

Balázs Kónya (editor), Lund University, ARC
Aleksandr Konstantinov, Oslo University, ARC
Laurence Field, CERN, gLite
Moreno Marzolla, INFN, gLite
Morris Riedel, Julich, UNICORE
Bernd Schuller, Julich, UNICORE

UMD, EGI:
<http://web.eu-egi.eu/>

April xxx, 2009

Survey of potential Universal Middleware Distribution (UMD) components

Status of the document

This document represents an incomplete draft of the survey. The content should be considered as a continuous work in progress. The technical group aims at producing a first version ready for public circulation by the time of the Geneva UMD workshop (23-24 April 2009).

Version 0.8

Last modified: 10:30, 23 April

Scope and purpose of the document

Technical experts of the three middleware consortia (i.e. ARC, gLite, UNICORE) have been jointly preparing a survey of the currently available middleware components. The survey intends to provide a functional decomposition, taxonomy and status evaluation of the components of the middleware stacks that have a potential relevance for UMD. Due to limited resources and time constraints the technical team initially can only focus on the components of the ARC/gLite/Unicore middleware stacks, nevertheless wherever it was possible third party solutions were considered in the survey.

The main purpose of this middleware component survey is to be used as the technical background document during the UMD component identification and selection process. The document itself contains no recommendations of particular components: instead its purpose is to present a technically sound state-of-the-art modular decomposition and inventory of existing and under development middleware solutions. In particular, the document provides information in terms of numerous categories along functionalities in the areas of computing, data, information, accounting and security.

Contents

1	Introduction	5
1.1	Component description template	7
2	Computing functionality	8
2.1	Standards landscape	8
2.2	Overview table: computing functionality	9
2.3	Inventory of components	9
2.3.1	ARC: Grid Manager	9
2.3.2	ARC: A-REX	11
2.3.3	ARC: libarcclient (job management module)	12
2.3.4	ARC: ng* job management CLI	13
2.3.5	ARC: arc* job management CLI	14
2.3.6	ARC: Janitor	15
2.3.7	gLite: WMS	16
2.3.8	gLite: WMS Clients	17
2.3.9	gLite: LB	18
2.3.10	gLite: CREAM Service	19
2.3.11	gLite: CREAM Command Line Tools	20
2.3.12	gLite: LCG-CE	21
2.3.13	UNICORE: TSI	22
2.3.14	UNICORE: XNJS	23
2.3.15	UNICORE: UAS (Job Submission and Management)	24
2.3.16	UNICORE: OGSA-BES Adoption	25
2.3.17	UNICORE: Workflow Engine	26
2.3.18	UNICORE: Service Orchestrator	27
2.3.19	UNICORE: Rich Client	28
2.3.20	UNICORE: Command-line Client	29
2.3.21	UNICORE: HILA	30
2.4	New development	32
3	Data functionality	32
3.1	Standards landscape	32
3.2	Overview table: data functionality	32
3.3	Inventory of components	33
3.3.1	ARC: Classic SE	33
3.3.2	ARC: libarcdata2	34
3.3.3	ARC: ng* data management CLI	35
3.3.4	ARC: arc* data management CLI	36
3.3.5	ARC: Chelonia	37
3.3.6	ARC: Chelonia client	38
3.3.7	gLite: DPM	39
3.3.8	gLite: lcg_utils	40
3.3.9	gLite: htcp	41
3.3.10	gLite: SRM client	42
3.3.11	gLite: DPM client	42
3.3.12	gLite: LFC client	43
3.3.13	gLite: FTS client	44
3.3.14	gLite: DPM/DICOM interface	45
3.3.15	gLite: LCG File Catalogue (LFC)	46
3.3.16	gLite: File Transfer Service (FTS)	47
3.3.17	gLite: Hydra	48
3.3.18	gLite: AMGA	49
3.3.19	gLite: AMGA Client API	50
3.3.20	UNICORE: UAS (Storage Management and File Transfer)	51
3.3.21	other: dCache	52
3.4	New development	53

4	Information functionality	53
4.1	Standards landscape.....	53
4.2	Overview table: information area.....	54
4.3	Inventory of components	54
4.3.1	ARC: Classic Infoserver.....	54
4.3.2	ARC: Classic Infoindex	55
4.3.3	ARC: ALIS	56
4.3.4	ARC: ISIS	57
4.3.5	ARC: libarcclient (infosys modules)	58
4.3.6	ARC: Grid Monitor	59
4.3.7	gLite: BDII.....	60
4.3.8	gLite: lcg-info and lcg-infosites.....	61
4.3.9	gLite: Service Discovery API.....	62
4.3.10	UNICORE: CIS	63
4.3.11	UNICORE: Service Registry	64
4.4	New development	65
5	Security functionality	65
5.1	Standards landscape.....	65
5.1.1	Authentication (AuthN).....	65
5.1.2	Authorization (AuthZ).....	65
5.2	Overview table: Security area.....	66
5.3	Inventory of components	67
5.3.1	ARC: HED security API.....	67
5.3.2	ARC: Shibbridge	68
5.3.3	ARC: arcproxy	69
5.3.4	ARC: Charon Authorization Service	70
5.3.5	ARC: Proxy Store.....	72
5.3.6	ARC: Fruitfly credential service.....	73
5.3.7	gLite: VOMS	74
5.3.8	gLite: VOMS-Admin	75
5.3.9	gLite: voms-proxy-*.....	76
5.3.10	gLite: Authorization Service	76
5.3.11	gLite: Short Lived Credential Service.....	78
5.3.12	gLite: SCAS	79
5.3.13	gLite: LCAS	80
5.3.14	gLite: LCMAPS	81
5.3.15	gLite: gLite-Exec (glExec)	82
5.3.16	gLite: gLite-Delegation Client.....	83
5.3.17	gLite: Shibboleth Short Lived Credential Service Client.....	84
5.3.18	gLite: Gridsite	84
5.3.19	gLite: CGSI_gSoap.....	86
5.3.20	gLite: delegation-Java.....	86
5.3.21	gLite: Trustmanager.....	88
5.3.22	UNICORE: Gateway	89
5.3.23	UNICORE: XUADB.....	90
5.3.24	UNICORE: XACML Entity	91
5.3.25	UNICORE: UVOS	92
5.3.26	other: MyProxy	93
5.4	New development	94
6	Accounting functionality	94
6.1	Standards landscape.....	94
6.2	Overview table: Accounting area.....	95
6.3	Inventory of components	95
6.3.1	ARC: JURA.....	95
6.3.2	gLite: Accounting Service	96
6.3.3	gLite: APEL.....	97

