



Project no: **RI031844-OMII-Europe**

Project acronym: **OMII-Europe**

Project title: **Open Middleware Infrastructure Institute for Europe**

Instrument: **Integrated Infrastructure Initiative**

Thematic Priority: **Communication network development**

M:JRA2.2
Design for a Generic Information Provider Framework

Due date of milestone: 31 December 2007

Actual submission date: 7 April 2008

Start date of project: **1 May 2006**

Duration: **2 years**

INFN

Revision [final]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Control Sheet

Document	Title: Report on Grid Activities relevant to the identification of new services	
	ID: M:JRA2.2	
	Version: 1	Status: final
	Available at: http://omii-europe.org	
	Software Tool: Microsoft Word 2003	
	File(s): MJRA2.2.doc and MJRA2.2.pdf	
	Written by:	Sergio Androozzi (INFN), Marco Canaparo (INFN), Michele Carpenè (INFN)
	Contributors:	INFN
	Reviewed by:	Project Management
Approved by:	Project Management Committee	

Document Status Sheet

Version	Date	Status	Comments
1	21 March 2008	Ready for internal review	
2	7 April 2008	Applied comments from internal reviewers	

Executive Summary

During the second year of the project, the JRA2 activity is working on defining and implementing an information model to describe Grid resources. The commitment for this work is based on the first year activity and on the mandate given by the Project Management Committee [DJRA2.1]. The decomposition of this work envisions two main contributions: 1) GLUE 2.0: a community-agreed information model defined in the context of the Open Grid Forum; 2) GLUEMan: an implementation of a generic information providers framework which supports the management of providers for GLUE 2.0. This document describes part of the work performed from M13 to M18 related to the design of GLUEMan. In the same period, part of the effort was dedicated to the definition of the GLUE 2.0 standard by co-leading and contributing to the OGF GLUE Working Group. The target audience for this document is project managers that have to make decision for adoptions, commissioner, and also developers. It can also be used by developers who want to have insight about the design choices and researchers in the area of information modeling and resource management.

The design of GLUEMan presented in this document is the base for its implementation to be performed in the last period of the project (M19-M24). The most relevant design choices are: leverage the DMTF WBEM standard suite and the related implementations; be generic in order to be easily extensible to support the evolution of information models and related extensions.

The implementation of GLUEMan that is being developed based on this design is an independent software module which can be easily integrated in several Grid middleware suites and is not necessarily bound to those part of the OMII-Europe project.

Table of Contents

- 1 Introduction..... 5
- 2 Background..... 5
- 3 GLUEMan: a Generic Framework for GLUE Information Providers 6
 - 3.1 Requirements 7
 - 3.2 Design..... 8
 - 3.2.1 Proxy Provider 8
 - 3.2.2 Client 9
 - 3.3 Relationship to the OGSA-BES Specification..... 10
- 4 Conclusion 11
- A References..... 11

1 Introduction

During the first year of the project, the Joint Research Activity (JRA) 2 team performed an analysis and comparison of different Grid middlewares (gLite, Globus, UNICORE, VEGA-GOS, CROWNGRID, OMII-UK, ARC) based on a decomposition of the individual capabilities following the OGSA conceptual architecture [OGSA]. Given this comparison, JRA2 identified a number of capabilities that could be considered for implementation during the second year of the project. A prioritization based on cost-risk-value was proposed in order to support the decision for the allocation of an extra-budget to be invested during the second year of the project [DJRA2.0].

In a second deliverable [DJRA2.1], the JRA2 team presented a summary of the analysis and prioritization activity performed in the first six months activity [DJRA2.0] and the proposed work-plans for the top three capabilities emerged from the former analysis. The final decision was to select the information model activity for the budget allocation. Such a budget was assigned to INFN for performing the defined work-plan in the context of JRA2.

The decomposition of this work envisions five main contributions: (1) participation to the OGF GLUE WG for the refactoring of the GLUE Schema and its standardization; (2) implementation of the OGF GLUE 2.0 specification in the XML Schema data model and inclusion into the OMII-Europe repository; (3) implementation of a generic information provider framework for managing GLUE 2 providers; (4) implementation of a client enabling to query GLUE 2 information and render them according to the XML Schema mapping of the conceptual model; (5) implementation of providers for the BES implementations developed within the OMII-Europe project.

