliar with JAX-WS and also because the JAX-WS documentation not being particularly good. It should be noted that the developer was familiar with other web service technologies more frequently used by the grid community such as Axis. It is advised that the USMT documentation should not assume users are experts in JAX-WS but instead should be written at a level that will allow users with a general understanding of web services to successfully develop USMT services.

The specific issues that slowed the development task include:

- To determine which instance of a stateful web service should service a request JAX-WS (and hence USMT) uses EPR reference parameters, in particular the XML element “jax-ws:objectId” carrying a UUID. Hence a client must use WS-Addressing EPRs when communicating with a service hosted by USMT. This was a novel feature as far as the developer of the OGSA-DAI USMT presentation layer was concerned and took some time to understand.
- WSDL to Java development is harder than it should be because of the lack of support in WSDL for inheritance.
- JAX-WS does not yet support WS-Addressing 1.0 Metadata which is unfortunate. It is hoped that when it is supported it will be possible to generate javax.xml.ws.Action annotations on Java code from WSDL.
- USMT is built on top of JAX-WS. This means that the presentation layer has to deal with a SOAP engine and a Java to XML binding (JAXB) that have not been used before in OGSA-DAI presentation layers

With a fully working implementation, running at least independently deployed sites, these problems have been overcome and should have no further impact.

In addition the development task highlighted the need for a more comprehensive developers' guide. Examples of information that should be in the guide are:

- Details on the annotations that USMT requires to be present.
- A description of the Toolkit class and how it may be used to find nuclei and hence the EPRs of services deployed in the nuclei.
- The contents of the build.xml file of ADMIRE-USMT.
- How USMT treats the different kinds of services :
  - plain JAX-WS based services
  - stateful USMT services
  - singleton (yet stateful) USMT services
  - USMT core services.

## Tooling

The development work starts with composing WSDLs for each of the OGSA-DAI services. Derived from an existing version of OgsadaiService.wsdl developed in the OMII-UK project, eight standalone WSDL files were composed, which include the six OGSA-DAI services, and two extra BasicService related WSDL files. Because DRIS, SMS, and RMS share same operations, and it is only the creation of each service instances that are different, by inventing a BasicService, the implementation of these

services can be made easier by employing inheritance approach later on in Java space. To guarantee the interoperability between USMT hosted presentation layer and OGSA-DAI backend, all the data types are generated from the original OGSA-DAI schemas with the current version 3.0.

An ant [7] build.xml file was developed to accomplish the Web services tooling process. The main task is the “wsimport” command from the JAXWS platform. Note that an “xjc” task is also provided in the build file, which mainly is used for JAXB schema compile. However, in this development strategy, the wsimport task is adopted since it not only generates the datatypes from schemas, but also generates the service interfaces and the service classes that create the services, therefore ease off the burden and avoid unnecessary mistakes when creating them manually. The only downside is that the faults are generated as well in the same packages, and in most cases, they are the same as they all derived from the same schemas. A simple and nice solution to this issue is to execute another task, which copies over all the faults to a centralized fault package, and removes the tooled ones, after the wsimport tooling process. This automation aid proved to be effective for its transparency to the developers under the circumstance that the JAXWS platform is inflexible and imperfect in this aspect.

There are two ways to match the schema/WSDL namespace to a package name in JAXWS and JAXB, despite the namespace is the package name by default: embedded binding declarations, and external binding declaration. As names suggest, embedded binding allows binding information appears inside a schema/WSDL file, as long as the element is declared in the JAXWS namespace, whereas external bindings read the information from external XML files. Both approaches have disadvantages: they either need to modify the schema/WSDL files themselves, which make maintenances difficult, or the tooling structural ends up with lots of external binding files. Nevertheless, the external binding method is chosen for the purpose of keeping the integrations of the schema/WSDL files. Whether or not the external binding files can be kept in one giant file remains a question, this could be further investigated later on in the project as manpower and effort allow.