6.3.4	other: SGAS	98
6.4	New development	99
7	Other functionality	99
7.1	Overview table: Other components	99
7.2	Inventory of components	99
7.2.1	ARC: HED	99
8	References	100

1 Introduction

The Universal Middleware Distribution (UMD) is a collection of compiled software components with integrated configuration from a single repository that add a layer of Grid functionality on top of operating systems (similar in concept to a Linux distribution). As such it is composed of multiple production quality software components originating from different providers each with their own separate software development infrastructure. UMD is the proposed approach to provide and manage a high quality middleware stack for the future European grid infrastructure, the EGI¹. The UMD software management process is defined in separate document "EGI: Managing the software process"²

This survey presents a functional decomposition and taxonomy of the existing middleware solutions ARC, gLite and UNICORE. The middleware solutions are decomposed into components which are categorized by their provided functionality. The document classifies the middleware components by the following functionality areas:

- Computing functionality

- Data functionality

- Information functionality

- Accounting functionality

- Security functionality

- and other.

Within each functionality area a brief overview of the current standard landscape was prepared, where it was possible, an initial set of specifications to be followed by UMD components was recommended. This standard landscape overview also tries to identify gaps within a certain functionality area where further standardization work is critically needed. In this particular context, the UMD Tech team also takes the work of the OGF Production Grid Infrastructure (PGI) group into account, but focuses more in-depths on the demands given by EGI.

The middleware components are surveyed and presented within a particular functionality area and were evaluated against a recommended set of standards. A rough estimate of efforts per components needed to converge towards the recommended minimum set of standards is provided.

A middleware component is defined as either being a service, a development library or a client implementing one of the identified grid functionalities. In the context of this document services represent self-contained functionality which can be remotely accessed by Grid users or other services. Libraries are publicly supported development interfaces which can be used in the implementation of services or user level applications and can be thought of as part of a Standard Development Kit (SDK). Clients are binary tools aimed at user to be able to carry out some Grid functionality.

For each functionality area a later version of the document will identify potentials for new development work as well.

¹ EGI Blueprint: <http://www.eu-egi.eu/blueprint.pdf>

² UMD Process document:

http://knowledge.eu-egi.eu/knowledge/index.php/UMD#Process_working_group

The UMD candidate components from the middleware stacks were evaluated against a set of criteria (see component template below).

1.1 Component description template

Name

Description

A short paragraph describing the component

Basic information

Component type: *{service, library, client}*

Developer: *the team providing development/maintenance/support for the module*

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: *{planned, development, pre-production, production}*

Usage: *Infrastructures and/or user communities depending on the component*

Interface maturity: *{development, pre-production, production}*

Support status: *{abandoned, best effort, short term, long term}*

Support team: *person/project/organization providing support for the component*

Development plans:

Distribution/Availability

Middleware stack: *The name and the release of the middleware stack distributing the component*

Standalone usability: *Can the component alone be part of UMD?*

Component dependencies: *components listed in this survey on which this component depends*

External software dependencies: *the major external software dependencies*

Component consumers: *list of other components relying on the component*

Supported platforms: *e.g. 32/64 linux, mac-os x, windows, RHEL4, ubuntu, ...*

Planned to be supported platforms: *e.g. 32/64 linux, mac-os x, windows, RHEL4, ubuntu, ...*

Service component information *(only for a service component)*

Public Interfaces:

Aligned security: *security component or security information transferred with the requests*

Capabilities: *capabilities offered by the service according to the Capability_t type of Glue2*

Library component information *(only for a library component)*

Consumers: *list of other components (services/clients) relying on the library*

Language binding: *programming languages in which the API is available*

Interoperability/Standards

Standard-compliance: *{none, non-UMD, early-UMD, full-UMD}*

Interoperable components:

Additional standards: *implemented and relevant non-UMD recommended standards*

Cost estimates

Maintenance/support effort: *in FTEs*

Standard convergence effort: *in FTEs*

New development effort: *in FTEs*

Total effort: *in FTEs*

Other

Any other kind of information

2 Computing functionality

2.1 Standards landscape

There are currently two major standard specifications which apply to job management services, namely the Basic Execution Service (BES) and the Job Submission Description Language (JSDL) specifications. Besides the core BES and JSDL specifications there exist several extensions and profiles created by different OGF activities.

The JSDL [1] is an XML-based notation for describing the requirements of computational jobs for submission to Grid environments. The aim of JSDL is to provide a notation for describing the structure and requirements of individual jobs. The core JSDL is deliberately narrow scoped.

The BES specification [2] describes a basic Web Service interface for creation, monitoring and control of computational jobs. In the BES terminology, jobs are called activities, and are described using the JSDL notation. While JSDL is used to describe the static structure of an activity, BES specifies a set of operations which can be executed on activities: creation, termination, obtaining the current status of an activity or a set of activities and so on. In general, a BES service acts as a frontend to one or more resources, where a resource is a generic term to denote anything from a supercomputer, to a pool of workstations managed through a batch system such as LSF, PBS or Torque, or individual computers.