In this document, we present the design of both the generic information provider framework for managing GLUE 2 providers (4) and of the client enabling to query GLUE 2 information and render them according to the available mappings (5). Both components will be provided within a single software project that we call GLUEMan [GLUEMan].

The document is structured as follows. In Section 2, we describe the background of this work; in Section 3, we present the requirements and the design choices for GLUEMan together with the integration points with the BES specification; in Section 4 we draw up the conclusions and the future work.

2 Background

There are two main activities that should be considered in order to understand the context and the design choices of this work. One is the OGF GLUE Working Group (WG) activity and the other is the Distributed Management Task Force (DMTF) organization defining the WBEM (Web-Based Enterprise Management).

The OGF GLUE (WG) was created during the OGF 19 in order to provide a recommendation for an abstract information model of Grid resources and services. The goal is to facilitate information interoperability between Grid infrastructures. The focus is on use cases which span multiple Grid infrastructures that may rely on different middleware suites. The existing models such those defined by the GLUE Schema 1.3 and by the NorduGrid schema are taken into account. The JRA2 team is contributing to this important activity by co-leading the whole definition and standardization process.

The DMTF is an industry-supported organization which develops management standards and promotes interoperability for enterprise and Internet environments. The main standard set that is interesting for this work is WBEM, a set of management and internet standard technologies developed to unify the management of distributed computing environments. WBEM provides the ability for the industry to deliver a well-integrated set of standard-based management tools, facilitating the exchange of data across otherwise disparate technologies and platforms.

The standards included in WBEM are the Common Information Model (CIM), CIM-XML, CIM Query Language, WBEM Discovery using Service Location Protocol (SLP) and WBEM Universal Resource Identifier (URI) mapping. In addition, the DMTF has developed a WBEM Management Profile template, allowing for simplified profile development to deliver a complete, standalone definition for the management of a particular system, subsystem, service or other entity.

The two relevant specifications for this activity are CIM (Common Information Model) and CIM Over HTTP. The former is a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. The OGF GLUE 2.0 specification potentially could become in the future an extension of CIM (there are actual plans based on the OGF-DMTF collaboration channels). The current interest for the time being is for the text-based representation of CIM classes and associations using the MOF (Managed Object File) format. CIM Over HTTP defines a mapping of CIM Messages onto HTTP that allows implementations of CIM to operate in an open, standardized manner.

3 GLUEMan: a Generic Framework for GLUE Information Providers

The GLUE 2.0 conceptual model will be adopted by many Grid projects in order to advertise the capabilities of the resources and services available in the various Grid infrastructures. The information providers to measure the defined attributes for the numerous implementations of Grid services have to be written. They are typically coded by the developers of the services themselves, since they know the best way to extract the information. Persons writing information providers need the support of some framework where to plug their sensors and which checks the conformance of the measured data against the normative specification.

On the other side, information consumers such as resource users (high-level services, end-users) or site administrators, need a common way to access the GLUE-based information. There are a number of potential formats that may require.

Our main approach in solving this issue is to leverage the WBEM standard suite and the related open source implementations in order to build this framework. We performed an analysis of the available solutions and we opted to build GLUEMan on top of Open Pegasus [PEG] which is an open-source implementation of the DMTF CIM and WBEM standard designed to be portable and highly modular. It is coded in C++ so that it effectively translates the object concepts of the CIM objects into a programming model but still retains the speed and efficiency of a compiled language. As a complementary component, we have also selected the CIMPLE library [CIM] which provides a CIM Provider Engine to simplify provider development and to make providers portable across different WBEM implementations.

3.1 Requirements

In the design of GLUEMan, we have considered a number of requirements that we classify in three main categories: general requirements that apply to the whole system; requirements for the proxy provider and requirements for the client.