The initial implementation started from code developed in the OMII-EU project to port OGSA-DAI to UNICORE. However, there are significant differences between these two Web services platforms and the implementations need to be carefully considered. In spite of this, the development is making good progress.

### **2.1.3 Deviations from plan**

This activity was slightly delayed from the plan and although WP4.2 and WP4.3 were not complete by the end of project month 6, they will be shortly afterwards. The reasons for this delay have been discussed above and rise essentially from the unfamiliarity of the project partners with the technology underlying USMT (JAX-WS). At the time of finalizing this deliverable, these tasks have been completed.

## **2.2 Activity: General Enhancement to USMT (WP 4.4, 4.5)**

### **2.2.1 Planned activity**

As a service-consolidating platform, USMT will provide (at the start of the project) a set of core, common management capabilities, including:

- Service monitoring, including monitoring of both generic and application-specific properties;
- Lifetime management, including creation and destruction of services;
- Lifecycle management, including provisioning, configuration, and activation of resources;
- Subscriptions, requests and notification, including lifecycle and property change events;
- Service naming and addressing;
- Basic registry provision;
- Security; and
- Fault propagation and handling.

Over the course of the project, these functions may need further development and enhancements as dictated by the requirements of the DMI platform. Any such work will be the responsibility of this work package and will be carried out by FLE.

To this end, over the lifetime of the project, as new tools and services are deployed to USMT, infrastructure requirements and enhanced capabilities will be identified. Where a capability is identified as a new infrastructure requirement, and can be generalized for provision to a number of tools and services, this capability will be integrated into USMT for wider applicability.

On the other hand, if an infrastructure functionality is identified that is specific to a given tool or service, that functionality will be provided specifically to support the tool or service for which it was identified. The overall design goal is to create as much reuse of functionality as possible, where that functionality is identified as being of a general and broadly applicable nature, while also supporting the specific functional requirements of every tool and service deployed to USMT.

### 2.2.2 Actual activity

#### Property access, support and management

USMT provides support for WS-ResourceProperties [8] allowing for uniform access patterns to properties regardless of type. USMT provides for read access to single properties, or to a set of properties. Clients to a USMT service may also update existing properties with new values if allowed: WS-ResourceProperties may be programmatically configured to allow or disallow WS-based updates. USMT does not, however, support more complex WS-ResourceProperty design patterns such as the creation and deletion of WS-ResourceProperties during the lifetime of the WS-Resource. Although the corresponding standards specification (WS-ResourceProperties) describes such design patterns, USMT does not support them for system design purposes.

Activities during this period in this component of USMT mostly comprised of better stability in multi-threaded access situations. Also, the interactions with other components of the USMT framework have been enhanced, such as extended functionality in the property value update functionality. The biggest improvement in this component was a result of the OGSAS-DAI porting efforts.

Previously, services on top of USMT had to programmatically declare WS-ResourceProperties in the WS connector layer. This effectively led to service implementations that concentrated the WS-ResourceProperty business logic in the WS connector, which in turn led to more monolithic Web Service implementations. The OGSA-DAI porting effort expanded the use cases on WS-ResourceProperty handling and management in that services, particularly pre-existing services, already implement the property business logic elsewhere, and use the USMT service implementation merely as a transport connector to that business logic. Hence USMT now supports service implementations *registering* the existence of WS-ResourceProperties with the framework at runtime. Two alternatives are provided: A service may register an implementation of the published interface for declarative WS-ResourceProperties for use cases where external (e.g. WS client) manipulation of property values is necessary or required; alternatively the business logic is interested in only providing read access for a WS-ResourceProperty. In this case the WS-ResourceProperty merely consists of a QName (uniquely identifying the property) and a value provider – the WS-ResourceProperty is *virtual* and exists only conceptually.

