Title:	*Provenance Implementation Design*
Workpackage:	*WP 9*
Author:	*John Ibbotson*
Contributors:	*Sheng Jiang*
Reviewers:	*All project partners*
Identifier:	*D9.3.3 including Security Implementation D4.3.1 and Scalability Implementation D5.3.1*
Type:	*Deliverable*
Version:	*1.0*
Date:	*30th November 2006*
Status:	*Public*

Summary

This document provides an implementation design for the Provenance deliverable D9.3.3 and includes descriptions of the security and scalability deliverables D4.3.1 and D5.3.1. It includes a description of the major interfaces, components and subcomponents together with a mapping to the architectural requirements it supports.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Members of the PROVENANCE consortium:

- IBM United Kingdom Limited United Kingdom
- University of Southampton United Kingdom
- University of Wales, Cardiff United Kingdom
- Deutsches Zentrum fur Luft- und Raumfahrt e.V. Germany
- Universitat Politecnica de Catalunya Spain
- Magyar Tudományos Akadémia Számítástechnikai és Automatizálási Kutató Intézet Hungary

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Table of Contents

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

1 Introduction

This document describes the design of the reference implementation of a Provenance Service as described in the project deliverable D3.1.1: An Architecture for Provenance Systems [1]. The architecture deliverable defines a data model and set of interfaces that allow Provenance aware applications to record documentation about processes and their execution and subsequently query the captured documentation. This reference implementation is a realization of the architecture using Grid standards and toolkit. This document lists the requirements relating to the Provenance Service and comments on how the implementation meets those requirements. It then provides details of the design together with the major components together with other third party software components required to deploy a Provenance Service.

In addition to the reference implementation of the Provenance Service, the project has developed a Client Side Library (CSL) for use by application developers. The CSL provides an interface to the Provenance Service functions whilst hiding the complexities of directly connecting to the store over a network. By separating the connection to the Provenance Service from a client application, users of the Provenance technology can take advantage of other implementations of the Provenance Service interfaces without altering their client code. The CSL has been tested with an alternative implementation of the Provenance Service interfaces; this implementation is the PreServ code developed as part of the PASOA project. The CSL design information is provided as a separate document D9.3.3a [2].

1.1 Security and Scalability Deliverables

The Provenance project has previously delivered two versions of a Security Specification ([3] and [4]) and a Scalability Specification ([5] and [6]). In the original project plan which forms part of the contract Technical Annex, these deliverables were to be followed by implementations of the specifications as deliverables D4.3.1 and D5.3.1. The second versions of the Security and Scalability specifications were produced at month 18 of the Provenance project and since then the implementation of the specifications have been included in the Workpackage 9 deliverable which is the subject of this document.

The initial project proposal was submitted in 2003 and included significant resources for the security and scalability implementation phase. In the intervening three years, there has been major progress in the development of toolkits and middleware for implementing and deploying Grid systems in a secure and scalable infrastructure. Notably this has resulted in the development of the Globus Toolkit which is currently at version 4. This toolkit has provided the underlying infrastructure for the reference implementation of the Provenance Architecture. As a result, the Provenance project has used the features of GT4 to provide the underlying services envisaged by workpackages 4 and 5. This has resulted in less resources than originally anticipated being needed to develop the security and scalability functions which have been included in the reference implementation of Workpackage 9. This also means that D4.3.1 and D5.3.1 do not exist as separate deliverables.

Descriptions of the security and scalability support are provided in this document.

1.2 Terminology

The following terms are used in this document:

AXIS The Apache SOAP engine provided as part of the GT4 release [8]
Collection A subdivision of an eXist XML database.
DOM The Document Object Model used for managing XML documents fragments [9]
eXist An open source implementation of an XML database [10]
GT4 The Globus toolkit version 4 [11]
OGSA-DAI Open Source implementation of the OGSA Data Access and Integration specification [12]
Pivot point The position in a Web Services interaction where AXIS handlers are invoked.
Tomcat The Apache Java based application server [13]
WS-RF The Web Services Resource Framework [7]
XMLDB A standardised interface to XML databases
XPath The XML Path language [14]
XQuery The XML Query language [15]
XUpdate An XML based language for updating XML documents and databases [16]

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

2 Requirements

The following implementation recommendations from the Provenance Architecture [1] are supported by the following design features.

ID	Description	Design Feature
IR-PS-4	Provenance stores for an application can be deployed in security domains to which all querying actors have credentials to access. Tools can provide single-sign on by storing credentials and using as appropriate for each provenance store.	This recommendation may be supported by a federated security infrastructure with actors having access to multiple security domains via a single sign-on infrastructure. GT4 supports several federated security models such as cross certification and the Community Authorization Service (CAS). Deliverable D4.2.1 identified CAS as being suitable for Provenance requirements. However, subsequent investigation revealed that GT4 does not support CAS-based authorization for web services (documented in the CAS administration guide). The Provenance project has therefore included information on cross certification in the release package for systems administrators that require this capability.
IR-PS-5	Process documentation cannot be removed from the data storage using the provenance store APIs; instead, it may be removed, following data retention policies, by making direct access to the storage layer, using curation methodology typical of long term storage.	The data storage system is a separate sub-component of the Provenance Service. The Provenance Service interfaces do not support the removal of process documentation, but this does not preclude its removal by direct access to the data storage by a user with an administration role.
IR-PS-6	The access control functionality of the provenance store may provide a way to model the expression of access control policies and rules so that they are functionally and / or semantically equivalent to the access control policies of other data stores.	For a Provenance Service to implement the same access control policies as other data stores, it must share the definition of those policies. The reference implementation uses an XML file that provides a user/role based access control feature for the Provenance Service. This allows access for users at the granularity of web services ports. This configuration file is used by a Provenance Service specific Policy Decision Point (PDP) which is compatible with the GT4 PDP architecture. If required, this feature may be configured by systems administrators. For the contents of the configuration file to be compatible with other data stores, there needs to be an administration process implemented to create the Provenance Service access control file from other system wide information. Since this process will be specific to different installations, a generic solution is not possible and therefore outside the scope of this project.
IR-PS-7	Scalability of query results can be implemented by caching of the results within the provenance store. A set of interface operations allows actors to iterate and retrieve the cached results in response to a query request.	The XPathFactoryPort port type implementation supports this requirement. The results of a query are cached in the Provenance Service and an iterator operation allows them to be retrieved by a querying client.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

IR-PS-8	Management of large number of p-assertions can be addressed by implementing the provenance store persistent store component using a proprietary or open source database management system.	The Provenance Service persistent store component is accessed via an OGSA-DAI data service. This allows alternative open source or proprietary database components to be used.
IR-PS-9	Implementations of the provenance recording, query and management interfaces should be clusterable.	The Provenance Service is identified by a single logical network endpoint address. This allows the implementation to be clustered by a set of application servers such as Apache Tomcat running instances of the Globus GT4 container. Clustering in this context means that copies of the GT4 container and its Web Services (such as the Provenance Service) can be run on multiple application servers to increase the capacity and performance of the service. The cluster of servers is identified by the single network endpoint address. A load balancing component captures any request messages to the cluster and passes the request to an instance of the application server according to the current loading status of the cluster thereby sustaining the performance of the cluster. Additional hardware and software components may be dynamically added and removed from the cluster to support its performance requirements without altering its logical interfaces to invoking applications.

The following implementation recommendations from the Provenance Architecture [1] are not supported by the reference implementation.

