EGEE

Design of the EGEE Middleware Grid Services

RELEASE 2

EU Deliverable DJRA1.5

Document identifier:	**EGEE-DJRA1.5-606574-v1.0**
Date:	**August 4, 2005**
Activity:	**JRA1: Middleware Engineering and Integration**
Lead Partner:	**CERN**
Document status:	**FINAL**
Document link:	**https://edms.cern.ch/document/606574/**

Abstract: This document describes the design of the external interfaces the EGEE middleware (called "*gLite*") exposes. It complements DJRA1.4 - the EGEE Middleware Architecture [4] (https://edms.cern.ch/document/594698/).

Delivery Slip

	Name	Partner	Date	Signature
From	EGEE Design Team	CERN, CCLRC/RAL, INFN DATAMAT, CESNET, NESC, NIKHEF, KTH/PDC, Univ. of Chicago Univ. of Wisconsin-Madison	05/07/2005	
Reviewed by	Gerben Venekamp Catalin Cirstoiu Dave Kelsey Birger Koblitz Massimo Lamanna Cristina Vistoly	NIKHEF CERN RAL CERN CERN INFN	25/07/2005	
Approved by	PEB		04/08/2005	

Document Change Log

Issue	Date	Comment	Author
0.2	29/07/2005	Implemented changes requested from reviewers	JRA1 Design Team
1.0	04/08/2005	Update document status for delivery to EU	Erwin Laure

Document Change Record

Issue	Item	Reason for Change

CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

This document describes the design of the external interfaces the EGEE middleware (called "*gLite*") exposes. In describing those interfaces, we assume the reader is familiar with the EGEE middleware architecture document revision [4] (https://edms.cern.ch/document/594698/). It is a revision of the DJRA1.2 [5] (https://edms.cern.ch/document/487871/).

The scope of this design is the second release of the gLite middleware due at PM21 (December 2005) which will be achieved incrementally according to the work plan defined in https://edms.cern.ch/document/573493/ and part of DJRA1.4.

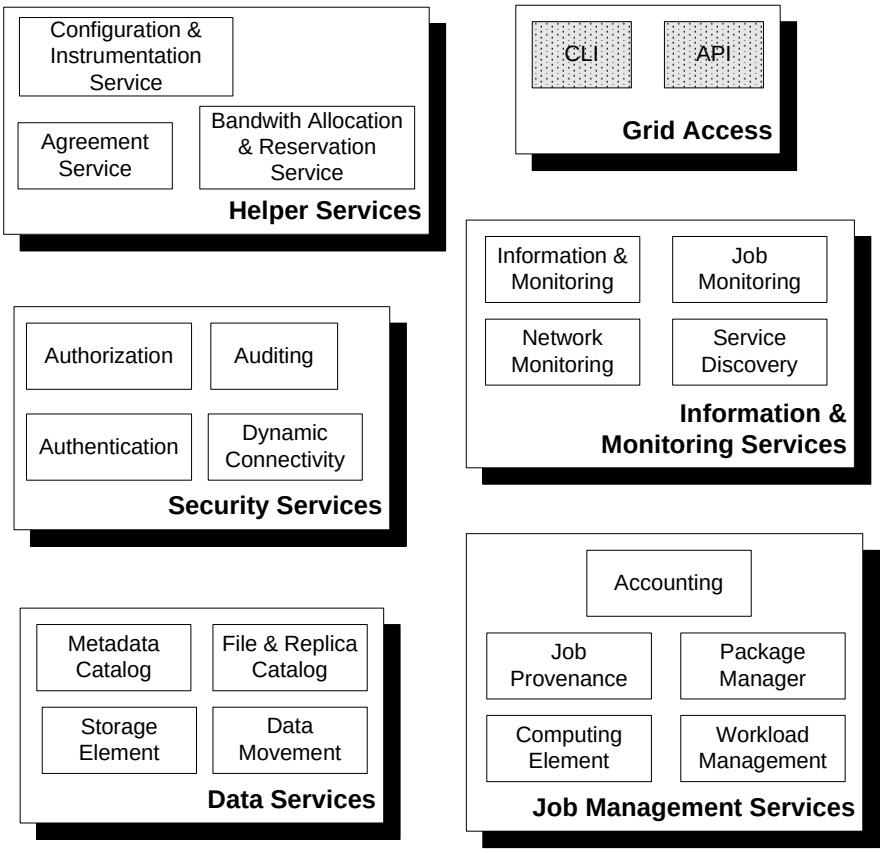

Figure 1: gLite Services

According to the architecture document, the gLite services are grouped into 5 logical service groups plus associated APIs and CLIs: *Security Services*, *Information & Monitoring Services*, *Data Services*, *Job Management Services*, and *Helper Services*. Figure 1 depicts the service groups and high level services each group contains.

Contrary to the first version of this document we don't elaborate all the service interfaces directly in this document but rather give links to the individual service interfaces (either generated from the WSDL directly or an appropriate high-level language binding, mostly Java) which describe in detail

- the semantics of the service,

- the interfaces it exposes,
- input and output parameters, and
- error codes or exceptions.

Note that a detailed description of the internal functionality of the services and their dependencies are beyond the scope of this document and are discussed in independent service specifications. These can be found at[1]:

- **Security Services**:
  - Authentication:
    - DJRA3.1 https://edms.cern.ch/document/487004/
  - Authorization:
    - AuthZ framework: https://edms.cern.ch/document/501718/
    - Delegation: https://edms.cern.ch/document/508598/
    - LCAS/LCMAPS: http://www.nikhef.nl/grid/lcaslcmaps/
    - VOMS: https://edms.cern.ch/file/571991/1/voms-guide.pdf
    - VOMS-Admin: https://edms.cern.ch/file/572406/1/user-guide.pdf
  - Auditing:
    - Job Repository: pending
  - Dynamic Connectivity: pending
- **Information and Monitoring Services**:
  - Information and Monitoring:
    - R-GMA: https://edms.cern.ch/document/490223
  - Job Monitoring:
    - log4j: http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/log4j-guide.pdf
  - Service Discovery:
    - SD Guide: http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/service-discovery.pdf
  - Network Monitoring:
    - DJRA4.2: https://edms.cern.ch/file/533215/2/EGEE-DJRA4.2-533215-v1.1.pdf
- **Job Management Services**:
  - Accounting:
    - DGAS guides:
      - HLR: https://edms.cern.ch/file/571271/1/EGEE-DGAS-HLR-Guide.pdf
      - Gianduia: https://edms.cern.ch/file/571271/1/EGEE-DGAS-Gianduia-guide.pdf
      - Price Authority: https://edms.cern.ch/file/571271/1/EGEE-DGAS-PA-Guide.pdf
  - Computing Element:
    - CE guides:
      - CE Monitor: http://grid.pd.infn.it/cemon/field.php