Although both the BES and JSDL specifications have been finalized, they are not suitable for production use because of several limitations and shortcomings. Some of the most relevant limitations are the following:

- Lack of a common security profile. Security considerations are outside the scope of the BES specification; the result is that different BES implementations might adopt incompatible security settings, which result in the services being not interoperable.
- Lack of common information model. Both BES and JSDL employ a limited-scope information model. The information model exposed through the BES interface is not satisfactory in production environment. Not all significant features of the contained resources are exposed, and there are no efficient ways to query the information model.
- Insufficient treatment of data staging and application environments.

For these reasons, the BES and JSDL specifications need to be suitably extended and reengineered in order to be usable in production infrastructures. Furthermore, the core specifications should be harmonized with GLUE v2.0 and security models. Currently the OGF Production Grid Infrastructures (PGI) Working Group is working towards defining a production profile based on refinements, extensions or redesign of BES, JSDL and GLUE. The outcome of the PGI WG will likely constitute one of the building blocks for the implementation of interoperable job submission and management components in the context of UMD.

It is recommended that UMD through the middleware consortia increases its involvement in the standardization work of PGI working group in order to define a production-ready common specification for job management and job description within one year. Once the PGI specification is released it is suggested that all computing functionality services included in UMD should comply with the PGI profile for job management. The standard convergence work related to PGI conformance within the computing area is estimated to require substantial development efforts from the middleware teams.

2.2 Overview table: computing functionality

	Component name	Type	Maturity	Standard compliance		
2.3.1	ARC: Grid Manager	Service	Production	none		
2.3.2	ARC: A-REX	Service	Pre-production	Early-umd		
2.3.3	ARC: libarcclient (job management module)	Library	Pre-production	Early-umd		
2.3.4	ARC: ng* job management CLI	Client	production	None		
2.3.5	ARC: arc* job management CLI	Client	Pre-production	Early-umd		
2.3.6	ARC: Janitor	Service	development	none		
2.3.7	gLite: WMS	service	production	None		
2.3.8	gLite: WMS Clients	client	production	None		
2.3.9	gLite: LB	service	production	None		
2.3.10	gLite: CREAM Service	service	production	Non-umd		
2.3.11	gLite: CREAM Command Line Tools	client	production	None		
2.3.12	gLite: LCG-CE	service	production	None		
2.3.13	UNICORE: TSI	service	production	None		
2.3.14	UNICORE: XNJS	library	production	Non-umd		
2.3.15	UNICORE: UAS (Job Submission and Management)	service	production	Non-umd		
2.3.16	UNICORE: OGSA-BES Adoption	service	Pre-production	Non-umd		
2.3.17	UNICORE: Workflow Engine	service	Pre-production	none		
2.3.18	UNICORE: Service Orchestrator	service	Pre-production	None		
2.3.19	UNICORE: Rich Client	client	production	None		
2.3.20	UNICORE: Command-line Client	client	production	Non-umd		
2.3.21	UNICORE: HILA	library	Pre-production	???		

2.3 Inventory of components

2.3.1 ARC: Grid Manager

Name ARC Grid Manager

Description

The ARC Grid Manager is a powerful realization of a pre Web Service computing element. It implements job execution capability over a large variety of computational resources. The computing element is built around the Grid Manager. The main features of the Grid Manager include the transparent, automatic and integrated data staging and caching capability, support for large number of batch systems, session directory management, support for RuntimeEnvironments, logging capability, flexible access control, etc...The pre-web service

computing element exposes the Grid Manager's functionality through a custom GridFTP- and XRSL-based interface.

Basic information

Component type: service
Developer: Oslo University, NorduGrid Collaboration
Documentation:
<http://www.nordugrid.org/documents/GM.pdf>
Code repository:
<http://svn.nordugrid.org/trac/nordugrid/browser/arc0>
Development language: c++, shell scripts, perl
Build environment: GNU autotools
Download: download.nordugrid.org
Licence: Apache License Version 2.0

Status

Maturity: production
Usage: as the core server-side ARC component the GM is deployed at every ARC site installation
Interface maturity: production
Support status: long term
Support team: NorduGrid Collaboration
Development plans: only bugfixes

Distribution/Availability

Middleware stack: ARC production releases (latest version 0.6.5)
Standalone usability: yes
Component dependencies: none
External software dependencies: globus pre-WS libraries, gsoap
Component consumers: ARC: ng*, ARC: arc*, ARC: libarcclient
Supported platforms: major linux flavours
Planned to be supported platforms: none

Service component information *(only for a service component)*

Public Interfaces: custom gridftp and XRSL based interface
Aligned security:
Capabilities:

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional standards: gridftp protocol is used for job submission/management

Cost estimates

Maintenance/support effort: 0.5
Standard convergence effort: 0
New development effort: 0
Total effort: 0.5

Other

As part of the development of a next generation ARC computing element the gridftp-based custom interface layer and the security framework of the Grid Manager are being replaced by standard compliant, WS-based technology solution. The new component, called A-REX, will gradually wrap all the existing functionality of the Grid Manager thus replacing the pre-WS technology computing element. All the new functionality development is already taking place as part of the A-REX. Support for Grid Manager will be offered as long as the service is deployed in production.

2.3.2 ARC: A-REX

Name ARC Resource-coupled EXecution service A-REX

Description

A-REX is the next generation computing element of ARC offering WS-interfaces and advanced security solutions. The powerful computing element implements job execution capability over a large variety of computational resources. A-REX is built around the Grid Manager of the pre-WS ARC computing element. A-REX interprets standard job description (JSDL with extensions), offers OGF-compliant WS interface, features transparent, automatic and integrated data staging and caching capability, support for large number of batch systems, session directory management, logging capability, RuntimeEnvironments. A-REX offers WS-based local information interface serving GLUE2 information. A-REX is capable working together with community approved security frameworks.