In the category of general requirements, we have identified:

- *isolate developers from WBEM-specific detail*: WBEM standards and related technologies have an inherent steep learning curve in order to be adopted and extended to support new information models; our goal is to isolate the developers of providers and the users of clients from the WBEM complexity
- *read-only providers*: WBEM supports three main types of providers, instance providers to create instances of properties defined in UML classes, association providers to create instances of UML associations, and method providers to support the invocation of methods defined in UML classes; considering that we only expose information, but do not address the issue of controlling the managed resource, we have only read-only providers (instance and association providers)
- *support any programming language*: Open Pegasus is coded in C++ and this is the natural language to be used for providers; typically, providers are written in scripting languages (such as Python, Perl or bash) for rapid prototyping and for easy of customization by sys admins in case of specific deployments; there should be no constraint on the required programming language for the providers
- *Reduce intrusiveness*: for each received query, Open Pegasus invokes the related providers and returns the obtained values formatted as CIM instances; this should be avoided because providers may be inherently intrusive versus the observed entity; the natural approach is to introduce a caching layer

In the category of requirements specific to the provider, we have identified:

- *enforce strong data conformance checking*: the providers for GLUE 2.0 should generate information that conform to the OGF normative specification; the framework should perform a conformance check of data from real providers before accepting them
- *easy the writing of configuration-based information*: properties of a class typically belong to two main categories; configuration based information and state; the former are often written by persons and it should be easy for them to set values for these types of properties

In the category of the requirements specific to the client, we have identified:

- *support multiple output renderings*: the GLUE 2.0 abstract model will be rendered in several concrete data models (initially XML Schema, LDAP and SQL); such formats should be supported by the client
- *easy the addition of new renderings*: the client design should consider modularity as regards the renderings of the output in order to simply the work from third-parties who may want to add new renderings

3.2 Design

In this section, we present the design choices together with the UML diagrams of GLUEMan. In Figure 1, we can see a simplified view of the GLUEMan components: a proxy provider and a client. These components will be described in the following sections.

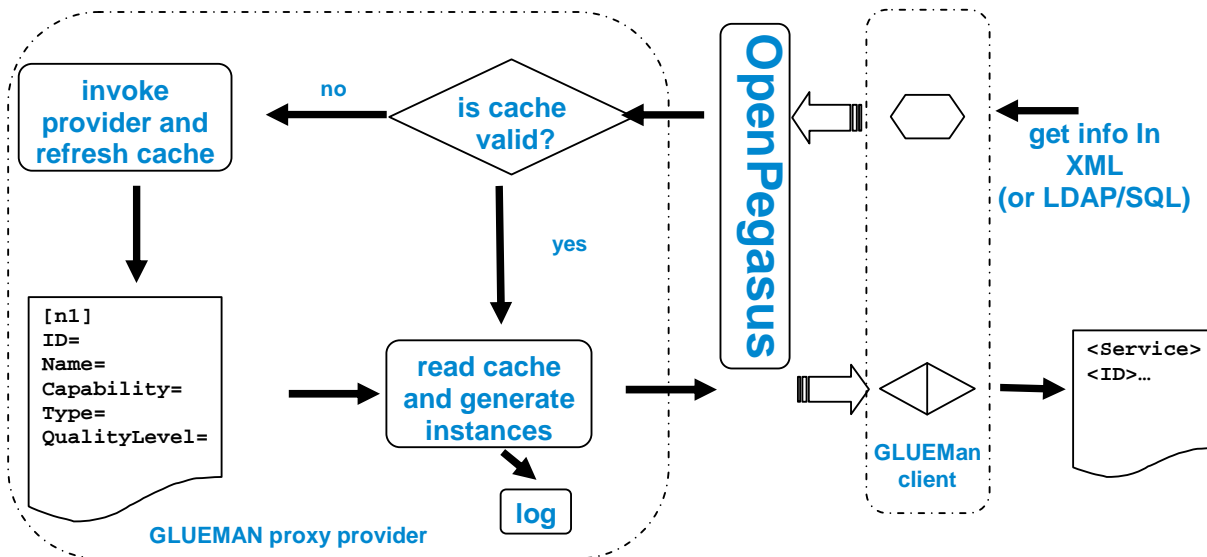


Figure 1 Simplified View of GLUEMan Components

3.2.1 Proxy Providers Module

As mentioned above, in Open Pegasus, each provider is related to a class or association. Each real provider is responsible therefore for generating a number of instances for a certain class/association. Open Pegasus also favours providers written in C++, even though decoupling from the provider programming language is available by using the CMPI standard binary interface (Common Manageability Programming Interface).