[1] note that for new services these documents are still pending and will appear in the near future

        - CREAM: http://grid.pd.infn.it/cream/field.php
      - Condor guides: http://www.cs.wisc.edu/condor/manual/index.html, Version 6.7.x
      - GT2/GT4 guides: http://www.globus.org/toolkit/docs/2.2/ http://www.globus.org/toolkit/docs/4.0/techpreview/wms/
  - Workload Management:
    - WMS guides:
      - WMS: https://edms.cern.ch/file/572489/1/WMS-guide.pdf
      - JDL: https://edms.cern.ch/document/590869/1
    - LB guides: https://edms.cern.ch/file/571273/1/LB-guide.pdf
  - Job Provenance:
    - JP guide: pending
  - Package Manager: pending

- **Data Services**:
  - Data Services overview:
    - https://edms.cern.ch/file/570643/1/EGEE-TECH-570643-v1.0.pdf
  - Metadata Catalog:
    - MD Guide: https://edms.cern.ch/file/570779/1/EGEE-TECH-570779-java-v1.0.pdf
  - File & Replica Catalog:
    - Fireman Guide: https://edms.cern.ch/file/570780/1/EGEE-TECH-570780-JAVA-API-v1.0.pdf
  - Storage Element:
    - gLite-I/O guide: https://edms.cern.ch/file/570771/1.1/EGEE-TECH-570771-v1.1.pdf
  - Data Movement:
    - FTS/FPS guide: https://edms.cern.ch/file/591792/1/EGEE-TECH-591792-Transfer-Java-v1.0.pdf

- **Helper Services**:
  - Configuration & Instrumentation Service : pending
  - Agreement Service:
    - Advanced reservation guide: https://edms.cern.ch/document/508055/
  - Bandwidth Allocation & Reservation Service:
    - MJRA4.5 https://edms.cern.ch/document/593453/1

The interested reader may also refer to the user and installation documentation of the already released gLite services (cf. the planning part of DJRA1.4 [4]) which can be found at:

http://glite.web.cern.ch/glite/documentation/.

In the remainder of this paper we discuss design principles in Section 3. Sections 4-8 present the design of the services. Open issues are discussed in Section 9 and we end this report with some concluding remarks.

1.2 Application Area

This document applies to the implementation of the gLite middleware within the scope of the EGEE project and the JRA1 and JRA3 activity mandate. It is also applicable to other activities within EGEE, in particular SA1, JRA4, NA3, and NA4.

1.3 Main Changes to DJRA1.2

The main changes to the previous version of this document, DJRA1.2 (https://edms.cern.ch/document/487871/) can be summarized as follows:

1. Section 4 – Security Services has been augmented with a discussion of *Sandboxing*, the *Dynamic Connectivity Service* and *Encrypted Storage Service*. These additions resolve the previously reported issues on *Sandboxing* (Section 10.4 of DJRA1.2), *Site Proxy* (Section 10.3 of DJRA1.2, and *Data Protection* (Section 10.5 of DJRA1.2).
2. The Grid Access Service (Section 5 of DJRA1.2) is not planned for anymore.
3. Section 5 – Information and Monitoring Services has been augmented with a discussion of *Service Discovery* and *Network Performance Monitoring* resolving parts of the previously reported *Network Element* issue (Section 10.6 of DJRA1.2).
4. The Package Manager (Section 8 of DJRA1.2) is not planned for anymore.
5. A new Section on *Helper Services* (Section 8) has been added containing *Bandwidth Allocation and Reservation* (resolving parts of the *Network Element* issue of DJRA1.2, Section 10.6), *Agreement Service*, and *Configuration and Instrumentation*.

Apart from these changes, only minor adjustments to some interfaces defined in DJRA1.2 have been applied.

1.4 Document Amendment Procedure

This document can be amended by the EGEE JRA1 design team. The document shall be maintained using the tools provided by the CERN EDMS system.

The API specifications contained in this document are subject to changes based on user feedback and implementation or deployment requirements.

1.5 GLOSSARY

AA	Attribute Authority
AC	Attribute Certificate
ACL	Access Control List
AFS	Andrew File System
API	Application Programming Interface
AuthN	Authentication
AuthZ	Authorization
BT	Bulk Transfer
CA	Certification Authority
CAS	Community Authorization Service
CE	Computing Element
CEA	Computing Element Acceptance
CLI	Command Line Interface
DGAS	DataGrid Accounting System
DNS	Domain Name System
DRMAA	Distributed Resource Management Architecture API
EDG	European DataGrid
EGEE	Enabling Grids for E-Science in Europe
FC	File Catalog
FPS	File Placement Service
FS	File System
FTP	File Transfer Protocol
FTS	File Transfer Service
GAS	Grid Access Service
GGF	Global Grid Forum
GN2	The GEANT2 Project
GOC	Grid Operations Centre
GRAM	Grid Resource Access Manager
GSI	Grid Security Infrastructure
GUID	Global Unique Identifier
GSM	Grid Storage Management
GUI	Graphical User Interface
HEP	High Energy Physics
HLM	Higher Level Middleware
HTTP	Hypertext Transfer Protocol
ISM	Information Super Market
I/O	Input / Output
JC	Job Controller
JDL	Job Description Language
JP	Job Provenance Service
L&B	Logging and Bookkeeping Service
LCAS	Local Centre Authorization Service
LCG	LHC Computing Grid
LCMAPS	Local Credential Mapping Service
LFN	Logical File Name
LHC	Large Hadron Collider
L-NSAP	Local Network Service Access Point
LRMS	Local Resource Management System