ID	Description	Design Feature
IR-PS-1	A provenance store implementation may use a metadata infrastructure such as RDF to annotate p-assertions with metadata information about time at which p-assertions are received through the recording interface or stored in persistent storage.	The P-Assertion schemas include elements of any type that are intended for application specific information and metadata. These elements may be used to annotate p-assertions with RDF metadata for application specific processing. The reference implementation is agnostic to application specific schemas therefore no specific support for RDF has been included
IR-PS-2	A provenance store implementation may document its interactions with asserters by storing p-assertions in some provenance store (itself or another one). This allows the provenance to document back ups, replication, time, etc.	A Provenance Service can itself be an actor that records p-assertions. Recording may be to itself or a separate Provenance Service by including the actor side libraries in an implementation of the Provenance Service. The Provenance Service implementation developed as part of the project does not include this feature.
IR-PS-3	A provenance store implementation may use a storage layer that offers replication of p-assertions in order to be fault-tolerant.	The Provenance Service implementation developed as part of this project separates the storage layer which is represented as a data service using OGSA-DAI. This separation allows a fault-tolerant database configuration to be used if required. So far, XML databases do not support

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

		fault tolerance to the same degree as relational database technologies so this feature is not supported by the project.
IR-PS-10	Implementations of provenance store should consider long term storage of p-assertions and the necessity to keep a long term copy of public keys (which may expire) to verify signed assertions.	Security lifecycle management considers the changes over an extended period of time of data security as threats to the data change. This is an ongoing research and development subject and out of scope for the Provenance project.

3 The Implementation Model

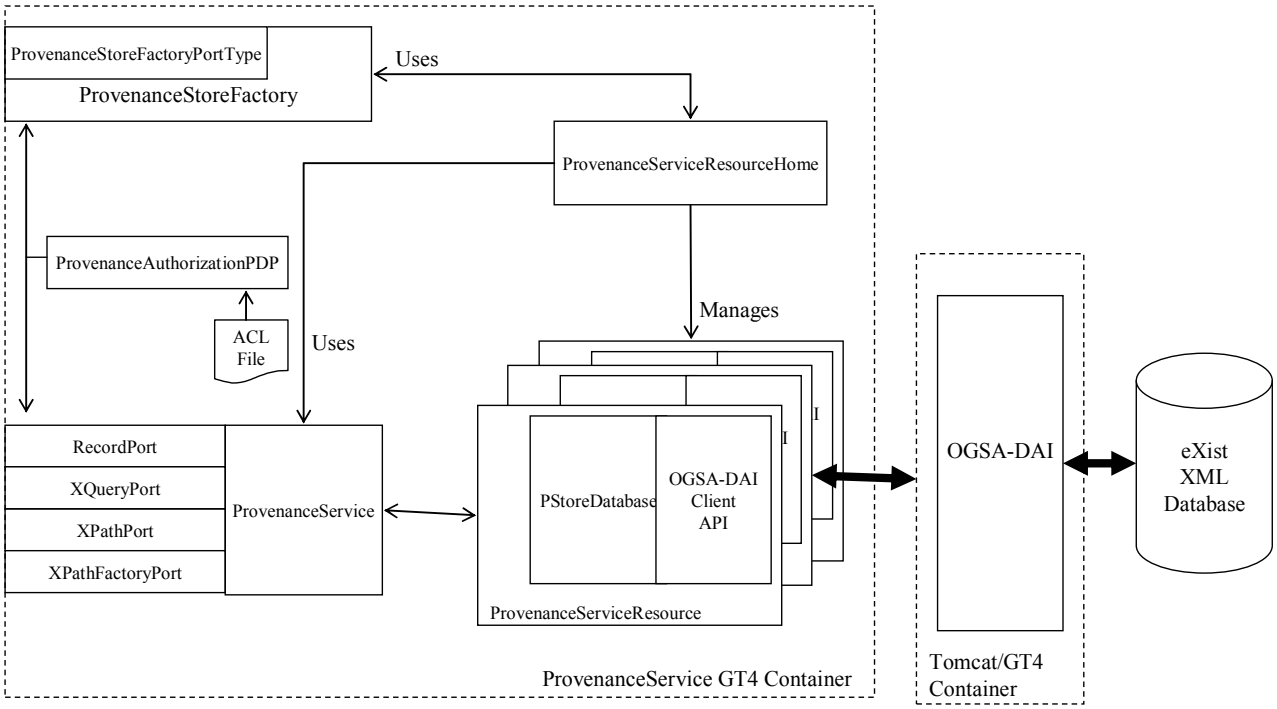

Figure 1: Provenance Service Components

The stateful Web Service pattern is described in a previous project deliverable [17]. We now identify the components of a Provenance Service that implements this pattern. These are shown in the Provenance Service Components figure.

1. The ProvenanceStoreFactory service creates an instance of a ProvenanceServiceResource. The service contains a single createResource operation that takes a CreateResourceRequest message as input and returns a createResourceResponse message. The ProvenanceStoreFactory service uses the ProvenanceServiceResourceHome to create a new instance of a ProvenanceServiceResourceHome whose WS-Addressing EndpointReference (EPR) is

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

returned to the requesting client in the createResourceResponse message. The client then uses this EPR to access the resource through the ProvenanceService interface.

2. The ProvenanceServiceResourceHome manages a set of ProvenanceServiceResource objects. It is used by the ProvenanceStoreFactory to create new instances of ProvenanceServiceResource objects and by the ProvenanceService find and access previously created instances.
3. The ProvenanceService is an implementation of the Provenance Service Web Service whose interfaces are defined by the WSDL in Appendix A2. The WSDL ports corresponding to the operations are:
    a. **RecordPort** – Record a set of p-assertions in the Provenance Service
    b. **XQueryPort** – Perform an XQuery search on the Provenance Service and return the results of the search
    c. **XPathPort** – Perform an XPath search on the Provenance Service and return the results of the search. This was provided for earlier releases of the eXist database which did not support XQuery.
    d. **XPathFactoryPort** – Perform an XPath search on the Provenance Service and return the number of entries in the result set. The GetItems operation then allows a client to iterate through the results
4. The ProvenanceServiceResource is an implementation of a WS-Resource as defined in [7]. An instance of a ProvenanceServiceResource is created at the request of the ProvenanceStoreFactory by the ProvenanceServiceResourceHome for use by an actor. The ProvenanceServiceResource contains an instance of a PStoreDatabase object which provides the internal interfaces to the persistent database through the OGSA-DAI client Application Programming Interface (API).
5. Persistent XML data storage is provided by an OGSA-DAI data service. This service may be hosted in the same container as the Provenance Service or alternatively on a separate, distributed container.
6. An eXist XML database provides the persistent database for the OGSA-DAI service.
7. The ProvenanceAuthorizationPDP provides access control to the Provenance Service through a customized Policy Decision Point (PDP). A PDP is an extensible point in the Globus GT4 architecture that allows implementations to add further security features. The Provenance Service can make use of an Access Control List (ACL) contained in an external file to grant access to the service at a port level.