The main design pattern used to address the provider requirements is the proxy design pattern. For each provider related to the GLUE information model, we define a proxy provider decoupling the interaction between the Open Pegasus server and the real provider. The proxy provider offers three main extra-functionalities: 1) invocation of the real provider written in any language and communication via the standard output; 2) caching of the result (the expiration time can be configured for each individual provider); 3) conformance check to the GLUE 2.0 specification.

The communication between the real provider and the proxy provider is performed via the standard output. The selected format for exchanging data is the INI format [INI]. The motivations for this choice are: 1) the INI format is simple, there are many parser in every language to handle this format and if no parser are expected to be used, it is pretty simple to generate it; 2) the complexity of the output is suitable for the INI format (list of instances of classes/associations); 3) the validation of the data is performed by the proxy provider.

All the proxy providers are packaged in a single software component written in C++ and compiled as a shared library which exposes a CMPI interface to the Open Pegasus server. This library contains as many providers as the classes and associations defined in GLUE 2.0.

When a query for a certain class arrives to the Open Pegasus server, this will invoke the related proxy provider. The proxy provider will read a configuration file where the relevant parameters to the real provider are maintained. Among these parameters, the cache time out is checked. If there is a previous output of the run of the real provider, then such an output will be used to create the instances of GLUE 2.0 information within the Open Pegasus server, otherwise the real provider will be invoked.

3.2.2 Client

The client design is based on two main design patterns: the Model-View-Controller pattern and the Strategy pattern. As regards the Model-Controller-View: the Model component is represented by CIM Classes and Instances, the View component is represented by the GLUE 2.0 renderings, and the Controller component is represented by the navigation strategy through GLUE instances.

The Strategy design pattern is used in order to handle the different renderings of the data acquired by Open Pegasus server. This pattern enables to isolate them and to easy the addition of new renderings.

In Figure 2, we present a UML class diagram showing the design of the main classes of the client according to the defined patterns.