NAT	Network Address Translation
NE	Network Element
NOC	Network Operations Centre
NPM	Network Performance Monitoring
NM-WG	GGF's Network Monitoring Working Group
NREN	National Research and Education Network
NSAP	Network Service Access Point
NTFS	(Microsoft) NT File System
OCSP	Online Certificate Status Protocol
OGSA	Open Grid Services Architecture
PAT	Policy Administration Tool
PCI	Policy Communication Interface
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PKI	Public Key Infrastructure
PM	Package Management
POSIX	Portable Operating System Interface (X)
PR	Policy Repository
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RC	Replica Catalog
RDMS	Relational DataBase Management System
RLS	Replica Location Service
SAML	Security Assertion Markup Language
SE	Storage Element
SIPS	Site Integrated Proxy Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SRM	Storage Resource Manager
SURL	Site URL
TQ	Task Queue
UC	User Context
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
VDT	Virtual Data Toolkit
VLL	Virtual Leased Line
VO	Virtual Organisation
VOMS	Virtual Organisation Membership Service
WM	Workload Manager
WMS	Workload Management System
WS	Web Services
WS-A	Web Services Addressing
WS-E	Web Services Eventing
WS-I	Web Services Interoperability
WS-N	Web Services Notification
WSRF	Web Services Resource Framework
XACML	eXtensible Access Control Markup Language

2 EXECUTIVE SUMMARY

This document presents the interfaces of each of the gLite (EGEE middleware) services described in the Architecture deliverable DJRA1.4 [4]. It is a revision of DJRA1.2 [5] and shows how each service may be used but does not explain how they work internally, which is documented in separate documentation listed in Section 1. Instead of providing all the service interfaces directly in this document, links to the appropriate interface documentation are given.

Chapter 3 explains and justifies the guiding design principles of the gLite middleware:

- Service Oriented Architecture
- Use of Web Services
- Web Service Interoperability WS-I
- Service Interoperability
- Service Autonomy

Chapter 4 sets out the security services: Authentication, Authorization, Auditing, and Dynamic Connectivity. These enable the identification of entities (users, systems, and services), allow or deny access to services and resources, provide information for post-mortem analysis of security related events, and allow to dynamically connect to services outside of firewalls.

Chapter 5 describes the information and monitoring services: the basic Producer and Consumer services or R-GMA, an API for use by job monitoring following the apache logging services, an API for service discovery, and a system for network performance monitoring.

Chapter 6 lists the job management services. The workload management system accepts and runs “jobs” for users and requests for interactive work. It is responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed. An accounting service accumulates information about the usage of Grid resources; a logging and bookkeeping service tracks jobs in terms of events gathered from various components; and a job provenance service archives the definition of submitted jobs, their execution conditions and environment, and important points of the job life cycle. The package manager mentioned in the architecture document is not contained in this document anymore since it is in very early stage and will not materialize for release 2 of gLite.

Chapter 7 describes the set of data services. These services allow files to be written, registered, read, updated and replicated. The services in particular include the Storage Element, Catalogs, and Data Movement services.

Chapter 8 lists a few helper services, in particular services for bandwidth allocation and reservation, an agreement service that can be used for advanced reservation, and a service configuration and instrumentation service.

Chapter 9 lists some outstanding issues.

Chapter 10 holds the conclusions.

The following Table 1 gives a summary of all the interfaces defined in this document:

Service	**Interface**
Security	
Proxyrenewal	http://egee.cesnet.cz/en/WSDL/renewal_8h.html
Authorization Framework	http://cern.ch/EGEE-JRA3/javadoc/org.glite.security.authz-framework-java/html/
LCAS	http://www.nikhef.nl/grid/lcaslcmaps/lcas_apidoc/html/
Delegation	http://cern.ch/EGEE-JRA3/javadoc/org.glite.security.delegation-interface/html/ http://cern.ch/EGEE-JRA3/javadoc/org.glite.security.delegation-java/html/
VOMS	http://cern.ch/EGEE-JRA3/javadoc/org.glite.security.util-java/html/ http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/voms/VOMS_C_API/html/ http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/voms/VOMS_CC_API/html/
VOMS Admin	http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.voms-admin-interface/html/
LCMAPS	http://www.nikhef.nl/grid/lcaslcmaps/lcmaps_apidoc/html/
Information & Monitoring	
R-GMA	http://hepunx.rl.ac.uk/jra1-uk/glite-r1/java-api
Job Monitoring	Using the Apache logging
Service Discovery	http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/java-service-discovery/
NPM Mediator	http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.middleware-mediator-interface/interface/
NPM Discoverer	http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.middleware-discoverer-interface/interface/
NPM Framework	http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.framework-interface/interface/
Job Management Services	
DGAS	http://www.to.infn.it/grid/accounting/dgas/wsdl
CE	Condor-C and GT2 http://grid.pd.infn.it/cream/wsdl
CE Monitor	http://grid.pd.infn.it/cemon/wsdl
WM Proxy	http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy
L&B	http://egee.cesnet.cz/en/WSDL/
JP Index and Storage	http://egee.cesnet.cz/en/WSDL/
Data Services	
gLite-I/O	http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-io-client/html/files.html
FiReMan	http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-catalog-interface/html/index.html
Metadata	http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-catalog-interface/html/index.html
FTS/FPS	http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-transfer-interface/html/index.html
Helper Services	
BAR	http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.bar.service-interface/interface/
Agreement	http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/allocation/wsdl/
Configuration & Instrumentation	http://cern.ch/egee-jra1-integration/ConfigurationService/Documentation/GliteServicesInterfaces.pdf

Table 1: Summary of the gLite Interfaces

3 DESIGN PRINCIPLES

In designing the gLite services we try to follow some key design principles that are also discussed in the EGEE Architecture Document [4]:

Service Oriented Architecture (SOA) The EGEE Architecture is defined as a Service Oriented Architecture. As described in detail in Section 4 of the Architecture Document, the key concept of SOA is a loose coupling between interacting services, which communicate with each other through well-defined interfaces and protocols.