#### Lifetime management

USMT provides full support for service lifetime management [9]. Each USMT Nucleus (a group of services deployed together, including the Nucleus itself) provides for generic factory functionality. Deployed services, if meeting certain criteria, may be instantiated into individual WS-Resources as per WSRF specification, by using this generic factory. Creating a WS-Resource constitutes the start of its lifetime. Potentially, a WS-Resource may exist for an indefinite amount of time (future versions of USMT may allow for system policies to restrict this). However, the lifetime of a WS-Resource may be restricted to a certain point of time in the future by giving a new termination time or a duration for how much time the WS-Resource is allowed to exist. Alternatively a WS-Resource may be destroyed immediately regardless of its currently configured termination time.

Lifetime information, and current server time information (to calculate time skews between client and server) are provided as WS-ResourceProperties allowing clients to access this type of information using a generic methodology.

Activities in this component comprised of fixing several bugs in the handling of the remaining lifetime and provisioning of that information in certain corner cases. Further integration with other components of USMT has been improved, for example the interaction with Lifecycle management.

### **Lifecycle management**

Each service in USMT has an individual life cycle within its lifetime. USMT does not restrict services to a particular lifecycle model (or state model). Instead, USMT allows services to super-impose their state model with a “management state model”. That specific state model provides a unified view on the service from a management perspective: To a manager it is important to know whether (and where) a service exists, whether or not it is commissioned or whether it is currently active (i.e. providing the function for which it was previously commissioned). Managers may be humans, but also more complex services that outsource or reuse distributed functionality. Mostly, however, a service specific state model folds completely into the “Active” management state.

Support as been improved for services that fold their state model into the “Active” management state, as well as for services that do deeply integrate the management state into their specific state model. Also, services are now allowed to directly manipulate the management state value without having to execute the WS-related code for generic lifecycle management.

### **Subscriptions and event notifications**

USMT provides support for event notifications [10]. Clients may subscribe to changes in WS-ResourceProperty value changes as well as any generic notification topic provided by the targeted service. The subscription model is basic; one specifies the QName of the topic to subscribe to. If the topic QName happens to be a WS-ResourceProperty QName then the client registered to WS-ResourceProperty value change notifications for the WS-ResourceProperty of the given QName. Generic topics are handled similarly. “Posting” a notification to a topic (so that subscribers are notified of the posting, or message) happens by modifying the contents of the declared topic.

Currently, clients cannot specify patterns of notification frequency so whenever a certain notifiable event happens notification messages are sent out to every subscriber.

Activities in this component mainly focused on bug fixing, mainly due to the complex nature of this component. USMT also now provides notification support for several generic topics:

- Lifetime ResourceProperty: Subscribers are notified of the new lifetime whenever it changed
- Destruction confirmation topic: This generic topic notifies the subscribers (including the USMT basic service registry) of the destruction of this WS-Resource, whether explicitly destroyed, or through lifetime expiration.
- Lifecycle state ResourceProperty: Subscribers are notified of changes in this management state.
- Any service specific Resource Property: If configured appropriately, any service specific ResourceProperty may also serve as a notification topic for its value changes.

### **Service naming and addressing**

A core concept in USMT identifies a service's type by the QName of the WSDL portType it implements. Service instances (i.e. WS-Resources) are primarily identified by WS-Addressing EndpointReferences [11] that must be used when communicating with the respective WS-Resource. The USMT framework does not provide for user-friendly names for services. Future versions of USMT will provide for unique names for WS-Resources. Those names, however are mainly used for programmatic service identification beyond WS-Addressing EndpointReferences, and are not necessarily suitable for use as human readable names. Clients, or humans, may use a RNS (Resource Namespace Service) [12] implementation to map user-friendly names to EndpointReferences.

ADMIRE has not yet required any changes in this function, however due to the importance of naming in data management, significant work in this area is expected as the project progresses.

### **Basic registry provision**

USMT provides a basic registry, which collects the EPRs to each WS-Resource instantiated on the associated USMT Nucleus. This Registry uses the provided USMT management infrastructure (including lifetime, lifecycle and notification management) to track the state and existence of the WS-Resource identified by an EPR held in its store.

Clients may access this list of EPRs via the WS-ResourceProperty access pattern.

ADMIRE has not required extensions to this simple registry concept, nor are such changes likely. ADMIRE is more likely to develop a complex registry service for data object instances on top of USMT as a specific service.