Basic information

Component type: service

Developer: Oslo University, NIIF

Documentation:

Code repository: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1>

Development language: c++, shell scripts, perl

Build environment: GNU aototools

Download: download.nordugrid.org

Licence: Apache License Version 2.0

Status

Maturity: pre-production

Usage: as the core server-side ARC component A-REX will be deployed at every ARC site installation

Interface maturity: production, but open for changes as the PGI profile emerges

Support status: short term support by KnowARC project, long term support by NorduGrid collaboration

Support team: Oslo University, NIIF

Development plans: add support for PGI profile

Distribution/Availability

Middleware stack: part of the technology preview releases (0.9.x) and will be included in the next production ARC release (0.8.0) scheduled for May 2009

Standalone usability: no

Component dependencies: ARC: HED

External software dependencies:

Component consumers: ARC: libarcclient, ARC: arc* CLI

Supported platforms: major linuxes, solaris, mac-osx

Planned to be supported platforms: windows

Service component information *(only for a service component)*

Public Interfaces: extended BES, extended JSDL, Glue v2.0 through WSRF-RP

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance: early-UMD

Interoperable components: Unicore client

Additional standards:

Cost estimates

Maintenance/support effort: 1
Standard convergence effort: 1
New development effort: 0.5
Total effort: 2.5

Other

A-REX is gradually replacing the pre-WS ARC computing element by wrapping and preserving the powerful functionality offered by the Grid Manager component into novel WS-based solution enriched by an enhanced security framework. All new development related to server-side ARC computing element takes places in A-REX.

2.3.3 ARC: libarcclient (job management module)

Name libarcclient (job management module)

Description

The next generation general purpose grid library (libarcclient) offers transparent access to grid resources through an intuitive and easy-to-use Multilanguage API. The libarcclient comes loaded with plenty of power features including Grid job management and control functionality, resource selection and brokering as well. It features a pluggable architecture which allows it to implement cross-grid job management. The current implementation comes with plugins for ARC Grid Manager, (pre-WS solution), A-REX (extended BES solution), Unicore and CREAM. Libarcclient is targeted for grid developers and advanced application integrators and provides a general toolkit for high-level cross-middleware, multiplatform client side development. The library is the core of the highly successful and powerful, nevertheless lightweight ARC clients, grid portals and client-side job managers.

Basic information

Component type: library
Developer: Uppsala University, Copenhagen University, NIIF of NorduGrid Collaboration
Documentation:
http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/client/client_technical.pdf
Code repository:
<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/acc> ,
<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/libs/client>
Development language: C++ (language bindings are offered for python, Java see below)
Build environment: GNU autotools
Download: download.nordugrid.org
Licence: Apache License Version 2.0

Status

Maturity: pre-production
Usage: as the underlying library of most of the ARC client-side tools it'll be used by every ARC user community
Interface maturity: pre-production
Support status: short term support by KnowARC project, long term support by NorduGrid collaboration
Support team: Uppsala University
Development plans: enhancing maturity of implementation, add support for the emerging PGI-profile

Distribution/Availability

Middleware stack: technology preview release of ARC (version 0.9.3)
Standalone usability: no
Component dependencies: ARC HED

External software dependencies: openssl, glibmm, libxml2
Component consumers: ARC: arc* CLI, ARC:LAP
Supported platforms: all flavours of Linux, MacOS, Solaris
Planned to be supported platforms: Windows

Library component information *(only for a library component)*

Consumers: ARC client tools
Language binding: C++, Python, Java (not tested)

Interoperability/Standards

Standard-compliance: early-UMD
Interoperable components: the library is able to handle CREAM and UNICORE computing elements
Additional standards: already supports many flavors of JSDL and BES with extensions and uses GLUE v2.0 as internal data representation

Cost estimates

Maintenance/support effort: 0.5
Standard convergence effort: 0.5
New development effort: 0.25
Total effort: 1.25

Other

The libarcclient development library is scheduled to be part of the next production ARC release scheduled for 3rd quarter of this year as a replacement of the aging previous generation ARCLIB.

2.3.4 ARC: ng* job management CLI

Name ARC ng* job management CLI

Description

The ARC ng* job management tools (ngsub, ngget, ngcat, ngstat, ngclean, ngkill, ngresub, ngrenew, ngresume, ngsync) are command line utilities for managing Grid jobs on computing elements equipped with the pre-WS Grid Manager ARC component. Their functionality includes resource selection according to criteria specified in job description, job submission, monitoring, re-submission and result retrieval. The ng* CLI provides a light-weight but powerful grid client including an integrated broker.

Basic information

Component type: client
Developer: Uppsala University of NorduGrid collaboration
Documentation: <http://www.nordugrid.org/documents/ui.pdf>
Code repository:
<http://svn.nordugrid.org/trac/nordugrid/browser/arc0/trunk/cli>
Development language: C++
Build environment: GNU autotools
Download: <http://download.nordugrid.org>
Licence: Apache License Version 2.0

Status

Maturity: production
Usage: one of the most often used tool of the broad NorduGrid community
Interface maturity: production
Support status: long term support

Support Team: Uppsala University from NorudGrid Collaboration
Development plans: none (see other comments)

Distribution/Availability

Middleware stack: ARC production releases (latest version 0.6.5)
Standalone usability: Yes
Component dependencies: ARC: Grid Manager
External software dependencies: pre-ws globus packages (globus-ftp-client, globus-rsl, globus-rsl-client, LFC-client-libs), gsoap, openldap
Component consumers: none
Supported platforms: all the major Linuxes
Planned to be supported platforms: none

Interoperability/Standards

Standard-compliance: none,
Interoperable components: none
Additional standards: makes use of gridftp-based custom interface to submit/manages jobs

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0
Total effort: 0.25

Other

The ng* CLI is going to be complemented by the arc* CLI. The transition is scheduled for the 3rd quarter of this year.