Note that in the Provenance Architecture, a management port was identified but no functionality assigned to it. Therefore no implementation of such a port has been provided.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

3.1 Implementation Classes

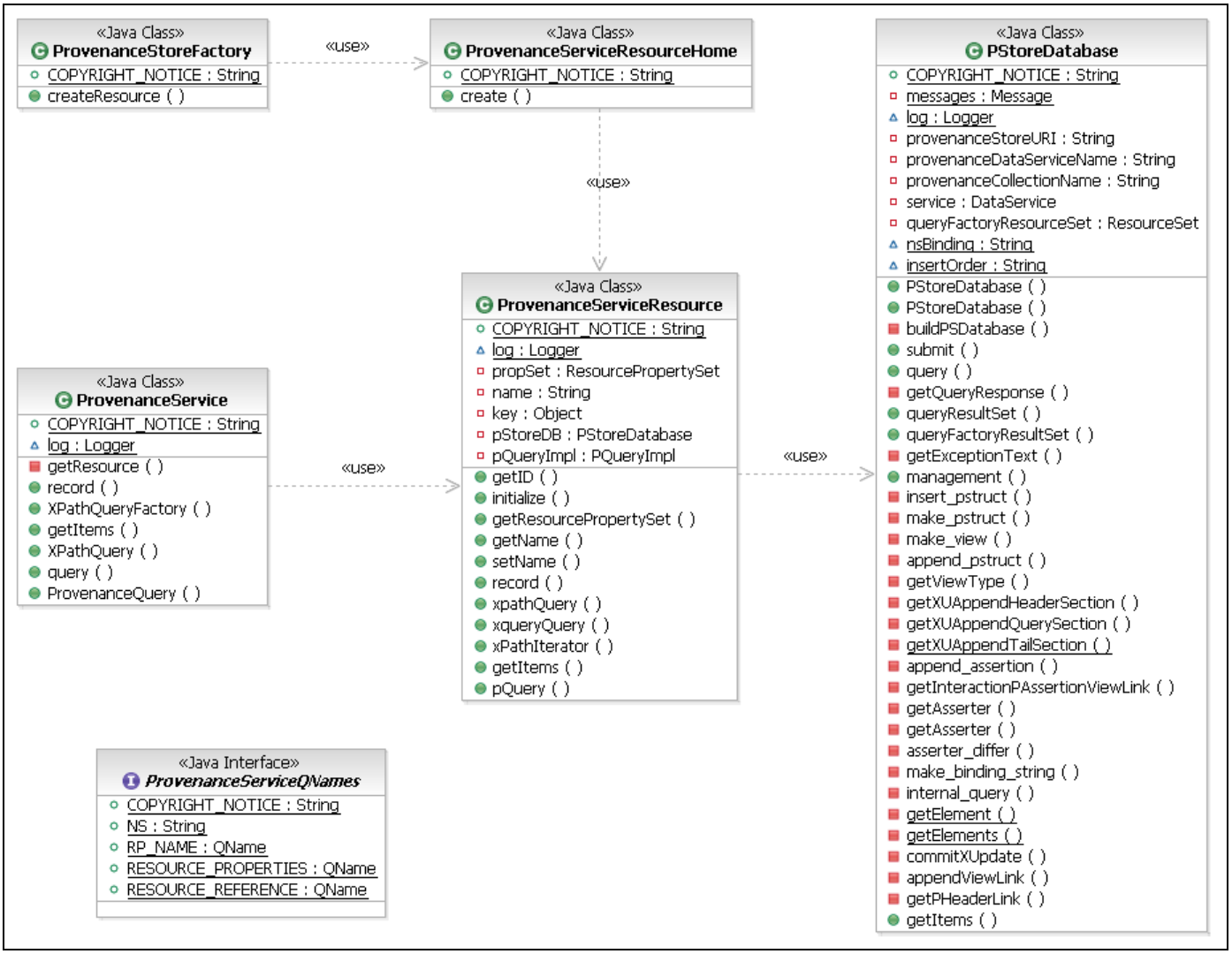

Figure 2: Provenance Service implementation classes

The significant Java classes that implement the Provenance Service are shown as a UML class diagram which represents the components shown in Figure 1. In this, the ProvenanceStoreFactory uses the ProvenanceServiceResourceHome class to create instances of a ProvenanceServiceResource. The ProvenanceService is an implementation of the Web Service defined in the Provenance Architecture that uses the WS-Addressing endpoint returned by the ProvenanceStoreFactory class to access a created ProvenanceServiceResource. The PStoreDatabase class acts as an internal interface to a created OGSA-DAI data service which interacts with the eXist XML database.

The implementation consists of the following Java packages:

org.gridprovenance.wsrf.server	Classes to implement the Provenance Service interfaces
org.gridprovenance.wsrf.security	Classes to implement the Provenance Service security support
org.gridprovenance.wsrf.client	A set of client examples used to test the Provenance Service interfaces without using the client side library

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

org.gridprovenance.commonservices A set of services used by other provenance Service implementation classes

4 Provenance Service Component Design

4.1 ProvenanceStoreFactory

The interface to the ProvenanceStoreFactory is shown in Appendix A.1. It consists of a single ProvenanceStoreFactoryPortType with a single createResource operation. The input message to the operation is of type CreateResourceRequest and is empty. The response is of type CreateResourceResponse. The response contains a WS-Addressing EndpointReference (EPR) to a newly created ProvenanceServiceResource. The EPR is used by subsequent invocations of the ProvenanceService Web service to correctly identify which ProvenanceServiceResource should be used.

An example EPR serialized as a string is shown:

```
<ns1:ProvenanceServiceResourceReference xsi:type="ns2:EndpointReferenceType"
    xmlns:ns1="http://www.gridprovenance.org/schemas/ProvenanceService"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/03/addressing">
    <ns2:Address xsi:type="ns2:AttributedURI">
        http://9.20.147.198:8080/wsrf/services/ProvenanceService
    </ns2:Address>
    <ns2:ReferenceProperties xsi:type="ns2:ReferencePropertiesType">
        <ns1:ProvenanceServiceResourceKey xmlns:ns1="http://org.gridprovenance.gt4">
            1832807742
        </ns1:ProvenanceServiceResourceKey>
    </ns2:ReferenceProperties>
    <ns2:ReferenceParameters xsi:type="ns2:ReferenceParametersType"/>
</ns1:ProvenanceServiceResourceReference>
```