### **Fault handling and propagation**

USMT models WS faults as Java exceptions. By implementing a custom Java exception class, declaring it as a WS fault using JAX-WS annotations and providing the fault content structure as a JAXB-annotated Java Bean a service developer easily plugs into the fault structure of USMT. Hence a service developer merely needs to throw a Java exception to cause the appropriate WS fault being communicated back to the client.

USMT does not provide for complex fault hierarchies and structures, mostly due to the limited support from the underlying JAX-WS infrastructure. Currently this is not seen as a problem for ADMIRE as so far ADMIRE services require only a very simple fault architecture.

### **Security**

USMT uses HTTPS as authenticated and secured communication channel for all WS message exchanges. System administrators can, by configuring USMT's trust store, control who may access WS-Resources on a USMT Nucleus.

#### **2.2.3 Deviations from plan**

The initial release of USMT V0 was available well ahead of schedule. This allowed other members of the project to provide feedback and bug reports sooner than expected and enabled FLE to release an updated version USMT V0.5.

The level of security support is not as advanced as was envisioned for this period. This was mainly due to an extended requirements gathering process and clarification process conducted by the project. The planned solution using SAML and XACML in conjunction ETD will provide a much stronger and flexible security framework than what had been originally envisioned.

### **2.3 Other Activities**

No other activities were active this period. For reference these are:

- Activity: Enhanced Monitoring
- Activity: Semantic Registry Integration
- Activity: Semantic Provisioning Services Integration
- Activity: WEEP Integration
- Activity: Standards Alignment

## **3 Risks and Issues**

### **3.1 Project issues**

No issues recorded against WP4 in this stage.

### **3.2 Significant problem reports**

No significant problem reports recorded against WP4 in this stage.

### 3.3 New risks

#### TR14 USMT reliance on IP multicast may hinder wider deployment and ADMIRE takeup

USMT V0 and V1 currently make use of IP multicast to perform parts of its discovery process. Many (possibly most) major router vendors' equipment has multicast routing and filtering *disabled* by default. This is the case with the Cisco routers at the core of Edinburgh's (and EPCC's) network, as well as with equipment from various other major vendors Edinburgh have used. The routers can, of course, have this functionality enabled (and, of course, most vendors support it to different degrees and with various levels of standards-compliant interoperability). However, over reliance on multicast may hinder wider deployment of USMT-based tools if typical network configurations have it turned off by default.

**Date:** 15/07/2008

**Owner:** WP 4 Leader

**Likelihood:** Medium

**Impact:** Severe – the impact is on usability of project products rather than an impact to the project itself. However, the impact is severe.

**Exposure:** High

**Control:** Provide ADMIRE partners with clear and simple requirements and configuration guidance as to how their networks should be configured.

**Contingency:** Should this become a major issue, discovery can be provided through static configuration mechanisms and made available as an option in USMT. This is not planned at this time.

## 4 Plans for next Period

### 4.1 Activity: OGSA-DAI Integration (WP 4.4)

See Section 2.1.1 for the description of this activity.

The next period's activity will be a continuation of the first period activity. The primary focus will be to complete the OGSA-DAI port to USMT so that all the existing OGSA-DAI system tests run on the USMT platform. To achieve this result requires:

- Completion of the USMT server-side presentation layer for all six OGSA-DAI services.
- Completion of a USMT client-toolkit presentation layer for OGSA-DAI. This is required because the system tests are implemented using the client toolkit.
- Implementation of a JNDI-like configuration method for OGSA-DAI USMT.

After these tasks have been completed we will integrate OGSA-DAI with the USMT security infrastructure. It is anticipated that OGSA-DAI USMT presentation layers will have to do very little in order to use the security infrastructure provided by USMT. However, some OGSA-DAI specific security functionality is anticipated. This will include:

- Integration with the access control security infrastructure. Develop a plug-in to the USMT security infrastructure that will extract all the resources referred to in an OGSA-DAI workflow and contact the access control infrastructure to ensure the user is able to access all these resources.
- Adapting the OGSA-DAI activities that communicate with other OGSA-DAI servers to use explicit trust delegation.
- Ensure that an appropriate level of access control is given to dynamically created resources. Initially this will probably grant access to the creator only.