2.3.5 ARC: arc* job management CLI

Name arc* job management client tools

Description

The arc* job management client tools implement basic Grid job management functionality – selection of resource aka brokering, submission of job, monitoring job status, retrieval of results – in cross-grid environment. They are based on the libarcclient library. The new client offers the same functionality as the old ng* client with significantly improved performance. It also offers new features such as job migration and reliable job resubmission.

Basic information

Component type: client
Developer: Uppsala University, Copenhagen University, NIIF from NorduGrid Collaboration
Documentation: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/client/arcclient.pdf>
Code repository:
<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/clients/arclib>
Development language: C++
Build environment: GNU autotools
Download: <http://download.nordugrid.org/>
Licence: Apache License Version 2.0

Status

Maturity: pre-production
Usage:
Interface maturity: pre-production
Support status: short term support by KnowARC project, long term support by NorduGrid

collaboration
Support team: Uppsala University
Development plans: enhancing maturity of implementation

Distribution/Availability

Middleware stack: ARC Version 0.9.x
Middleware stack: technology preview releases of ARC (0.9.x)
Standalone usability: no
Component dependencies: libarcclient, ARC:HED
External software dependencies:
Component consumers: none
Supported platforms: all the major Linuxes, MacOS, Solaris
Planned to be supported platforms: windows

Interoperability/Standards

Standard-compliance: early-UMD
Interoperable components: can manage jobs on CREAM and Unicore computing elements
Additional Standards: supports JSDL, BES and uses GLUE 2.0 as internal data representation

Cost estimates

Maintenance/support effort: 0.5
Standard convergence effort: 0
New development effort: 0.25
Total effort: 0.75

Other

The arc* commands are phasing out the ng* commands. The transition is scheduled for the 3rd quarter of this year.

2.3.6 ARC: Janitor

Name Janitor

Description

Software installation & management component with a WS interface

Basic information

Component type: service
Developer: Nordugrid Collaboration

Documentation:

Code repository:

Development language: perl
Build environment: gnu autotools
Download: not yet distributed
Licence: Apache License Version 2.0

Status

Maturity: development
Usage: heavily needed e.g. by the bioinformatics community
Interface maturity: development
Support status: best effort
Support team: NorduGrid Collaboration
Development plans: finish the original design

Distribution/Availability

Middleware stack: will be distributed as part of the technology preview releases of ARC (0.9.x)
Standalone usability: no
Component dependencies: ARC:A-REX
External software dependencies: couple of perl modules
Component consumers: none
Supported platforms: all linuxes
Planned to be supported platforms: mac-osx, windows

Service component information (*only for a service component*)

Public Interfaces: custom WS interface for managing application software environment
Aligned security: under development

Capabilities:

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional standards: none

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0.75
Total effort: 1

2.3.7 gLite: WMS

Name gLite Workload Management System (WMS)

Description

The Workload Management System (WMS) is a service responsible for the distribution and management of computational jobs across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed. A job is expressed in a flexible Job Description Language, based on Condor ClassAds. Different types of jobs are supported: simple batch jobs, DAGs, collections, parametric. On top of them other types of jobs, such as MPI and interactive, can be implemented.

Basic information

Component type: *Service*
Developer: EGEE3 JRA1/Workload Management System team
Documentation: <https://edms.cern.ch/document/572489/1>
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language: C++
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache license

Status

Maturity: *production*
Usage: EGEE sites
Interface maturity: *production*
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1 distribution
Standalone usability: no
Component dependencies: LB, BDII, depending on job request (MyProxy, LFC)
External software dependencies:
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces: WMPProxy (WSDL-based)
Aligned security:
Capabilities:

Library component information

Consumers: gLite WMS User Interface (UI)
Language binding: Java, C++, Python

Interoperability/Standards

Standard-compliance: *none*
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

2.3.8 gLite: WMS Clients

Name gLite Workload Management System (WMS) Clients

Description

The WMS clients allow a user or a higher-level application to access the job management services made available by the WMS

Basic information

Component type: *Client*
Developer: EGEE3 JRA1/Workload Management System team
Documentation: <https://edms.cern.ch/document/572489/1>
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language: C++/Python
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache license

Status

Maturity: *production*
Usage: EGEE sites
Interface maturity: *production*
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3 JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1 distribution
Standalone usability: no

Component dependencies: gLite Workload Management System

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance: none

Interoperable components:

Additional Standards: none

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

2.3.9 gLite: LB

Name gLite Logging and Bookkeeping (LB) Service

Description

The primary purpose of the Logging and Bookkeeping service (L&B) is tracking Grid jobs as they are processed by various Grid middleware components. The information that is gathered is used to inform Grid users on the job state, can be used for debugging and detecting problems, and can also serve as an input for various Grid monitoring and statistics tools. The data are gathered in the form of L&B events that are logged actively by middleware components. The L&B infrastructure concentrates the events on a per-job basis and processes the information to give a high-level view on the job state that is presented to the user.

Basic information

Component type: service

Developer: EGEE3 JRA1/LB team

Documentation: <http://egee.cesnet.cz/en/JRA1/LB/>

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language: C/C++

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache license

Status

Maturity: production

Usage: EGEE sites

Interface maturity: production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability: no

Component dependencies: gSOAP-plugin

External software dependencies: MySQL, gSOAP

Component consumers: gLite WMS

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces: Legacy, includes C/C++ bindings for client applications

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance: none

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

2.3.10 gLite: CREAM Service

Name CREAM Computing Resource Execution and Management (CREAM) Service

Description

The CREAM (Computing Resource Execution And Management) Service is a web service for job management operations at the Computing Element (CE) level. Its interface is well-defined using the Web Service Description Language (WSDL). The main functionality of CREAM is job submission: CREAM jobs are specified using the same Job Description Language (JDL) used to specify the characteristics and requirements of jobs submitted to the gLite WMS. Other operations supported by CREAM include job cancellation, job status (with configurable level of verbosity and filtering based on submission time and/or job status), job list, job suspension and resume, job purge for terminated jobs. CREAM also allows automatic disabling of new job submissions, when certain conditions are met, e.g. when the number of active jobs is higher than a specified threshold or if the CE administrator explicitly requires this. The interface with the underlying Local Resource Management System (LRMS) is implemented via Batch Local Ascii Helper (BLAH).