The concept of loose coupling needs to be observed in the service design. This in particular means that all services need to have a well-defined interface and may be accessed, installed and upgraded independently.

Use of Web Services There is a broad consensus in the Grid community that Web Services (see for instance http://www.w3.org/2002/ws/) provide a useful and widely supported environment for developing Grid services. This is also reflected by the Global Grid Forum (GGF) that proposes to base Grid Services on a common, Web Services based infrastructure, which is currently being defined in the context of the Web Service Resource Framework (WSRF) [11].

We therefore base the design of most of the gLite Grid Services on Web Services and describe their interfaces by means of the Web Service Definition Language.

Web Service Interoperability WS-I The Web Services Interoperability Organisation is an open industry effort chartered to promote Web Services interoperability across platforms, applications, and programming languages [15]. The first basic WS-I document was released in April 2004, and contains simple guidelines on how to use WSDL and SOAP to be interoperable within the Web Services domain.

The service specifications developed in this document are required to conform to WS-I.

Note that as discussed in Section 11.3 of the Architecture Document, compliance with upcoming standards, like WSRF, is not an immediate goal for gLite.

Service Interoperability Grid services need to be interoperable in such a way that a client may talk to different independent implementations of the same service. Following a strict SOA approach with well-defined interfaces specified in WSDL conforming to WS-I will help achieve this goal. Apart from this low-level interoperability, higher-level services interoperability will be facilitated through international standardization efforts and best practices. A good example for this is the Storage Resource Manager (SRM) [13]. We plan to achieve this goal through active participation in international standardization efforts, like the Global Grid Forum (GGF).

In addition, the Grid services should co-exist with, and leverage existing Grid infrastructures like LCG `http://cern.ch/lcg`, OSG `http://www.opensciencegrid.org//`, or NDGS `http://www.nordugrid.org`. This can be achieved by developing lightweight services that only require minimal support from their deployment environment and are based on widely accepted (de-facto) standards.

Service Autonomy Although the services defined in this document are supposed to work together in a concerted way as discussed in the Architecture Document, they should be usable also in a stand-alone manner in order to be exploitable in different contexts. Ideally, if a user only requires a subset of services

to achieve his task, he should not be forced to use additional services; in reality, this goal might not always be achievable and at least stubs or dummy services might be needed, however, the service design and implementation should proceed in that direction. The inverse of this argument needs to be supported as well: in certain circumstances there might be the need to include additional services into the Grid system. While a service oriented architecture in general foresees this dynamic extension, the service implementations also need to be compliant with this goal, by using dynamic discovery mechanisms, for instance.

4 SECURITY SERVICES

Security services encompass the Authentication, Authorization, Delegation, and Auditing services which enable the identification of entities (users, systems, and services), allow or deny access to services and resources, and provide information for post-mortem analysis of security related events.

Many of the security services in the gLite architecture are implemented at the Transport layer, or are concerned with obtaining credentials in ways that are outside the scope of the Architecture. For each of the security components, a brief overview is presented here:

authentication is concerned with identifying users. This is done using many different mechanisms, to which the interface as such is not part of the gLite design.

authorization is concerned with allowing or denying access to services based on policies. The core problem with authorization in a Grid setting is how to handle the overlay of policies from multiple administrative domains (user policy, VO specific policy, operational procedures, site-local policy), and how to combine them.

The interface to the acquisition of authorization tokens using VOMS, and the interface to manipulate the VOMS repository is presented in Section 4.2.2.

delegation is concerned with giving some subset of an entity's privileges to another (dynamically created) entity on relatively short notice, and only for a short amount of time. Since it is difficult to predict the need for delegation beforehand, the delegation interface provides a means to delegate credentials at an arbitrary point in time to a specified service.

sandboxing is concerned with containment of a user's actions (such as running a job) in order to minimise the impact on the local system. Generic creation of such sandboxes, based on an entity's Grid credentials such as VO membership, relieves the local system administrator from the burden of administering individual users.

auditing is a very general term: here, we primarily mean the system security aspects of auditing, such as monitoring and providing information for post-mortem analysis of security related events.

data protection is concerned with providing additional security protection mechanisms of long-term storage of (sensitive) data.

Details regarding the security architecture are given in the Global Security Architecture document [6]. The security requirements can be found in the Activity User Requirements document [7].

4.1 AUTHENTICATION

The authentication model for EGEE is based on the concept of "trusted third parties" (TTPs): entities that are not related to either the user, the virtual organization, or the resource provider (called "relying parties" in general), except through a trust relationship. Underlying that trust relationship is the digital signature of the TTP, based on conventional asymmetric cryptographic techniques (like RSA or DSA). The TTP will bind the cryptographic keypair to one or more identifiers that represent the entity. Although other models are possible, as explained in the gLite Architecture [4], it is this model that is the one best suited to grids today.

Although theoretically a single TTP could service the entire community, in practice a mesh of TTPs exists. One can thus define a set of resources, users and services that agree to use a common set of TTPs

for authentication. Such a common authentication domain of course does not imply common rights of access, and as such this domain does not yet constitute an "Infrastructure".

The services will, when crossing an administrative or domain boundary, use mutually authenticated (and possibly mutually authorized) communication mechanisms. The authentication will be based on the TTP model outlines above, with both entities in the communication authenticating each other. The current design will be using transport-layer authentication based on the TLS protocol, with suitable enhancements to implement the grid single sign-on: support for "proxy" credentials in the form of RFC3820. The services or application servers must accept and forward the information received to the authorization systems and the service.

The authentication information is provided through mechanisms at the transport layer. This information must be available not only to the services but also to the service hosting environments. Access to this information is provided through the GSS-API (RFC2853) for native applications, and through JSSE for Java applications.