If time allows, the following areas will also be investigated:

- Integration of OGSA-DAI with the advanced monitoring functionality of USMT. This task should be driven by scenarios of the ADMIRE architecture.
- Integration with an efficient SOAP-based data transfer mechanism such as MTOM.
- The ability for OGSA-DAI to deliver OGSA-DAI data to a file than can be accessed directly as an HTTP or HTTPS URL. Here USMT will need to allow the creation of a file in a HTTP(S) server and ideally specify some access control to that file.
- Investigation of the advantages of making OGSA-DAI workflows and activity instances explicit resources. This task should be driven by goals of the ADMIRE architecture.
- Enable OGSA-DAI services with lifetime and lifecycle management.

## 4.2 Activity: Enhanced Monitoring (WP 4.5, 4.6)

From the Description of Work: *USMT provides an extensive, standards based framework for monitoring a variety of services. However, there are a number of properties specific to the monitoring of data integration processes that will need to be integrated into the infrastructure. The University of Vienna has done extensive work in this area and will work with FLE to integrate extended monitoring functionality into the infrastructure.*

Work will start on this activity during this period.

USMT provides an infrastructure for passing information about changes in the properties of service instances. The integration of property change notification and resource properties makes it an easy task for service developers wishing to expose changes in any service level managed property. The limitations include not having a mechanism to limit the frequency of notifications for rapidly changing properties nor an ability to periodically notify recipients of the state or a property even when it has not changed. These extensions will be considered in light of ADMIRE requirements.

ADMIRE has also identified two other areas where extensions to notification support may be required, namely event aggregation and resource usage monitoring.

**Event Aggregation:** This amounts to creating a configurable implementation of the WS-Notification specification for a NotificationBroker [10]. The configuration would allow implementers to define which notification types to aggregate and how to perform the aggregation, e.g. batching, averaging, max/min collection, scatter/gather, or simply recording event for audit purposes. The baseline functionality and the balance between configuration and development through extension will be based on requirements from ADMIRE.

**Resource Usage:** This monitoring requirement amounts to gathering information about the service execution environment, i.e. CPU usage and load, disk and memory occupancy, network bandwidth available, etc. This will represent several challenges. In modern service environments like USMT, the service itself is completely isolated from this information. USMT, while having access to this information for the collection of all services running under its control, has no mechanism available to isolate the load/usage made by a specific service instance. Furthermore, with the increase in popularity of system virtualization in data centre deployments, this information is not really available to the hosting operating system let alone the USMT platform. Solutions to these monitoring requirements will require careful balancing of ADMIRE requirements against the level of flexibility expected in deployments. For example, all this information can probably be made available if there is only one USMT instance on each (non-virtual) server (and nothing else on that server) and only one service hosted by each USMT instance. This however, is probably too restrictive with respect to deployment. Compromises will probably be required.

## 4.3 Activity: Semantic Registry Integration

From the Description of Work: *The initial version of the USMT includes a basic registry that supports service discovery and management of property-based information about the target services. Enhanced support is also available to use this registry mechanism to provide collective management of collections of services, e.g. a workflow of processes or a network of clusters. Within this activity we will assess the ability to either extend this functionality to provide semantic content, i.e. by providing a*

---

*SPARQL based query capability, or alternatively by integrating a third part semantic registry, e.g. Grimoires, the semantic registries of the NeOn Toolkit or the Metadata Registry of the Semantic Web Framework.*

No work planned for this period.