Basic information

Component type: Service

Developer: EGEE3 JRA1/CREAM team

Documentation:

CREAM User's Guide <https://edms.cern.ch/document/595770>;

CREAM JDL Attributes Specification <https://edms.cern.ch/document/592336>;

CREAM Java CLI Documentation

<http://grid.pd.infn.it/cream/field.php?n=CREAM.CREAMJavaCLIDocumentation>

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language: Java

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage: EGEE sites

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans: Implement failover mechanisms to enhance resiliency; implement additional standards-compliant job management interface.

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability: Yes

Component dependencies: BLAH, LCAS/LCMAPS, gLite Delegation Service

External software dependencies: Apache Axis 1.4; Apache Tomcat application server

Component consumers: gLite: CREAM Command Line Tools, ICE (Interface to CREAM Environment)

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces: Legacy (WSDL-based), includes Java and C++ bindings

Aligned security: TLS/SSL with Globus-style X509 proxies with VOMS Extensions

Capabilities:

Interoperability/Standards

Standard-compliance: non-UMD (OGSA-BES/JSDL only in development branch, not used in production)

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

Other

A prototype standard BES/JSDL interface has been implemented in a development branch of the gLite CREAM service. This branch is currently not maintained and unsupported, as the standard BES/JSDL specifications proved not to be appropriate for production use. We plan to enhance the gLite CREAM service by providing an additional standards-compliant interface based on the outcome of the OGF Production Grid Infrastructures (PGI) Working Group.

2.3.11 gLite: CREAM Command Line Tools

Name gLite: CREAM CLI Command Line Tools

Description

The gLite CREAM CE has its own set of CLI tools, which can be used for direct job submission (as opposed to submission through the gLite Workload Management System, for which other command-line tools must be used). The CREAM CLI is made of a collection of commands implemented in C++ using the gSOAP engine for implementing the client-side Web Service interface. Using the CREAM CLI users can access all the operations of the CREAM interface, and submit and manage their jobs. Note that interactions using the CREAM CLI are subject to the same authentication/authorization mechanisms which are employed for job management through the gLite WMS. Authentication and authorization are enforced by CREAM itself, so they are in place regardless of the specific clients which are being used.

Basic information

Component type: Client

Developer: EGEE3 JRA1/CREAM team
Documentation:
CREAM Client API C++ Tutorial and Documentation
<http://grid.pd.infn.it/cream/field.php?n=Main.CREAMClientAPITutorialAndDocumentation>
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language: C++
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production
Usage: EGEE sites
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability: No
Component dependencies: gLite CREAM, gSoap-plugin
External software dependencies: gSoap

Component consumers:

Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional Standards: none

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

2.3.12 gLite: LCG-CE

Name gLite LCG-CE

Description

LCG-CE is a native computing resource access service with Globus Gatekeeper. LCG has modified some of its component to improve its performance like job manager.

Basic information

Component type: Service
Developer: EGEE3 JRA1
Documentation: <http://glite.web.cern.ch/glite/packages/R3.1/deployment/lcg-CE/lcg-CE.asp>
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language:
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production
Usage: EGEE sites
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans: The LCG-CE will be replaced by the CREAM CE

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability: Yes
Component dependencies:
External software dependencies:
Component consumers: gLite WMS (in particular, the gLite Job Controller/Log Monitor modules)
Supported platforms: SLC4/x86
Planned to be supported platforms:

Service component information

Public Interfaces: Legacy
Aligned security: TLS/SSL with *Globus-style X509 Proxy Certificates with VOMS extensions*
Capabilities:

Interoperability/Standards

Standard-compliance: none
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

2.3.13 UNICORE: TSI

Name UNICORE Target System Interface (TSI)

Description

The UNICORE TSI is the lowest layer of the UNICORE middleware stack and responsible for interactions with one local resource management system (RMS) that performs the scheduling on a specific computational resource. For each of the supported RMSs (e.g. Torque, Slurm, PBSPro, LoadLeveler, etc.) a dedicated TSI exists. Most of them are implemented in Perl. UNICORE jobs are forwarded by the enhanced Network Job Supervisor (XNJS) to the respective TSI for the execution on a computational resource.

Basic information

Component type: *service*
Developer: UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/system-layer.php#anchor_tsi (2009-03)
Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/tsi/>
Development language: Perl
Build environment: UNIX

Download:
<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>
Licence: BSD License

Status

Maturity: *production*
Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*
Interface maturity: *production*
Support status: *long term*
Support team: *supported by UNICORE SourceForge Community*
Development plans: *Refactoring for one DRMAA-based Java TSI, improved batch system support*

Distribution/Availability

Middleware stack: *UNICORE 6.2*
Standalone usability: *Yes*
Component dependencies: *none*
Software dependencies from external providers: *Perl*
Component consumers: *UNICORE XNJS*
Supported platforms: *all systems that can run Perl*
Planned to be supported platforms: *None*

Service component information

Public Interfaces: *Proprietary interface*
Aligned security: *none*
Capabilities: *executionmanagement.jobexecution*

Interoperability/Standards

Standard-compliance: *None*
Interoperable components: *None*
Additional Standards: *DRMAA (development)*

Cost estimates

Maintenance/support effort: *0.125*
Standard convergence effort: *0.125*
New development effort: *0.25*
Total effort: *0.5*

2.3.14 UNICORE: XNJS

Name The UNICORE enhanced Network Job Supervisor (XNJS)

Description

The UNICORE enhanced Network Job Supervisor (XNJS) is the execution backend of the UNICORE service layers (UNICORE Atomic Services or OGSA-BES). It tracks the job progress and interacts with a specific TSI on a respective computational resource. The XNJS is capable of transforming an abstract JSDL-based job submission to a concrete site-specific execution using the UNICORE Incarnation Database (IDB).