The proxy renewal service was designed to support long-running jobs. The service utilizes the MyProxy [14] server where the user stored her credentials and by periodically contacting the MyProxy repository it ensures that the proxy is kept valid for the whole lifetime of the job. The service is also able to renew VOMS attributes (cf. section 4.2.2) if they are present in the proxy certificate. The interface description is located at

http://egee.cesnet.cz/en/WSDL/renewal_8h.html.

Details about the authentication architecture and design can be found in the Global Security Architecture document DJRA3.1 [6].

4.2 AUTHORIZATION

The authorization mechanism is in charge of deciding whether to grant access to a resource or not.

The authorization mechanism collects information from several sources. One source of information is the authentication system which provides the user certificate chain and possibly a VOMS Attribute Certificate that includes the VO group and role information. Another source of information is the site policy which defines site specific access restrictions. Other information sources include application-specific access control lists at file level and the VO policy.

All the information collected is then used to make the authorization decision, which is either yes or no.

The authorization mechanism is integrated into each service that uses it and does not expose any web service interfaces. Java services can use the EGEE java authorization framework and C/C++ services can use LCAS for coarse grained site access control. The description of the interfaces can be found here:

- java authorization framework: http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.authz-framework-java/html/.
- LCAS: http://www.nikhef.nl/grid/lcaslcmaps/lcas_apidoc/html/.

4.2.1 DELEGATION

This section describes the interface to (externally) delegate a credential (a full or limited user proxy as well as any composite credential containing authorization assertions) to a service. This delegation interface supersedes the delegation extensions added to TLS in the HTTPG protocol and is a Web Service version of the G-HTTPS[1] protocol.

The java interface documentation can be found here:

- Client interface: http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.delegation-interface/html/.
- The interface to the delegation library: http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.delegation-java/html/.

4.2.2 Virtual Organization Membership Server

The Virtual Organization Membership Service (VOMS) is an attribute authority service that issues attributes to users and (for mutual authorization) services. The VOMS server can also be considered as a central (but possibly distributed) user management service.

The VOMS server issues digitally (cryptographically) signed Attribute Certificates (ACs) that prove that the VOMS server assures that the user belongs to the virtual organization, belongs to a group in that VO and what role he has in that group. Also the user might have a special capability within a VO/group/role.

When a service gets an AC, and trusts the VOMS server that issued it, it can use these attributes to do further VO/group/role/capability based authorization steps for final authorization decision.

The (Java binding) of the VOMS APIs is described in: http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.util-java/html/.

A description of the C/C++ bindings can be found at: http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/voms/VOMS_C_API/html/ and http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/voms/VOMS_CC_API/html/.

The VOMS administration service is described in: http://egee-jra3.web.cern.ch/EGEE-JRA3/javadoc/org.glite.security.voms-admin-interface/html/.

4.2.3 Sandboxing

A user's actions on a (remote) worker node should be *sandboxed* in order to minimise the impact on the local system. Automatic and generic creation of sandboxes, based on e.g. VO membership, relieves the local system administrator from the burden of administering individual users. The solution currently available in gLite is based on LCMAPS, which supports the so-called poolaccount mechanism. LCMAPS can be integrated with services using the API described here: http://www.nikhef.nl/grid/lcaslcmaps/lcmaps_apidoc/html/. The GT4 workspace service [17] can provide a management interface to LCMAPS. This is described in further detail in [8].

While more secure and refined solutions are being worked upon, specific techniques or implementations (such as use of virtual machines) are planned for future releases of gLite.

4.3 Auditing

The auditing process in itself does not require a separate service. Rather, it imposes a set of common principles and practices on all system components, e.g. to log correct, complete and relevant information and restrict the access to the allowed set of people and services. The audit information should itself be traceable and verifiable throughout its lifetime and throughout service invocations, so that any anomalies in the audit trail can be traced to identifiable service components. Creating a "real" audit service would clearly have benefits in terms of enforcing uniformity of information, providing secure remote access to the information, easing the process of merging and combining the information from different services.

Although no specific architecture for such a component in currently pursued, we explicitly acknowledge the possibility to include such a service in the future, and will ascertain that it can accommodate existing or emergent services.

4.4 Dynamic Connectivity Service

In a generic deployment model no connectivity can be assumed between the execution host(s) and the Internet. This is valid both for inbound connectivity, which may be hampered by both firewalls and non-routable IP address space, as well as for outbound connectivity. In the latter case, although the possible use of non-routable address space may have been alleviated using NAT, this is neither a robust solution nor always desirable given site policies.

A dynamic connectivity service (DCS), either implemented as a separate forwarding service on an assigned machine, or implemented inside existing functions like routers and firewalls, should implement policy-based connectivity provisioning. The implementation thereof could be similar to on-demand bandwidth provisioning services[3].

Since the prototype connectivity provisioning service is still under construction, the inclusion of the DCS design is planned for inclusion only in a future release of gLite.

4.5 Encrypted Storage Service

Many real applications, such as those in the biomedical area, are concerned with additional data protection mechanisms in addition to normal access control. In particular, plaintext (non-encrypted) storage of data (e.g. disks) could expose sensitive data to people having physical access to a computer centre handling those data. This is further explained in [6], where the *Encrypted Storage Service* (formerly known as “data key management service”) is introduced to solve the problem. The service will involve the following three components:

- *data storage system* to store the encrypted file and create/return the GUID,
- *metadata catalog system* to store the key and store/retrieve based on the GUID,
- *OpenSSL system* to create the key and encrypt/decrypt the file (file stream).

Storing a file is decomposed in four steps: (1) create the key, (2) encrypt the file, (3) store the file to data storage system and (4) store the key. Retrieving a file is decomposed in three steps: (1) retrieve the file, (2) obtain the key and (3) decrypt the file. Documentation of the interfaces will appear with first release of this service.

5 Information and Monitoring Services

5.1 Basic Information and Monitoring Services

R-GMA is a relational implementation of the Grid Monitoring Architecture from the GGF. It is as an information and monitoring system for use both by the middleware and by applications.