#### **4.4 Activity: Semantic Provisioning Services Integration**

From the Description of Work: *In order to exploit the ontologies developed in WPI and the descriptions stored in the Semantic Registry integrated in the previous activity, it will be necessary to integrate a set of semantic provisioning services in the USMT infrastructure. These services are those identified in the S-OGSA architecture which have been developed in the OntoGrid project: the Ontology Access Service (formerly known as RGAB or WS-DAIOnt-RDF(S)), the Semantic Binding Service, and the Semantic Housekeeping Service. Within this activity ADMIRE – Framework 7 ICT 215024 these services will be ported to the USMT infrastructure and integrated with the Semantic Registry.*

No work planned for this period.

#### **4.5 Activity: OGSA-DQP Integration**

From the Description of Work: *The OGSA-DQP software augments OGSA-DAI with query planning and distributed query capabilities, which allow large queries to be expressed and split across federated data resources. An objective of this work package is to deploy the OGSA-DQP extensions to the USMT, ensuring that the query planner benefits from the increased information provided about resources to make planning decisions. The University of Edinburgh will work with FLE in this activity.*

No work planned for this period.

#### **4.6 Activity: General Enhancement to USMT (WP 4.6)**

See Section 2.2.1 for the description of this activity.

The enhancements to the USMT Security component will be completed to make use of SAML (Security Assertion Markup Language) [13] based explicit trust delegation for authenticating a client, and XACML [14] based access control for authorizing individuals for WS-Resource access. By authoring the XACML policies users, system administrators and service developers will be in full control of the granularity of the access control.

Otherwise, and beyond the specifics noted elsewhere in this section and general bug fixes and performance improvements, there are no enhancements planned at this stage. However, as the development period progresses and ADMIRE requirements become clear, enhancements may be added as needed.

#### **4.7 Activity: Standards Alignment**

From the Description of Work: *The development of USMT has been based on Web services standards. More specifically, USMT is largely based on the Open Grid Services Architecture (OGSA) of the Open Grid Forum (OGF) and the WS-RF specifications developed in OASIS. Future USMT developments, and any modifications made as required by the DMI platform, will continue to be based on existing and developing standards.*

During the first period ADMIRE has identified an initial set of standards which will be tracked to ensure that ADMIRE remains aligned with them.

In the development of monitoring functions it may be necessary to pass information around related to resource usage. If this is the case, the OGF specification for Usage Record [15] will be used as the

content for messages relating to usage. As this specification is extensible, ADMIRE will be able to test additional types of usage related data, e.g. database transaction load. If ADMIRE sees a potential for this type of usage information in a wider context than this project, these extensions can be submitted to the Usage Record working group in the OGF.

In support of data transfer capabilities, MTOM [16] or some other standards based attachment technology will be needed. In particular, USMT will need to implement a deliverToMTOM activity and also a readFromMTOM activity in the context of OGSA-DAI.

Likewise the implementation of security will be based on SAML and XACML standards. It is possible, but unlikely, that ADMIRE will be able to provide feedback to these groups, because they are already quite mature specifications.

The DMI platform will support the management of metadata and ontology languages proposed by the World Wide Web Consortium (W3C), such as Resource Description Framework (RDF) [17], RDF Schema (RDFS) [18] and possibly OWL (Web Ontology Language) [19]. These languages will be used as they are proposed in their respective specifications, and no modifications will be done to them, although ADMIRE may profile them to narrow the implementation scope and these profiles could be made available to interested standards development groups.

Besides, the management of these languages will be done following the ongoing OGF specification WS-DAI-RDF(S), where UPM is involved.