Basic information

Component type: *library*
Developer: *UNICORE SourceForge Community*
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_xnjs
Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/xnjs/>
Development language: *Java*

Build environment: Apache Maven
Download:
<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>
Licence: BSD License

Status

Maturity: *production*
Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*
Interface maturity: *production*
Support status: *long-term*
Support team: *UNICORE SourceForge Community*
Development plans:

Distribution/Availability

Middleware stack: UNICORE 6.2
Standalone usability: **yes**
Component dependencies: *UNICORE TSI.*
External software dependencies: none
Component consumers: The UNICORE Atomic Services (UAS) and UNICORE OGSA-BES rely on this particular execution backend.
Supported platforms: *All Java-supported platforms*
Planned to be supported platforms: *None*

Library component information

Consumers: *UNICORE Atomic Services, UNICORE OGSA-BES, UNICORE Workflow service*
Language binding: *Java*

Interoperability/Standards

Standard-compliance: *non-umd (JSDL)*
Interoperable components: *None*
Additional Standards: *File transfers support HTTP, FTP, GridFTP*

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.25
New development effort: 0.5
Total effort: 1

2.3.15 UNICORE: UAS (Job Submission and Management)

Name The UNICORE Atomic Services (UAS) elements TargetSystemFactory service, TargetSystem service, and JobManagement service

Description

The WS-RF-based UNICORE Atomic Services (UAS) are the proprietary Web service interfaces of the UNICORE middleware. Among other WS related to file transfers and storage management, three services are related to job submission and management. Firstly, the TargetSystemFactory service acts as a factory for user-specific TargetSystem service instances that represent computational resources. One TargetSystem service in turn allows for the submission of JSDL-compliant jobs that can be handled via JobManagement service instances. The UAS rely on the XNJS for optimal execution. For managing the job's working directory and handling file transfers, the StorageManagement and FileTransfer service are used.

Basic information

Component type: *service*

Developer: UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_uas (2009-03)
<http://www.unicore.eu/documentation/manuals/unicore6/unicorex/index.html>
Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/unicorex/trunk/uas-core/>
Development language: Java
Build environment: Apache Maven
Download:
<http://downloads.sourceforge.net/unicore/unicore-servers-6.2-0-p1.tgz>
Licence: BSD License

Status

Maturity: production
Usage: DEISA, D-Grid, SKIF-Grid, commercial partners
Interface maturity: production
Support status: long-term
Support team: UNICORE SourceForge Community

Development plans:

Distribution/Availability

Middleware stack: UNICORE 6.2
Standalone usability: Yes
Component dependencies: UNICORE/X service container, UNICORE storage and filetransfer services
External software dependencies: none
Supported platforms: All Java-based platforms
Planned to be supported platforms: None

Service component information

Public Interfaces: TargetSystemFactory service, TargetSystem service, JobManagement service
Aligned security: TLS/SSL (optional with proxies) and SAML Assertions in the SOAP Header
Capabilities: executionmanagement.jobmanager

Interoperability/Standards

Standard-compliance: non-umd (JSDL)
Interoperable components: *None*
Additional Standards: WS(RF)

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.5
New development effort: 0.25
Total effort: 1

2.3.16 UNICORE: OGSA-BES Adoption

Name UNICORE OGSA – Basic Execution Services (BES) Adoption

Description

The OGSA-BES services of UNICORE represent the standard alternative to the job submission and management elements of the UNICORE UAS.

Basic information

Component type: *service*
Developer: FZJ, UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_ogsa
Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/unicorex/trunk/ogsabes-core/>

Development language: Java
Build environment: Apache Maven
Download:
<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>
Licence: BSD License

Status

Maturity: pre-production
Usage: WISDOM, Virtual Physiological Human, EUFORIA
Interface maturity: production
Support status: long-term
Support team: supported by FZJ and the UNICORE SourceForge Community
Development plans: Push for real production usage

Distribution/Availability

Middleware stack: UNICORE 6.2
Standalone usability: No
Component dependencies: UNICORE/X container
External software dependencies: none
Component consumers:
Supported platforms: All Java-based platforms
Planned to be supported platforms: None

Service component information

Public Interfaces: BESFactory service, BESManagement service, BESActivity service
Aligned security: TLS/SSL (optional with proxies) and SAML assertions in SOAP header
Capabilities: executionmanagement.jobmanager

Interoperability/Standards

Standard-compliance: non-umd (*OGSA-BES JSDL*)
Interoperable components: *interoperability with component CREAM-BES of gLite and A-REX of ARC, GridSAM of OMII-UK, and other OGSA-BES components*
Additional Standards: *HPC-Basic Profile, WS(RF)*

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.25
New development effort:
Total effort: 0.5

2.3.17 UNICORE: Workflow Engine

Name UNICORE Workflow Engine

Description

The workflow support in UNICORE 6 is implemented as a two-layered architecture consisting of a workflow engine and the service orchestrator layers. Besides enhanced scalability this also has the benefit of a pluggable workflow engine. Different workflow description dialects, which comply to domain-specific requirements, can be plugged in. Due to the service oriented approach, even plugging in a complete different workflow engine is possible. The currently available workflow engine is based on the open-source Shark XPDL engine and was developed within the Chemomomentum project.