We describe here the user interfaces to R-GMA. They correspond to the functionality described in section 7.1 of the Architecture deliverable DJRA1.4 [4]. The system and administration interfaces are not currently shown here.

The API (Java binding) is defined at:

http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/java-api.

A good place to start reading is the `PrimaryProducer` which is able to publish information. You can call `createPrimaryProducer` which communicates with the web service to create a new resource and stores a reference to it inside a new instance of a PrimaryProducer which is returned as the value of the createPrimaryProducer operation. One can then call `declareTable` to declare a table and then publish data by a call to `insert` which has an SQL INSERT statement as a parameter.

To retrieve data a `Consumer` is created by a call to `createConsumer` with arguments including the specified `queryType` and `selectStatement`. After a call to `start` one can then call `pop` to retrieve available tuples. For more explanation of how to use the API please see the user guide at: http://edms.cern.ch/document/503617/

5.2 Job Monitoring

For the Job Monitoring, the user will simply use the APIs offered by the apache logging services. The java binding may be found at:

http://logging.apache.org/log4j/docs/api/index.html.

This corresponds to the functionality described in section 7.2 of the Architecture deliverable DJRA1.4 [4].

An R-GMA appender has been written to publish the log messages via R-GMA. This requires some configuration as explained in http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/log4j-guide.pdf.

5.3 Service Discovery

Service Discovery exists for end users and other services to locate suitable services.

We describe here the APIs to Service Discovery corresponding to the functionality described in section 7.3 of the Architecture deliverable DJRA1.4 [4].

The API allows the specification of the desired properties of the service and then all matching services are returned in random order. This API is implemented as as a wrapper around other information service APIs using a plugin mechanism. Support is currently only available for C and Java but it will be extended to Python and perhaps C++.

The Java binding of the API is described at:

http://hepunx.rl.ac.uk/egee/jra1-uk/glite-r1/java-service-discovery/.

5.4 NETWORK PERFORMANCE MONITORING

In the Network Performance Monitoring (NPM) architecture, clients send requests for network monitoring information to the NPM Mediator. The NPM Mediator contacts the NPM Discoverer in order to locate a measurement framework that can answer the request, and then forwards the request to that framework. The Mediator acts as a single point of contact for clients and adds value by providing caching and aggregation.

Each network monitoring request is submitted using the Request and Report schemas defined by the GGF Network Measurement Working Group (NM-WG). This allows greater interoperability and extensibility than would be possible with an arbitrary schema.

Design of these components and their interfaces is ongoing, but the gLite CVS repository contains current WSDL for the Mediator

http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.middleware-mediator-interface/interface/

and Discoverer

http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.middleware-discoverer-interface/interface/

web services, which make reference to the NM-WG schemas. It also includes WSDL to which Monitoring Framework web services should adhere:

http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.npm.framework-interface/interface/.

The security framework for the webservices is discussed in [4].

6 JOB MANAGEMENT SERVICES

The interfaces for Job Management Services, which have been defined in section 8 of the EGEE middleware architecture document [4], is described in the following subsections. Interfaces for the accounting, computing element, workload management, job provenance and package manager services are presented.

6.1 ACCOUNTING

The service interface of the Home Location Register (HLR) component of the accounting service, responsible for archiving the accounting usage records and providing querying functionality, is described in:

http://www.to.infn.it/grid/accounting/dgas/wsdl.

Whilst the querying functionality is exposed through a web service interface, the usage records gathered by dedicated sensors running on the resources are stored into the HLR through a C++ producer API that is hence not described. An HLR service keeps the accounts for both Grid Users and Grid Resources and stores the records of the transactions between user and resources arising from user job submissions. It is also designed to serve as an economic accounting bank service, that is, it can manage economic transactions of virtual credits in an exchange market where Grid Users are supposed to pay for the amount of computational resources consumed on Grid Resources. More details can be found in section 8.1 of the architecture document [4].

6.2 COMPUTING ELEMENT

As described in section 8.2 of the Architecture deliverable DJRA1.4 [4], the main functionality of the Computing Element (CE) service is job management (job submission, job control, etc.), which can be used by an end-user interacting directly with the Computing Element, or by the WMS, which submits a given job to an appropriate CE found by a matchmaking process.

The current CE based on Condor-C and Globus GT2 GRAM is interfaced by the WMS through Condor "grid" universe with a "condor" grid type as described in http://www.cs.wisc.edu/condor/manual/v6.7/5_4Condor_C.html.

A new design of a CE service implementing a web service interface, CREAM [2], has been proposed, considering also the attempt to standardize the interfaces to Computing Resource Managers (CRM).

CREAM interface is described at: http://grid.pd.infn.it/cream/wsdl.

6.3 COMPUTING ELEMENT MONITOR

This service is responsible to deal with all asynchronous notifications.

As explained in section 8.2 of the architecture document [4], the CEMon service is also responsible, for a CE working in pull mode, to request jobs to the Workload Management Service.

The documentation of the CEMon interface, generated from the WSDL of the service, is available at:

http://grid.pd.infn.it/cemon/wsdl.

6.4 WORKLOAD MANAGEMENT

This section describes the service interface of the Workload Manager (WM) that represents the core component of the Workload Management System described in section 8.3 of the EGEE middleware architecture document [4]. The purpose of the Workload Manager Service is to accept requests for job submission and management coming from its clients and take the appropriate actions to satisfy them. The complexity of the management of applications and resources in the grid is hidden by the WM to the users, whose interaction with the service is limited to the description, via a high-level, user-oriented specification language, the Job Description Language (JDL) [16], of the characteristics and requirements of the request and to the submission of it through the provided interfaces. It is then responsibility of the WM to translate these abstract resource requirements into a set of actual resources, taken from the overall grid resource pool, for which the user has access permission. The JDL allows the description of the following request types supported by the WM service:

- Job: a simple application
- DAG: a direct acyclic graph of dependent jobs
- Collection: a set of independent jobs

Jobs in turn can be batch, interactive, MPI-based, checkpointable, partitionable and parametric. Besides requests submission, the WM interface also exposes additional functionality for request management and control such as cancellation and output retrieval. Requests status follow-up can be instead achieved through the functionality exposed by the Logging & Bookkeeping (LB) service interface described in Section 6.5 of this document.