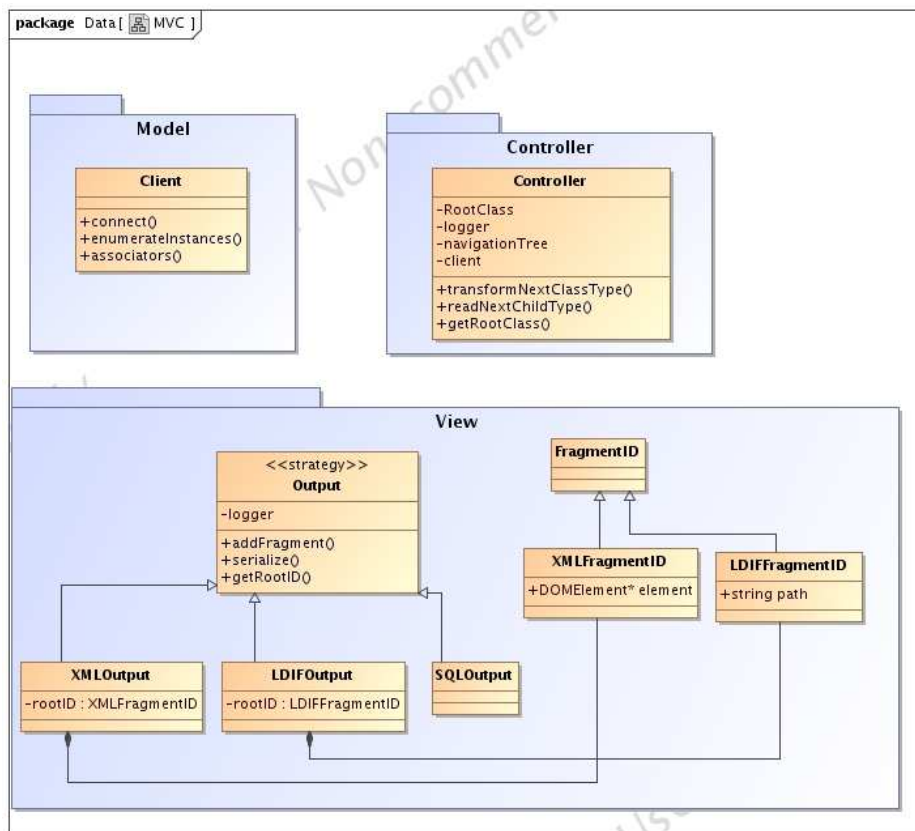


Figure 2 UML Class Diagram for the GLUEMan client

The Model package contains a class called Client that manages the connection with the OpenPegasus server and is responsible for querying the information about GLUE 2 instances and classes. In the Controller package there is the Controller class that is responsible for the navigation

through GLUE instances organized as a tree. It is also responsible for the building of a data structure which contains all the information about instances and their associations. The View package contains the class Output that has three sub-classes, one for each supported serialization of the GLUE information (initially, XML Schema, LDAP, SQL).

The client is meant to be invoked from the command line and will not be in the form of an API library for specific programming language. Programs which want to use it, have to invoke the client via a system call and, depending on the command line arguments, it should require the output to be written to a file or to standard output.

3.3 Relationship to the OGSA-BES Specification

The OGSA-BES specification describes a Web Service interface to which clients can send requests to initiate, monitor, and manage computational activities [BES]. The OGF GLUE 2.0 Information model contains a model for a computing service which can be adopted by BES implementers in order to advertise the capabilities of the BES-managed computing resource to potential clients.

In this section, we describe how GLUE-based information can be exposed via an OGSA-BES interface. The OGSA-BES specification defines the operation GetFactoryAttributesDocument to request a document containing the BES-Management attributes. There is no input parameter, while the output parameter is an XML document compliant to the XML Schema definition of the complex type bes-factory: BESResourceAttributeDocumentType (see fragment as follows):

```
<xsd:complexType name="BasicResourceAttributesDocumentType">
  <xsd:sequence>
    <xsd:element name="ResourceName" type="xsd:string" minOccurs="0"/>
    <xsd:element name="OperatingSystem" type="jsdl:OperatingSystem_Type" minOccurs="0"/>
    <xsd:element name="CPUArchitecture" type="jsdl:CPUArchitecture_Type" minOccurs="0"/>
    <xsd:element name="CPUCount" type="xsd:double" minOccurs="0"/>
    <xsd:element name="CPUSpeed" type="xsd:double" minOccurs="0"/>
    <xsd:element name="PhysicalMemory" type="xsd:double" minOccurs="0"/>
    <xsd:element name="VirtualMemory" type="xsd:double" minOccurs="0"/>
    <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

The integration of the GLUE-based information into OGSA-BES will be performed by extending the XML document returned by the operation GetFactoryAttributesDocument relying on the following XML Schema “hook”:

```
<xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
```

An XML schema mapping of GLUE 2 will be used in order to create an XML document that will be inserted in the BasicResourceAttributesDocument using the GLUE namespace. This approach was discussed with implementers of both gLite CREAM-BES and UNICORE-BES and there is common agreement on it.

In this sequence diagram, we can appreciate a possible interaction between the involved components:

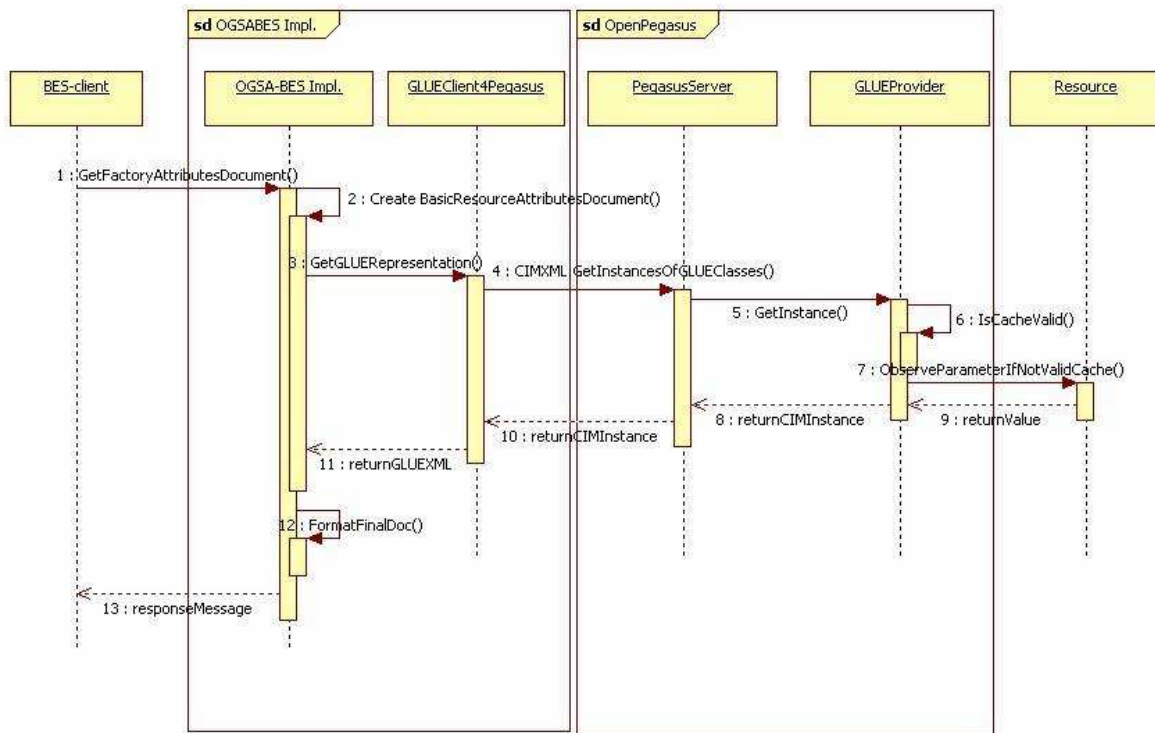


Figure 3 Integration of BES, and GLUEMan

4 Conclusion

In this document, we have described the design of GLUEMan, an implementation of a generic information providers framework for the OGF GLUE information model which leverages the WBEM standards. The goal is to support the middleware developers in the activity of writing providers for the GLUE specification by releasing them for the common problems such as the need for having to deal with multiple rendering of the output, the integration with other providers and the caching capabilities. GLUEMan is also an initial step towards the addition of a management layer to the Grid middleware.

A References

- [OMIIEU] OMII-Europe Project, <http://omii-europe.org>
- [JRA2] JRA2 Activity Wiki, <http://omii-europe.forge.cnaf.infn.it/jra2>
- [GLUE2WWW] OGF GLUE Working Group, <http://forge.ogf.org/sf/sfmain/do/viewProject/projects.glue-wg>
- [GLUE2]GLUE 2.0 Specification (latest draft) <http://forge.ogf.org/sf/go/doc14639>
- [DJRA2.0] D:JRA2.0 Report on Grid Activities relevant to the identification of new services. Deliverable. OMII-Europe project. 31 Oct 2006. <http://omii-europe.forge.cnaf.infn.it/media/jra2/documentation/djra20.pdf>
- [DJRA2.1] D:JRA2.1 Report on Identification of new services activity. Deliverable. OMII-Europe project. 5 June 2007. <http://omii-europe.forge.cnaf.infn.it/media/jra2/djra2.1-revision-2.doc>
- [PEG] Open Pegasus. <http://www.openpegasus.org>
- [CIM] CIPLE Library <http://cimple.org>
- [INI] The Unofficial specification of the INI format. <http://www.cloanto.com/specs/ini.html>
- [GLUEMan] GLUEMan website. <http://glueman.sf.net>