## Appendix A WP 4 overall timeline

WP	Time	Activity Description	Status
4.1	M1-2	Support deployment of existing USMT middleware on test bed – V0.	Done
4.2	M2-5	Support the porting of the OGSA-DAI middleware for deployment to the USMT.	Done
4.3	M4-6	Initial deployment of the OGSA-DAI middleware to the USMT – V1.	Done
4.4	M5-7	Start integrating available DMI platform services into the USMT.	Ongoing
4.5	M5-7	Ensure that the DMI basic required management capabilities (i.e. data service monitoring, lifecycle management and process enactment) are adequately supported by the USMT.	Ongoing
4.6	M6-11	Extend USMT capabilities specifically for DMI requirements as dictated by the DMI platform services.	
4.7	M10-12	Second round of deploying DMI platform services to the USMT – V2.	
4.8	M13-18	Improve performance of data oriented processes as supported by the USMT's capabilities, including advanced dataflow management.	
4.9	M17-19	Include the Enactment Engine platform and capabilities.	
4.10	M19-22	Enhance related lifecycle management capabilities supporting enactment engine functions.	
4.11	M22-24	Third round of deploying DMI platform services to the USMT – V3.	
4.12	M24-27	Improve USMT performance.	
4.13	M27-30	Add automated self-management capabilities for reliability, resilience and scalability. Add OGSA-DQP services to the USMT.	
4.14	M30-32	Fourth round of deploying DMI platform services to the USMT – V4.	
4.15	M32-36	Add plug-in capability to the USMT, to support component and DMI platform extensibility.	

## Appendix B WP 4 deliverable schedule

Deliv.	Time	Description	Status
D4.1	M6	<p>Development and Deployment Report for USMT V0 and V1: capabilities of USMT V1</p> <p>This report will describe the experience of deploying the USMT to the resources of the ADMIRE test platform. No major issues are expected in this task. The process of deploying the USMT will be documented, if different from that which is already available in the standard USMT release documentation.</p> <p>In addition, the report will describe what has been achieved in this period in porting the data access, integration and mining technologies (e.g. OGSA-DAI) from the ADMIRE partners to the USMT technology. It is expected that early stage problems will be identified in this process. These will be noted as a set of enhancement requirements for the USMT, to be addressed in the WP4 tasks in the upcoming periods.</p>	This document
D4.2	M12	<p>Development and Deployment Report for USMT V2: capabilities of USMT V2</p> <p>This report will describe the DMI services developed for the USMT in line with the requirements of Release 1 of the ADMIRE platform. The design and implementation of these services will be detailed and DMI developer documentation provided.</p> <p>It is expected that the design of the USMT may have to be evolved in order to gain performance and design benefits in line with the requirements of the ADMIRE platform. Therefore, the design of USMT extensions will be described in the context of the motivations for those extensions.</p>	
D4.3	M18	<p>Development Progress Report: requirements for USMT V3</p> <p>It is expected that the USMT deployment on the ADMIRE test platform will be fairly stable by the end of this period. The work in this period is expected to focus on any components and USMT improvements that increase performance of the ADMIRE platform. This report will describe the design of any such extensions to the USMT and the performance improvements gained as a result.</p> <p>In addition, the report will describe the work identified to integrate the Workflow Enactment Engine (WEEP) into the ADMIRE platform, by developing the necessary services to support the WEEP technology on the USMT. It is expected that a number of USMT service interfaces are required for the WEEP technology to be integrated with the ADMIRE platform based on the USMT.</p>	
D4.4	M24	<p>Development and Deployment Report for USMT V3: capabilities of USMT V3</p> <p>This report will be an update of deliverable D4.3 as it is expected that the work on the WEEP integration with the ADMIRE platform will continue to be the main focus of the USMT enhancements in this period.</p>	
D4.5	M30	<p>Development Progress Report: requirements for USMT V4</p> <p>This report will describe USMT performance enhancement work required following the integration of the WEEP technology into the ADMIRE platform and the evolution of the platform itself.</p> <p>As part of the USMT enhancement work required by ADMIRE, it is expected that a number of self-management services will be developed for the USMT, which will allow for the automated monitoring and management of services within an ADMIRE specific context. This work will be described in this report, including the designs of the ADMIRE specific self-management services.</p>	
D4.6	M36	<p>Development and Deployment Report for USMT V4 and Roadmap for Further Development.</p> <p>This report will describe the efforts to integrate the DQP technology into the ADMIRE platform based on the USMT. This work will be closely linked to the development of the final ADMIRE platform release, the details of which will be described in this report. A basic ADMIRE developer's guide will be produced in order to document technical details of the deployment and use of the ADMIRE platform.</p>	