The documentation of the web service interface of the Workload Management System, a.k.a. WMProxy, generated from the WSDL of the service is available at:

http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wmproxy.

6.5 LOGGING AND BOOKKEEPING

The following describes the Logging and Bookkeeping (L&B) service which is responsible to track jobs. The LB service receives job events (from the various components of the Workload Management System), stores them, and derives then the corresponding job states. The LB service is described in section 8.3.5 of the Architecture document [4].

Job information is fed into LB through a logging interface (legacy C and shell API, described in [10]) and it is not exposed as a web service yet.

The users may query job states or retrieve LB events either via C/C++ interface [10] or via web-service interface, described in detail at:

http://egee.cesnet.cz/en/WSDL/.

Besides querying for the job state actively the user may also register for receiving notifications on particular job state changes. This functionality is supported only in the legacy C/C++ interface.

6.6 JOB PROVENANCE

The Job Provenance (JP) service (Section 8.4 of DJRA1.4 [4]) is composed of the JP Index server and and JP Primary storage which are detailed below.

6.6.1 JOB PROVENANCE INDEX SERVER

The following describes the Job Provenance Index Server, which provides a limited data mining capability on the JP data: each index server is configured by its administrator to support a set of queryable attributes chosen from various system attributes (like job submission, start, and termination time, CE name, exit code, arbitrary user tags, etc.).

WS interface definition and documentation is available at:

http://egee.cesnet.cz/en/WSDL/.

6.6.2 JOB PROVENANCE PRIMARY STORAGE

The Job Provenance (JP) Primary Storage Service is responsible to keep the JP data (definition of submitted jobs, execution conditions and environment, and important points of the job life cycle) in a compact and economic form.

The JP Primary storage provides public interfaces for data storing, retrieval based on basic metadata, and registration of Index servers for incremental feed.

Bulk data file transfers are realised with specialised protocols (like gridftp), the control WS interface definition and documentation is available at:

http://egee.cesnet.cz/en/WSDL/.

7 DATA SERVICES

The data services have been defined in section 9 of the EGEE middleware architecture document [4]. At the end of the same document it is explained that the services built here are based on data stored in files, but that the architecture is flexible enough to allow access to data in databases and metadata catalogs.

7.1 STORAGE ELEMENT

The SE is defined in section 9.2 of [4]; see especially Figure 12. As explained there, it has the following interfaces:

- The SRM interface [13].
- A native I/O interface (see Section 7.1.1 below).
- A gridftp interface [12].

The SE is tightly bound to the gLite I/O component which gives the end-user access to the logical file namespace, enforcing the access control list. The SE is also closely integrated with the gLite File Transfer Service component.

7.1.1 THE GLITE FILE I/O INTERFACE

The actual file I/O interface is not a webservice. To implement a proper I/O, the library needs to keep the state of the file while it is read/written using a file handle. Such a handle cannot be kept efficiently over a stateless web service. Also, reading binary data over a web service is very inefficient, hence this is the only service in our architecture that does not come with a webservice description but rather a standard C client interface.

The file I/O interface is described in the following document:

http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-io-client/html/files.html

7.2 CATALOGS

The catalog interfaces are described in detail in section 9.3 of the updated Architecture Document [4]; see especially Figure 14.

7.2.1 FIREMAN

The File and Replica catalog interfaces are implemented by the gLite Fireman catalog, to which the user can connect through web services. The user can also use API wrappers provided for most common programming languages (C, C++, Java, Perl and JavaScript). A simplified API is provided for C. A set of command line tools is also provided.

The web-service description of the API (in Javadoc format) can be found at:

http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-catalog-interface/html/index.html

The method names and objects are identical for the other language implementations (examples of use are in the User guide). The simplified C API is also described in the User guide.

7.2.2 METADATA CATALOG

The web-service interface of the metadata catalog can be found at the same location as for the Fireman catalog, see above, again in Javadoc format. The relevant package is called `org.glite.data.catalog.service.meta`.

Since metadata is a very application specific issue we don't provide a common implementation of a metadata catalog with the above interface but our data services will interoperate with any catalog exposing the interface defined above.

7.3 DATA MOVEMENT

The data movement services are described in detail in section 9.4 of the updated Architecture Document [4]; see especially Figure 16.

The interface which has been defined and documented is that of the File Transfer Service (FTS) and File Placement Service (FPS). The web-services description of both can be found at (in Javadoc format):

http://cern.ch/egee-jra1-data/glite-data-stable/stage/share/doc/glite-data-transfer-interface/html/index.html

The FTS and FPS both copy files. The FPS additionally updates the Fireman catalog upon successful copy (to indicate the creation of a new replica). For the FTS (which has no catalog interaction), the user specifies the file source-destination pairs as physical storage file names. For the FPS (which does interact with the replica catalog), the user specifies the file sources as logical files with an associated copy destination.

The interface of the data scheduler has not yet been defined, but it will be generally the same as that of the FPS. The architectural differences between the FPS and the DS are discussed in the Architecture Document.

8 HELPER SERVICES

8.1 BANDWIDTH ALLOCATION AND RESERVATION

The Bandwidth Allocation and Reservation (BAR) architecture consists of BAR clients, BAR services and Network Service Access Points (NSAPs). Clients send application-level requests for bandwidth allocation to a BAR service. The BAR service then translates these into network-level requests and sends them to an NSAP. Local configuration is taken care of by a Local NSAP. The architecture is described in detail in the BAR Architecture Document [9].

The BAR service interface WSDL is available from the gLite CVS repository:

http://jra1mw.cvs.cern.ch/cgi-bin/jra1mw.cgi/org.glite.bar.service-interface/interface/

The NSAP service is not developed as part of EGEE, but the current revision of the interface is available from the EGEE JRA4 CVS repository:

http://egeejra4.cvs.cern.ch/cgi-bin/egeejra4.cgi/NSAP/interface/

The security framework for the webservices is discussed in [4].