The returned EPR contains the address of the Web Service to be used to access the ProvenanceServiceResource together with a key to identify the newly created resource. The GT4 client API allows a portType to be returned given an EPR as an input argument. The following code illustrates the 2 stage process of invoking the ProvenanceServiceFactory followed by the Provenance Service.

```
String factoryURI = "http://127.0.0.1:8080/wsrf/services/ProvenanceStoreFactory";
EndpointReferenceType factoryEPR, instanceEPR;
ProvenanceStoreFactoryPortType psFactory;
ProvenanceStorePortType pStore;

// Get the ProvenanceStoreFactory portType
factoryEPR = new EndpointReferenceType();
factoryEPR.setAddress(new Address(factoryURI));
psFactory = factoryLocator.getProvenanceStoreFactoryPortTypePort(factoryEPR);

// Create resource and get endpoint reference of WS-Resource.
// This resource is our Provenance Service "instance".
CreateResourceResponse createResponse = psFactory.createResource(new CreateResource());
instanceEPR = createResponse.getEndpointReference();

// Get instance PortType
pStore = instanceLocator.getProvenanceStorePortTypePort(instanceEPR);

// Create a Record structure .....
Record rec = new Record();

// ..... and invoke the Provenance Service Web service
RecordAck recAck = pStore.record(rec);
.....
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

4.2 ProvenanceServiceResourceHome

The ProvenanceServiceResourceHome class extends the GT4 ResourceHomeImpl class which implements the ResourceHome interface. The Provenance Service implementation has to provide a create() method which:

1. Creates an instance of a ProvenanceServiceResource
2. Initializes the resource
3. Gets a key for the resource and adds it to the set of current keys
4. Returns the key

4.3 ProvenanceServiceResource

The ProvenanceServiceResource class implements the GT4 Resource, ResourceIdentifier and ResourceProperties interfaces. Its purpose is to:

1. Encapsulate the interactions between the Provenance Service and the OGSA-DAI data service
2. Be a managed object that conforms to the lifecycle rules of WS-RF

The ProvenanceServiceResource.initialize() method is invoked by ProvenanceServiceResourceHome.create() when creating a new resource. This constructs a new instance of a PStoreDatabase that exists for the lifetime of the ProvenanceServiceResource instance.

The ProvenanceServiceResource also has methods for each of the three Provenance operations; record, query and manage. Each method receives as input argument the message sent by an actor to the Provenance Service. It then invokes the appropriate method on the PStoreDatabase for processing. The returned acknowledgements from the PStoreDatabase are then returned by the ProvenanceServiceResource object.

4.4 ProvenanceService

The ProvenanceService component implements an instance Web Service through which all interactions with the Provenance Service take place. Its interface is described by the WSDL in Appendix A.2. It consists of a set of port types that define operations on the Provenance Service. These are now described in more detail.

4.4.1 RecordPort

The record interface is defined in Appendix A.3 and its associated XML Schema file PRecord.xsd. The schema describes the record request and response messages defined in the Provenance Architecture document [1].

4.4.2 XQueryPort and XPathPort

Appendix A.4 shows the WSDL for the XQuery and XPath port types. These are supported by additional schema files XQuery.xsd and XPath.xsd. The two port types differ only in the structure of the query being submitted. The query request object contains two elements:

1. An XPath or XQuery expression
2. An optional namespace mapping element

The response object for the query operation is a QueryResponse object. This contains a single element of any type which contains the results of the query.

4.4.3 XPathFactoryPort

The XPathFactoryPort WSDL is shown in Appendix A.5 and implements a caching operation where the results of an Xpath query are stored by the Provenance Service rather than being returned as in the case of the XPathPort. The port defines two operations:

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

1. The XPathQueryFactory operation takes an XPath expression contained within a request message. It performs the query and returns the number of entries in the result set as a response message.
2. The second operation GetItems, allows a client application to retrieve subsets of the result set by providing an offset into the result set and number of items to retrieve. The response message returns the requested items.

4.5 OGSA-DAI Data Service

A design decision was made to adopt XML database technology as the best choice for storing Provenance documentation. The database implementation has been abstracted away from the core Provenance Service code by using OGSA-DAI to manage the connection to the database. This allows the Provenance code to concentrate on the key business functionality rather than database specific issues and provides the flexibility to deploy different database implementations supported by OGSA-DAI.

OGSA-DAI is a web service that is hosted from an application container such as Apache Tomcat or GT4. Just like the Provenance code, it uses the same client/server model with SOAP messages being sent between client applications and the OGSA-DAI service.

OGSA-DAI provides two ways of interacting with a database:

1. Perform document: This is an XML document that is constructed and passed to OGSA-DAI. It specifies the interaction to perform against the database, such as a query or an insert.
2. Client API: This is a Java API for interacting with OGSA-DAI. It does not require the programmer to develop perform documents and provides a neater, faster and more reliable way of using OGSA-DAI.

The Provenance database access code uses the OGSA-DAI client API to interact with the database. This client API is used by the PStoreDatabase class.

The fact that the database is abstracted away does not hide all database problems from us. OGSA-DAI makes no attempt to "normalize" database access, so we are still subject to the problems inherent with databases from multiple vendors, such as varying data languages and varying support at the JDBC/XMLDB driver level.

As the database access code is being written to access an XML database, we must use XPath, XQuery and XUpdate protocols to search and update the database. These protocols are exposed in the OGSA-DAI client API through the XPathQuery and XUpdate classes in the uk.org.ogsadai.client.toolkit.activity.xmldb package.

4.6 The eXist XML database

OGSA-DAI provides support for two XML databases; eXist and Xindice from the Apache consortium. Early in the project, we evaluated both databases for potential use. Xindice supported only Xpath for querying and had no published plans for adding XQuery support. The eXist project had published plans for XQuery support with availability early in 2006. We then chose eXist as the XML database for the project. As promised, XQuery support was available early in 2006 and has been successfully used in the Provenance project.

The eXist database manages XML documents as a set of collections. A database can contain multiple collections with each collection containing multiple XML documents. Collections can be created and deleted within eXist. For storing Provenance information, a single collection is used called ProvenanceService. All documents within this collection conform to the PStructure type defined in the Provenance architecture. All PStructures contain a unique InteractionKey as specified by the Provenance architecture. The names for the OGSA-DAI data service URI, database resource, and collection name are specified using the GT4 deploy-server.wsdd file as detailed in Section 7 Administration and are set as part of the configuration process.

The following sections detail some of the issues encountered in developing with XML databases which so far are not as well established or mature as relational database technology. For more details of the eXist database, see the paper "eXist: An Open Source Native XML Database" available at http://exist-db.org/webdb.pdf.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

4.6.1 XML Namespaces

XML uses namespaces to provide a means of avoiding naming conflicts when document structures are derived from multiple schemata. An XML document within eXist will contain declarations for each namespace used within the document. This takes the form of a prefix and a namespace value to which the prefix is associated. For example, the p-structure schema used by the Provenance project would have:

Prefix: ps
Value: http://www.pasoa.org/schemas/version025/PStruct.xsd

When the OGSA-DAI client API is used to record, query or update data in an XML database, the namespace prefix and values must be pre-registered with OGSA-DAI. The OGSA-DAI client API supports this for XPath, XQuery and XUpdate. For an XPath query, the prefix and values are set by calling the setNamespaceBinding() method on the XPathQuery object.

To update an existing document using XUpdate, the prefix and value must be included in the XUpdate modifications element as an attribute, a sample is provided:

```
<xu:modifications version="1.0"
     xmlns:xu=http://www.xmldb.org/xupdate
     xmlns:ps="http://www.pasoa.org/schemas/version025/PStruct.xsd">
     <xu:update select="/ps:pstruct/ps:interactionRecord">
      <xu:element name="ps:client">
          <ps:data> some data </ps:data>
      </xu:element>
    </xu:update>
</xu:modifications>
```

Note: The process that led to the creation of this document is described in Section 4.6.3 Insert and Section 4.6.4 Append sections.

4.6.2 Recording

When the record operation on the ProvenanceService is invoked, a Record object is generated by AXIS and is passed to the service. The object contains the data from the recording actor to be recorded and the interface must determine whether to insert the data as a new PStructure document or whether to update an existing document. To do this, the database must be searched to retrieve any PStructure documents that have a key that corresponds to the one we are recording.
The search is performed with an XPath statement of the form:

```
/ps:pstruct[ps:interactionRecord/ps:interactionKey/ps:interactionId ="http://interactionId/URI" and
ps:interactionRecord/ps:interactionKey/ps:messageSource//ps:Addr = "http://messageSource/URI"
and ps:interactionRecord/ps:interactionKey/ps:messageSink//ps:Addr = http://messageSink/URI"]
```

This returns an entire PStructure document if the key is matched, otherwise it will return an empty set. Note that this will return all matching documents, so if more than 1 document has the same key then they will all be returned.

If the result set of the search is empty, then the Provenance Service inserts the PStructure into the database. If the result set is not empty, then it performs an append to the existing PStructure entry.

4.6.3 Insert

Inserting PAssertions into the database is accomplished as follows. A Record object is received by the PStoreDatabase.submit() method. This contains the data from the recording actor to be stored in the database. The IdentifiedContent element (which is a keyed record) is converted into a PStructure object and then written into the database by calling the OGSA-DAI client API. This is shown in the following code fragment:

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

```
// Create a pstruct from the identifiedContent.
PStructure pstruct = make_pstruct(collection, key, iDContent);

// Now serialize the PStructure into an XML string.
String result = Utils.PStructureToXMLString(pstruct);

XMLCreateResource xmlCreateResource = new XMLCreateResource(result);
xmlCreateResource.setParentCollectionName(collection);
service.perform(xmlCreateResource);
ActivityOutput output = xmlCreateResource.getOutput();
String response = output.getData();
```

4.6.4 Append

Appending PAssertions to existing entries in the database are accomplished in a manner similar to that described in the following pseudo-code:

```
Determine the type of View we're updating
Extract the appropriate View Element from the identified content
If we got an element
    We know the view already exists
    Extract all the PAssertions from the identified Content and add them to an array of appends
Else
    We need to create a new View of the appropriate type
    Extract all the PAssertions from the identified Content and add them to the View
    Convert the entire View into a single append entry
EndIf

For Each entry in the append array
   Append the entry to the document in the database
Done
```

5 Security Implementation

This section describes the security features implemented for the Provenance Service. It fulfils the requirements of deliverable D4.3.1 of the contract Technical Annex. As described in the introduction to this document, developments over the last three years have meant that security services in the underlying Grid toolkits can readily be used for developing Grid services. The Provenance project has chosen to use the Globus Toolkit version 4 and so the security support is based on its capabilities.

Security support in the reference implementation of the Provenance Service consists of four parts which will be discussed in the following sub sections. These four parts are:

1. Secure communications between the client and Provenance Service
2. Support for cross domain security
3. Role based Access Control to the Provenance Service
4. Documentation Style supporting p-assertion signing and encryption

5.1 Secure Communications

The reference implementation supports three secure communication options between the CSL and a Provenance Service. The CSL is used to set these options before communicating with the remote service. The options depend upon the container being used to run the service. The possible containers are the basic GT4 container which may also be run within an Apache Tomcat application server. If Tomcat is used, then the CSL supports secure communications using HTTP basic authentication or alternatively HTTPS.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

The CSL includes a number of example clients which are contained in the org/gridprovenance/documentationstyle/test/ folder. The SecurityCommunicationExample.java example shows how the security options are configured by a client using the CSL. The various security communication options are configurable via a concrete class that implements the various accessor and mutator methods for the parameters required (in this case org.gridprovenance.client.policy.PolicyParameter). The individual parameter values for the various secure communication options as listed below are set programmatically as shown in the following Java code snippets:

1. For a Provenance Service deployed in Tomcat, using HTTP Basic-Auth

```
org.gridprovenance.client.policy.PolicyParameter pp = new DefaultPolicyParameter();
pp.setUseBasicAuth(true);
pp.setBasicAuthPassword("wspwd");
pp.setBasicAuthUsername("wsuser");
```

2. For a Provenance Service deployed in Tomcat, using HTTPS

```
org.gridprovenance.client.policy.PolicyParameter pp = new DefaultPolicyParameter();
pp.setUseStandardSSL(true);
pp.setSSLKeystoreName("/home/victor/mykeystore");
pp.setSSLKeystorePassword("victor");
```

Note that the usual port for HTTPS (8443) is different from that of standard HTTP, and the URL for the Provenance Service must contain the appropriate port number.
In order for HTTPS to work, the keystore on the client end (as indicated by the value given to setSSLKeystoreName()) must contain the trusted certificate used to setup SSL on the Tomcat server. The process of importing this certificate is outlined in: http://www.informit.com/articles/article.asp?p=24604&rl=1

3. For a Provenance Service deployed in a basic GT4 container, using WS Secure Conversation.

```
pp.setUseGSISecureConversation(true);
```

For the last case, additional preliminary steps are required to set up a valid user proxy certificate on the CSL. The instructions provided below are applicable for a Linux environment (for both the CSL and the Provenance Service).

1. First, the Provenance Service must be deployed so that it is able to support secure communication with the CSL. In order for this to happen, a Certificate Authority (CA) certificate must be installed on the GT4 container hosting the provenance service. This can either be the SimpleCA package that comes with the complete GT4 distribution, or can be an alternative such an e-Science CA certificate. Instructions for installing SimpleCA should be available from Chapter 7 of the GT4 administrator's guide (http://www.globus.org/toolkit/docs/4.0/admin/docbook/). If an alternative is used instead, the GT4 environment variables must point to this certificate (Chapter 5 of the administrator's guide). Consult the documentation of the Provenance Service for further information
2. A user certificate request must be generated by the user intending to use the CSL to communicate securely with the Provenance Service. This certificate request must subsequently be passed to the administrator of the Provenance Service GT4 container, who will then sign it using the installed CA for that container.
3. The signed user certificate must then be copied to the appropriate location in the user account (usually the .globus subdirectory). Guidelines for generating, signing and installing the user certificate can be found in Chapter 7 of the administrator's guide.
4. A proxy certificate must be generated for the user. This can be done with the command line grid-proxy-init (remember to set the appropriate environment variables first with $GLOBUS_LOCATION/etc/globus-user-env.sh) .

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Once this is complete, the CSL SecureCommunicationExample can then be run. A network/packet sniffer such as Ethereal can be used to monitor the invocation port (e.g. 8080) of the Provenance Service to observe that the SOAP message carrying the recorded message is encrypted using the WS-SecureConversation protocol.

5.2 Cross Domain Authentication

The Security Specification deliverable D4.2.1 discussed the need for cross domain authentication of actors. The reason for this is that the Provenance Architecture can be distributed with Provenance Services deployed in different security domains. Actors, both recording and querying, may require access to these different security domains and therefore they need to be authenticated cross domain. The Security Specification identified the GT4 Community Authorization Service (CAS) as a potential technology for use by the Provenance project. After further investigation, it was found that CAS would not support authentication of Web Services under GT4; this was finally confirmed by an entry in the GT4 CAS Release Notes at http://www.globus.org/toolkit/docs/4.0/security/cas/WS_AA_CAS_Release_Notes.html which stated that "There currently is no support for CAS-based authorization for web services". An alternative technique is to ensure that certificates are cross-certified amongst the collaborating security domains.

GT4 provides implementations of the WS-SecureConversation protocol and WS-Security standard in order to provide message level security mechanisms. Incoming SOAP requests to a service deployed in a secure mode within a GT4 container needs to be encrypted and/or signed appropriately in compliance with the standard. The root certificate for the Certificate Authority (CA) of the domain in which the container is running in, is used in the verification of the signatures on these incoming requests. The user certificates issued to the various clients making these requests will therefore need to be signed by that particular CA.

Each independent security domain maintains its own root CA certificate; therefore incoming requests from clients in a different domain cannot be authenticated since their certificates were signed or issued by a different CA. In order to support inter-domain authentication at the level of the GT4 message level security framework, two approaches are possible:

1. If there are N different security domains, then the root certificates of all the domains are installed into each GT4 container in the different domain. This means that each domain knows the root certificate of all other domains and can authenticate requests coming from them. The GT4 administrator's guide (http://www.globus.org/toolkit/docs/4.0/admin/docbook/ch05.html) explains how a GT4 container can be set up to trust certain CAs by installing the required information into specific configuration directories.
2. Using cross-certificates. A cross-certificate is a certificate issued by one CA that signs the public key for the root certificate of another CA. Cross-certificates provide a means to create a chain of trust from a single, trusted, root CA to multiple other CAs. The process of cross signing certificates is dependent on the particular CAs involved in the cross signing process. An example guideline to how this is achieved in the Globus environment is provided at (https://www.pki.virginia.edu/nmi-bridge/)

5.3 Role Based Access Control

Access to a Provenance Service can be controlled by an Access Control List (ACL) which accepts or rejects incoming SOAP messages based on their credentials. The ACL is an XML file which allows or restricts user requests to specific operations of the Provenance Service. The ACL is loaded by the GT4 container at startup and each SOAP request to the Provenance Service is validated using the GT4 extensible Policy Decision Point (PDP) mechanism. For further details of the GT4 Authorization services, see the tutorial article at http://www-128.ibm.com/developerworks/grid/library/gr-gt4auth/.

The Provenance Service ACL is defined in an XML file, an example can be found at /etc/provenance-authfile.xml in the Provenance Service distribution. Instructions for deploying the ACL and other security configuration are found in the Install.txt file.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

The contents of the ACL should be set prior to deploying the Provenance Service. The contents of the ACL is read and stored in memory when the GT4 container is first started, after which no further reference is made to the file. An example of the contents of the ACL file is shown below:

```
<authlist>
   <permission identity = "EMAILADDRESS=client1@hotmail.com, CN=client1, OU=UK, O=UK, L=none, ST=UK, C=UK"
               operationList = "createResource // destroy"
               mode = "allow" />
</authlist>
```

Here, the request originating from an entity with the X509 Distinguished Name of "EMAILADDRESS=client1@hotmail.com, CN=client1, OU=UK, O=UK, L=none, ST=UK, C=UK" will be executed if either the createResource or destroy service operations are requested. The listed operations are from those specified in the Provenance Service WSDL, and are separated by the separator token //. The mode "allow" indicates that these are the only authorized actions. The other option for mode, "restrict", would indicate otherwise, i.e. that all operations are authorized except the ones listed in the operationList parameter.

The ACL provides support for grouping of operations and identities in order to provide for a basic implementation of role-based access control. An example is shown below:

```
<authlist>
   <groupIdentity groupName = "admin"
                  identityList = "EMAILADDRESS=client1@hotmail.com,CN=client1,OU=UK,O=UK,L=none,ST=UK,C=UK //
                                  EMAILADDRESS=client2@hotmail.com,CN=client2,OU=UK,O=UK,L=none,ST=UK,C=UK"/>

   <groupIdentity groupName = "staff"
                  identityList = "EMAILADDRESS=client3@hotmail.com,CN=client3,OU=UK,O=UK,L=none,ST=UK,C=UK //
                                  EMAILADDRESS=client4@hotmail.com,CN=client4,OU=UK,O=UK,L=none,ST=UK,C=UK"/>

<groupOperation groupName = "all"
                   operationList = " createResource // destroy // Record // ProvenanceQuery "/>

   <permission identity= "admin"
               operationList = " destroy "
               mode = "restrict"/>

   <permission identity = "staff"
               operationList = "all"
               mode = "allow"/>

</authlist>
```

In this example, we have two additional elements, groupIdentity and groupOperation. The groupIdentity element is used to group X509 distinguished names into a specific group or role; for example, we could classify all identities into either administration ("admin") or normal staff ("staff"). Similarly, we can group together operations that are logically related such as the createResource, destroy, Record and ProvenanceQuery. These groups can then be used when specifying the actual authorizations in the permission element. Thus, it becomes easier to specify higher level authorizations, such as: "All administration is allowed to perform all operations except destroy" or "All staff are only permitted to perform basic operations on a provenance Service". Groups can be nested and specified alongside atomic elements, thus the following elements are valid:

```
   <groupIdentity groupName = "everyone" identityList = "staff // admin "/>
   <groupOperation groupName = "allOperations" operationList = " save //  find //  get "/>
   <groupOperation groupName = "selectedOperations" operationList = " find_binding // get_bindingDetail // save "/>
   <permission identity = "everyone" operationList = "allOperations" mode = "allow"/>
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Any number of entries can be specified in the ACL for the three elements permission, groupOperation and groupIdentity. If there is more than one element with the same name, their contents will be merged automatically.

5.4 Documentation Style

The Provenance Architecture [1] describes a data model for the process execution documentation stored by the Provenance Service. In section 6.5, it describes the Documentation Style which is a mechanism that allows actors to transform the content of a p-assertion when it is recorded and when it is retrieved. This architectural feature is supported by a documentation style library within the CSL.

The documentation style library is an additional component of the CSL that is used to transform inputs (these can be messages or actor states) in some manner before they are put into a p-assertion. In the simplest case, a message or actor state of interest is put as it is (verbatim) into a p-assertion by an asserting actor. However, transformations are useful in other circumstances, for example:

1. There may be a need to obscure information in an input before it is put into a p-assertion to ensure that querying actors cannot access this information

2. A message may potentially contain large amounts of information, which in turn increases the size of the p-assertion it is put into and subsequently, the load on the provenance store. This information can be removed and stored elsewhere, and a reference to its location put into the documentation store.

3. There may be a need to assign liability for certain parts of a message to certain actors. These parts of a message can then be signed first before putting into a p-assertion.

There is clearly a wide range of possible transformations possible, and this would very likely be dependent on the application domain that a provenance system is deployed in. The documentation style library contains a number of basic transformations, but is designed to be extensible to accommodate developers creating their own customized application-specific transformations. The Documentation Style features of the Provenance Architecture have been documented as part of the Open Provenance Specification.

Generating Digital Signatures for the contents of a p-assertion has been implemented in the CSL using the Documentation Style facilities since signing the contents of a p-assertion is an example of the type of transformation actors may wish to perform before recording the p-assertion. To perform a transformation, a transformation definition document is required. This is an XML file that provides information on how a specific transformation is to be performed. Each transformation will have a set of parameters associated with it; the transformation is performed using the XML file as input to supply values for these parameters. The transformed input is put into a p-assertion along with a URI pointing to the transformation definition document, which should now be put on a publicly accessible site.

A querying actor that subsequently retrieves this p-assertion can also locate the transformation definition document associated with it through this embedded URI. This document can then be used to understand what transformation was used to produce the contents of the p-assertion. Often (although not necessarily always), it is possible as well to perform a reverse transformation on the p-assertion content to obtain the original message/actor state. The querying actor can then elect to use this original information in some application specific manner.

For some transformations, extra information is required in addition to those provided in the transformation definition document in order to perform a transformation. This information, is referred to as private parameters to contrast it against the information in the transformation definition document, which is publicly available. These parameters must be instantiated by an actor in its local environment, and will hence assume different values for different environments. As an example, the signature transformation operation requires the value of the signature algorithm to be used (which can be specified in the transformation definition document) as well as the private key of the actor (which is specified as a private parameter).

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

As private parameters are not shared, it is the responsibility of asserting actors to propagate the values for these to querying actors if such need arises, and such propagation should happen outside the context of the provenance system. As an example, for a querying actor to be able to decrypt an encrypted content it would require the original encrypting key used by the asserting actor. This key will obviously not be exposed in the transformation definition document; instead it must be passed on to the querying actor concerned.

The schema for the transformation definition document is located in docstyle\docstyledata\schema and also in DocStyleData\example. Note that this schema differs slightly from the schema presented in the standardization document as this is designed to support JAXB class generation in Java 1.4.

5.4.1 Available Transformations

Currently there are six types of transformations supported in the CSL, all assume XML input documents and produced XML documents as outputs. Examples of transformation definition documents for all of these transformations can be found in the CSL release under the DocStyleData\example folder.

1. Verbatim: The default null transformation, i.e. nothing is transformed (verbatimtransform.xml)
2. Signing: A part of the input (specified by an XPath search expression) is signed using XML Signature standards (signaturetransform.xml)
3. Encryption: A part of the input (specified by an XPath search expression) is encrypted using XML Encryption standards (encryptiontransform.xml)
4. Reference: A part of the input (specified by an XPath search expression) is removed and put into a different location, and the removed part in the input is replaced with a reference URI to this different location. A digest may be computed on the removed part (referencetransform.xml)
5. ReplaceElement: A part of the input (specified by an XPath search expression) is removed and replaced with a String or an XML fragment (replacetransform.xml)
6. CompositeSequence: This is essentially a transformation that is a collection of any of the 5 previous transformations. The input will be passed through a sequence of given transformations, where the output from one transformation becomes the input to the next transformation in the sequence (complextransform.xml)

With the exception of Verbatim, the source code for all these various implementations can be found in the CSL release in the .\docstyle\src folder. There is also a build file in .\docstyle which contains ant targets to build these various implementations into individual jar files. These jars can subsequently be dropped in the execution classpath of applications that need to use them, and the implementations themselves must be registered programmatically in the code of the application.