8.2 AGREEMENT SERVICE

The Agreement Service is a component supporting the signalling and monitoring of Service Level Agreements. Clients of the Agreement Service can be users, Data Managers, the Workload Management System, etc. The Agreement Service relies on resource-specific reservation and allocation service providers for the enforcement of the agreement guarantees.

The Agreement Service functionality is detailed in Chapter 10.2 of the gLite architecture document [4].

Documentation about the Agreement Service interface, generated from the WSDL of the service, is available at:

http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/allocation/wsdl/.

8.3 CONFIGURATION AND INSTRUMENTATION

The Configuration Service is a site service to store and deploy service configuration information. Clients of the service can be system administrators or other services. A Configuration Management Console is also provided to access the configuration repository via the Configuration Service.

The Instrumentation Service is an interface and a reference implementation that gLite Services must implement or inherit from to allow direct management operations and to interact with the Configuration Service.

The Configuration and Instrumentation Services are described in Section 10.3 of the gLite architecture document [4]. Design documentation of the services is available at:

http://cern.ch/egee-jra1-integration/ConfigurationService/Documentation/GliteServicesInterfaces.pdf

9 ISSUES

9.1 ERROR REPORTING

All gLite services should use consistent means of reporting errors. This in particular requires a common (set of) base fault class(es) all services base their own fault upon. In the current design, most service groups have consistent fault mechanisms and appropriate base faults, however, these are not yet synchronized between service groups. A proposal to base all gLite errors on three base fault classes is currently being evaluated.

9.2 VERSIONING

All gLite services should provide a consistent way of querying their implementation version. An appropriate method is available in the services described in this document, however, these methods are not yet fully synchronized. It would also be possible to define a common port type exposing this functionality, every gLite service would have to implement.

9.3 IMPLEMENTATION CONSIDERATIONS

The Grid system realised by the services described above should allow a maximum of flexibility in service deployment, service composition, and service interoperability.

In order to achieve this, implementations of the services need to take into account the requirements discussed in Section 3 of the Architecture document [4]. The main issues include:

Interoperability Service implementations need to be interoperable in such a way that a client may talk to different independent implementations of the same service. Following a strict SOA approach with well-defined interfaces specified in WSDL will help achieve this goal.

In addition, the Grid services need to be able to co-exist with, and leverage existing Grid infrastructures like LCG `http://cern.ch/lcg`, OSG `http://www.opensciencegrid.org`, or NorduGrid `http://www.nordugird.org`. This can be achieved in developing lightweight services that only require minimal support from their deployment environment.

Service Deployment Grid services need to be easily deployable and configurable across a wide range of platforms. This goes along the lines of the goal of interoperability with existing infrastructures discussed above. In addition, several deployment scenarios need to be supported (e.g. services running together on the same physical machine, services supporting single or multiple VOs, etc.). Section 14 of the Architecture document [4] illustrates a typical deployment scenario.

Re-use of existing services As much as possible, existing, well-tested components should be adopted according to the presented design, rather than implementing everything from scratch. We envisage in particular the re-use of components originating from projects like AliEn, Condor, EDG, Globus, LCG, VDT, and others.

Client libraries In this document we have defined the gLite services by means of WSDL which will allow the creation of client libraries on demand. However, this procedure exposes all the low level details of the involved protocols to the user and thus proper client libraries hiding these details should be

provided where appropriate. In addition, the current WSDL tooling available is tedious and error-prone to use, hence ready made client libraries will ease the usage of the services.

10 CONCLUSIONS

This document presented the revised design of the interfaces exposed by *gLite*, the EGEE Grid middleware. It is a revision of the original design presented in DJRA1.2 [5] and complements DJRA1.4—the EGEE Middleware Architecture [4]. We also discussed some implementation considerations that should be observed when implementing and re-engineering the gLite services.

The design described in this document is subject to a continual evolution, based on experiences from early prototype implementations of the services described, user feedback, and the evolution of application and operational requirements.

REFERENCES

[1] A. Frohner A. McNab, J. Hahkala. Grid use of HTTP(S). GGF draft, February 2004. https://forge.gridforum.org/projects/data-rg/document/draft-ggf-DataTransport-RG-G-HTTPS-11/en/1.

[2] CREAM Home Page. http://grid.pd.infn.it/cream/field.php.

[3] DataTAG. Demonstration of Advance Reservation and Services. EU DataTAG Deliverable D2.5, February 2004. https://edms.cern.ch/document/431913.

[4] EGEE JRA1. EGEE Middleware Architecture—Release 2. https://edms.cern.ch/document/594698/.

[5] EGEE JRA1. EGEE Middleware Design—Release 1. https://edms.cern.ch/document/487871/.

[6] EGEE JRA3. Global Security Architecture. https://edms.cern.ch/document/487004/.

[7] EGEE JRA3. Security requirements. https://edms.cern.ch/document/485295/.

[8] EGEE JRA3. Site Access Control Architecture. https://edms.cern.ch/document/523948/.

[9] EGEE JRA4. Specification of Interfaces for Bandwidth Reservation Service. http://edms.cern.ch/document/501154/1.

[10] A. Křenek et al. L&B Users Guide. https://edms.cern.ch/file/571273/1/LB-guide.pdf.

[11] Karl Cajkowski et. al. The WS-Resource Framework, 2004. http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.

[12] W. Allcock et al. GridFTP Protocol Specification. Global Grid Forum Recommendation GFD.20, March 2003.

[13] The GGF Grid Storage Resource Manager Working Group.

[14] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: Myproxy. In *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, August 2001.

[15] The Web Services Interoperability Organization. WS-I Documents. http://www.ws-i.org/Documents.aspx.

[16] F. Pacini. Job Description Language Attributes Specification. EGEE-JRA1-TEC-590869-JDL-Attributes, 2005. https://edms.cern.ch/document/590869/1.

[17] The GLOBUS team. The WorkSpace Management Service. http://www-unix.mcs.anl.gov/workspace.