6 Scalability Implementation

This section describes the scalability features implemented for the Provenance Service. It fulfils the requirements of deliverable D5.3.1 of the contract Technical Annex. As described in the introduction to this document, developments over the last three years have meant that Grid toolkits can readily be used for developing scalable Grid services.

Deliverable D5.2.1 made a number of recommendations about how architectural features could be implemented. These recommendations were analysed against the application requirements. We concluded that some aspects although desirable, did not meet the needs of the demonstration applications from workpackages 7 and 8. Therefore the following specification recommendations were not implemented:

1. Section 5.1.3 – Repeated p-assertions. Not required.
2. Section 5.1.4 – Import and Export. Whilst not specifically implemented as ports, the function is available using the features of the eXist database used to implement the persistent store.
3. Section 5.1.5 – Non-XML Data. Not required.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

4. Section 5.2.1 – Clustering and Non-Affinity. Supported by the GT4 toolkit which can be deployed in clustered configurations using Apache Tomcat. The Provenance Service is addressed by a single network endpoint which meets the non-affinity requirements.
5. Section 5.2.2 – Persistent Stores. The Provenance Service implementation uses OGSA-DAI data services which support persistent stores.

The only part of the Scalability specification implemented during this period was the support for large query result sets identified in section 5.1.1 of deliverable D5.2.1. This was implemented as the XPathFactoryPort in the Provenance Service interface. It allows the results of a query to be cached within the Provenance Service and retrieved using an iteration mechanism.

7 Administration and Configuration

This section describes the setup and configuration of the Provenance Service and its associated components. Installation instructions for all components are available in the Install.txt file shipped with the reference implementation release. To make installation easier for administrators, the reference implementation comes with a comprehensive set of Ant scripts to manage the building and deployment into GT4 of the code. Scripts are also provided for managing the eXist database ant its collections. See the build.xml file provided with the release.

7.1 Pre-requisite components

This Provenance Service reference implementation makes use of the following components:

1. Java 1.4.2 or higher - http://java.sun.com (or IBM Java 1.4.2).
2. Apache Ant 1.5 - http://ant.apache.org
3. Globus Toolkit WSRF-compliant WS java container version 4.0.1 http://www.globus.org/toolkit/downloads/4.0.1/#wscore_bin
4. OGSA-DAI version OGSA-DAI WSRF 2.1 http://www.ogsadai.org.uk
5. eXist 1.0 latest snapshot - http://exist.sourceforge.net

These have to be installed and configured as part of the Provenance Service setup process.

7.2 Other Externals and Dependencies

The reference implementation relies on the OGSA-DAI Client API and the Apache AXIS API in addition to GT4. The configuration of these components is done through files; examples of these are provided with the software release.

Axis is configured using three files:

1. deploy-server.wsdd defines the set of services exposed by this implementation
2. server-config.wsdd created during the deployment using data from deploy-server.wsdd
3. client-config.wsdd defines the client-side handlers for GT4 security

OGSA-DAI is configured using two files:

1. data.service.resource.properties defines resource identity, user credentials and eXist server parameters
2. DatabaseRoles.xml defines database credentials

GT4 is configured using four files:

1. NStoPkg.properties defines the mapping between XML namespaces and generated Java classes
2. deploy-jndi-config.xml defines the service resource parameters used by the Provenance Service
3. jndi-config.xml generated from deploy-jndi-config.xml during service deployment
4. container-log4j.properties defines log4j properties for use once GT4 container has started

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Bibliography and External Web Links

[1] Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. D3.1.1: An architecture for provenance systems. Technical report, University of Southampton, February 2006. http://eprints.ecs.soton.ac.uk/12023/.

[2] Sheng Jiang, Luc Moreau, Paul Groth, Simon Miles, Steve Munroe, and Victor Tan. D9.3.3a: Client side library design and implementation. Technical report, University of Southampton, November 2006. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-ClientSideLibrary-D933a-Month27.pdf

[3] John Ibbotson, Neil Hardman, and Victor Tan. D4.1.1: Security specification - version 1. Technical report, IBM United Kingdom, September 2005. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-SecuritySpecV1-D411-Month12.pdf.

[4] John Ibbotson, Neil Hardman, and Victor Tan. D4.2.1: Security specification - version 2. Technical report, IBM United Kingdom, February 2006. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-SecuritySpecV2-D421-Month18.pdf.

[5] John Ibbotson, Paul Groth, and Simon Miles. D5.1.1: Scalability specification - version 1. Technical report, IBM United Kingdom, September 2005. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-ScalabilitySpecV2-D421-Month18.pdf.

[6] John Ibbotson. D5.2.1: Scalability specification – version 2. Technical report, IBM United Kingdom, February 2006. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-ScalabilitySpecV2-D521-Month18.pdf.

[7] OASIS Web Services Resource Framework: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

[8] AXIS; the Apache Web Services implementation http://ws.apache.org/axis

[9] DOM; the W3C Document Object Model http://www.w3.org/DOM/

[10] eXist; An Open Source XML database http://exist.sourceforge.net/

[11] GT4; The Globus Toolkit version 4 http://www.globus.org

[12] OGSA-DAI; Open Source implementation of the OGSA Data Access and Integration specification http://www.ogsadai.org

[13] Tomcat; the Apache Java application server http://tomcat.apache.org/

[14] XPath; the W3C XML Path language http://www.w3.org/TR/xpath

[15] XQuery; the W3C XML Query Language http://www.w3.org/XML/Query/

[16] XUpdate; an XML language for updating documents and databases http://xmldb-org.sourceforge.net/xupdate/

[17] John Ibbotson, Neil Hardman, and Alexis Biller. D9.3.2: Functional prototype (public release). Technical report, IBM United Kingdom, February 2006. http://www.gridprovenance.org/deliverables/GRID_PROVENANCE-FunctionalPrototype-D932-Month18.pdf.

The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

Appendix A External Interfaces

This appendix described the external service interfaces to the Provenance Service. For brevity, the WSDL is provided with the accompanying XML Schema files omitted. These XML Schemas are provided with the implementation package available from the Provenance project website.

A.1 The ProvenanceStoreFactory Interface

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) Copyright International Business Machines Corporation 2005. -->
<!--              See CPL.txt for licensing information.             -->

<definitions name="ProvenanceStoreFactory"
             targetNamespace="http://www.gridprovenance.org/namespaces/ProvenanceStoreFactory"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:tns="http://www.gridprovenance.org/namespaces/ProvenanceStoreFactory"
             xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!-- T Y P E S -->
    <types>
        <xsd:schema targetNamespace="http://www.gridprovenance.org/namespaces/ProvenanceStoreFactory"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    xmlns:tns="http://www.gridprovenance.org/namespaces/ProvenanceStoreFactory">

            <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/03/addressing"
                        schemaLocation="../ws/addressing/WS-Addressing.xsd" />

            <!-- REQUESTS AND RESPONSES -->
                <xsd:element name="createResource">
                     <xsd:complexType/>
                </xsd:element>
                <xsd:element name="createResourceResponse">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref="wsa:EndpointReference"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
        </xsd:schema>
    </types>

    <!-- M E S S A G E S -->
    <message name="CreateResourceRequest">
        <part name="request" element="tns:createResource"/>
    </message>
    <message name="CreateResourceResponse">
         <part name="response" element="tns:createResourceResponse"/>
    </message>

    <!-- P O R T T Y P E -->
    <portType name="ProvenanceStoreFactoryPortType">
        <operation name="createResource">
            <input message="tns:CreateResourceRequest"/>
            <output message="tns:CreateResourceResponse"/>
        </operation>
    </portType>
</definitions>
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

A.2 The ProvenanceService Interface

```
<?xml version="1.0"?>
<definitions name="ProvenanceService"
    targetNamespace="http://www.gridprovenance.org/namespaces/version025/ProvenanceService.wsdl"
    xmlns:tns="http://www.gridprovenance.org/namespaces/version025/ProvenanceService.wsdl"
    xmlns:prwsdl="http://www.pasoa.org/schemas/version025/record/PRecordSOAPBinding.wsdl"
    xmlns:xqwsdl="http://www.pasoa.org/schemas/version025/xquery/XQuerySOAPBinding.wsdl"
    xmlns:xpwsdl="http://www.gridprovenance.org/namespaces/version025/xpath/XPathSOAPBinding.wsdl"
    xmlns:xpfwsdl=http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIteratorSOAPBinding.wsdl
    xmlns:pqwsdl="http://www.pasoa.org/schemas/version025/pquery/PQuerySOAPBinding.wsdl"
    xmlns:rewsdl="http://www.gridprovenance.org/namespaces/ProvenanceResources/bindings"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

    <import namespace="http://www.pasoa.org/schemas/version025/record/PRecordSOAPBinding.wsdl"
            location="./record/PRecordSOAPBinding.wsdl" />
    <import namespace="http://www.pasoa.org/schemas/version025/xquery/XQuerySOAPBinding.wsdl"
            location="./xquery/XQuerySOAPBinding.wsdl" />
    <import namespace="http://www.gridprovenance.org/namespaces/version025/xpath/XPathSOAPBinding.wsdl"
            location="./xpath/XPathSOAPBinding.wsdl" />
    <import namespace="http://www.pasoa.org/schemas/version025/pquery/PQuerySOAPBinding.wsdl"
            location="./pquery/PQuerySOAPBinding.wsdl" />
    <import namespace="http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIteratorSOAPBinding.w
sdl"
            location="./xpathIterator/XPathIteratorSOAPBinding.wsdl" />
    <import namespace="http://www.gridprovenance.org/namespaces/ProvenanceResources/bindings"
            location="./wsrf/ProvenanceResources_bindings.wsdl" />

    <!-- Defines the WSDL interface to a particular instantiation of the Provenance Store.
         Each instantiation would offer its own version of this file.  -->
    <service name="ProvenanceService">
        <port name="RecordPort" binding="prwsdl:RecordSOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/record" />
        </port>
        <port name="XQueryPort" binding="xqwsdl:XQuerySOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/xquery" />
        </port>
        <port name="XPathPort" binding="xpwsdl:XPathSOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/xpath" />
        </port>
        <port name="XPathFactoryPort" binding="xpfwsdl:XPathIteratorSOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/xpathiterator" />
        </port>
        <port name="ProvenanceQueryPort" binding="pqwsdl:PQuerySOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/pquery" />
        </port>
        <port name="ProvenanceResourcesPort" binding="rewsdl:ProvenanceResourcesPortTypeSOAPBinding">
            <soap:address location="http://localhost:8080/wsrf/services/ProvenanceService/provenanceresource" />
        </port>
    </service>
</definitions>
```

A.3 The PRecord WSDL

```
<?xml version="1.0"?>
<definitions name="PRecord"
             targetNamespace="http://www.pasoa.org/schemas/version025/record/PRecord.wsdl"
             xmlns:tns="http://www.pasoa.org/schemas/version025/record/PRecord.wsdl"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:pr="http://www.pasoa.org/schemas/version025/record/PRecord.xsd">

    <import namespace="http://www.pasoa.org/schemas/version025/record/PRecord.xsd"
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

```
            location="./PRecord.xsd" />

    <message name="Record">
        <part name="body" element="pr:record"/>
    </message>
    <message name="RecordAck">
        <part name="body" element="pr:recordAck"/>
    </message>

    <portType name="RecordPortType">
        <operation name="Record">
            <input message="tns:Record"/>
            <output message="tns:RecordAck"/>
        </operation>
    </portType>
</definitions>
```

A.4 The XQuery and XPath WSDL

```
<?xml version="1.0"?>
<definitions name="XQuery"
             targetNamespace="http://www.pasoa.org/schemas/version025/xquery/XQuery.wsdl"
             xmlns:xs="http://www.w3.org/2001/XMLSchema"
             xmlns:tns="http://www.pasoa.org/schemas/version025/xquery/XQuery.wsdl"
             xmlns:xq="http://www.pasoa.org/schemas/version025/xquery/XQuery.xsd"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

    <import namespace="http://www.pasoa.org/schemas/version025/xquery/XQuery.xsd" location="./XQuery.xsd"/>

    <message name="Query">
        <part name="body" element="xq:query"/>
    </message>

    <message name="QueryAck">
        <part name="body" element="xq:queryAck"/>
    </message>

    <portType name="XQueryPortType">
        <operation name="Query">
           <input message="tns:Query"/>
           <output message="tns:QueryAck"/>
        </operation>
    </portType>
</definitions>
```

```
<?xml version="1.0"?>
<definitions name="WSXPath"
       targetNamespace="http://www.gridprovenance.org/namespaces/version025/xpath/XPath.wsdl"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:tns="http://www.gridprovenance.org/namespaces/version025/xpath/XPath.wsdl"
       xmlns:xp="http://www.gridprovenance.org/namespaces/version025/xpath/XPath.xsd"
       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns="http://schemas.xmlsoap.org/wsdl/">

       <documentation>
        Description: An XPath interface for Web Services
        Author: John Ibbotson
        Last Modified: 12 June 2006

        Copyright (c) 2006 IBM United Kingdom Limited
       </documentation>

       <import
              namespace="http://www.gridprovenance.org/namespaces/version025/xpath/XPath.xsd"
              location="./XPath.xsd" />
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme

```
        <message name="XPath">
                <part name="xpathbody" element="xp:xpathquery" />
        </message>

        <message name="XPathAck">
                <part name="xpathbody" element="xp:xpathqueryAck" />
        </message>

        <portType name="XPathPortType">
                <operation name="XPathQuery">
                        <input message="tns:XPath" />
                        <output message="tns:XPathAck" />
                </operation>
        </portType>
</definitions>
```

A.5 The XPathIterator WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) Copyright International Business Machines Corporation 2005. -->
<!--              See CPL.txt for licensing information.             -->

<definitions name="XPathIterator"
        targetNamespace="http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIterator.wsdl"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:tns="http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIterator.wsdl"
        xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xp="http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIterator.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <import namespace="http://www.gridprovenance.org/namespaces/version025/xpathIterator/XPathIterator.xsd"
            location="./XPathIterator.xsd" />

    <!-- M E S S A G E S -->
    <message name="XPathQueryFactoryRequest">
        <part name="body" element="xp:xpathqueryFactory" />
    </message>

    <message name="XPathQueryFactoryResponse">
        <part name="body" element="xp:xpathqueryFactoryAck" />
    </message>

    <message name="GetItemsRequest">
        <part name="body" element="xp:getItemsRequest" />
    </message>

    <message name="GetItemsResponse">
        <part name="body" element="xp:getItemsResponse" />
    </message>

    <!-- P O R T T Y P E -->
    <portType name="XPathFactoryPortType">
        <operation name="XPathQueryFactory">
            <input message="tns:XPathQueryFactoryRequest" />
            <output message="tns:XPathQueryFactoryResponse" />
        </operation>
        <operation name="GetItems">
            <input message="tns:GetItemsRequest" />
            <output message="tns:GetItemsResponse" />
        </operation>
    </portType>
</definitions>
```


The PROVENANCE project receives research funding from the European Commission's Sixth Framework Programme