Basic information

Component type: *service*

Developer: FZJ and UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_workflow
Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/workflow>
Development language: Java
Build environment: Apache Maven
Download:
<http://downloads.sourceforge.net/unicore/unicore-workflow-6.1.3.tgz>
Licence: BSD License

Status

Maturity: *pre-production*
Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners* Interface maturity: *production*
Support status: *long-term*
Support team: *support by FZJ and UNICORE SourceForge Community*
Development plans: *feature enhancements, production hardening*

Distribution/Availability

Middleware stack: UNICORE 6
Standalone usability: **Yes**
Component dependencies: UNICORE Service Orchestrator
External software dependencies: none
Component consumers:
Supported platforms: All Java-based platforms
Planned to be supported platforms: None

Service component information

Public Interfaces: *WorkflowFactoryservice, WorkflowManagement service*
Aligned security: *TLS/SSL (optional proxies) and SAML assertions in SOAP Headers*
Capabilities: *executionmanagement.executionandplanning*

Interoperability/Standards

Standard-compliance: *None*
Interoperable components: *None*
Additional Standards: *WS(RF)*

Cost estimates

Maintenance/support effort: *0.25*
Standard convergence effort: *n/a*
New development effort: *0.25*
Total effort: *0.5*

2.3.18 UNICORE: Service Orchestrator

Name UNICORE Service Orchestrator

Description

The service orchestrator layer is responsible to executing the individual tasks in a workflow, handling job execution and monitoring on the Grid. Different brokering strategies are implemented to find the best suited resources for each workflow step. Other brokering strategies can easily be plugged-in. To increase performance and scalability of the service orchestrator layer, multiple instances can be installed.

Basic information

Component type: *service*
Developer: *FZJ & UNICORE SourceForge Community*

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_servorch

Code repository:

Development language: Java

Build environment: Apache Maven

Download:

http://downloads.sourceforge.net/unicore/unicore-workflow-6.1.3.tgz?use_mirror=switch

Licence: BSD License

Status

Maturity: *pre-production*

Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners* Interface maturity: *production*

Support status: *long-term*

Support team: *supported by FZJ & UNICORE SourceForge Community*

Development plans: *More brokering algorithms, integration of different brokers, standards support (e.g. OGSA-BES)*

Distribution/Availability

Middleware stack: UNICORE Workflow System Version 6

Standalone usability: **Yes**

Component dependencies: UNICORE Atomic services

External Software dependencies: *none*

Component consumers: *UNICORE Workflow service*

Supported platforms: *All Java-based platforms*

Planned to be supported platforms: *None*

Service component information

Public Interfaces: *ServiceOrchestrator service*

Aligned security: *TLS/SSL (optional proxies) and SAML assertions in SOAP Headers*

Capabilities: *executionmanagement.executionandplanning*

Interoperability/Standards

Standard-compliance: *None*

Interoperable components: *None*

Additional Standards: *WSRF*

Cost estimates

Maintenance/support effort: *0.125*

Standard convergence effort: *0.25*

New development effort: *0.25*

Total effort: *0.625*

2.3.19 UNICORE: Rich Client

Name UNICORE Rich Client

Description

The Eclipse-based UNICORE Rich Client (URC) offers the full set of functionalities to the users in a graphical representation. It targets a wide range of users with varying Grid and IT experience. It provides a useful graphical view of the Grid, which can be filtered in order to find specific resources, services or files. Detailed resource requirements for jobs (e.g. required number of CPUs, amount of RAM) can be specified. Most notably, it allows for powerful mechanisms to construct directed acyclic graphs (DAGs) used for Grid workflows.

Basic information

Component type: *client*

Developer: *UNICORE SourceForge Community*
Documentation: http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_urc
Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/eclipseclient/>
Development language: Java
Build environment: Apache Ant, Apache Maven
Download: http://sourceforge.net/project/showfiles.php?group_id=102081&package_id=263955
Licence: BSD License

Status

Maturity: *production*
Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners* Interface maturity: *production*
Support status: *long-term*
Support team: supported by FZJ and UNICORE SourceForge Community
Development plans: Improve and extend functionality

Distribution/Availability

Middleware stack: *UNICORE 6*
Standalone usability: Yes
Component dependencies: UNICORE Atomic services
Supported platforms: *All Java-based platforms (except Mac OS/X)*
Planned to be supported platforms: *Mac OS/X*

Interoperability/Standards

Standard-compliance: *None*
Interoperable components: *None*
Additional Standards: *Web Service Resource Framework (WS-RF), Eclipse rich client platform*

Cost estimates

Maintenance/support effort: *0.5*
Standard convergence effort: *0.5*
New development effort: *0.5*
Total effort: *1.5*

2.3.20 UNICORE: Command-line Client

Name UNICORE Command-line Client (UCC)

Description

The UNICORE command line client (UCC) is a very versatile command-line tool that allows users to access all features of the UNICORE service layer in a shell or scripting environment. It allows to run jobs, monitor their status and retrieve generated output, both in single job mode or in a powerful and flexible batch mode for multiple jobs. Additionally, workflows can be submitted and controlled with the UCC. Via extensions it also allows for OGSA-BES-based job submissions.

Basic information

Component type: *client*
Developer: *UNICORE SourceForge Community*
Documentation:
http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_ucc (2009-03)
Code repository:
<http://unicore.svn.sourceforge.net/viewvc/unicore/ucc/>
Development language: Java
Build environment: Java
Download: http://sourceforge.net/project/showfiles.php?group_id=102081&package_id=263954
Licence: BSD

Status

Maturity: *production*

Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*

Interface maturity: *production*

Support status: *long-term*

Support team: *FZJ and UNICORE SourceForge community*

Development plans: *keep up with server-side developments, new features*

Distribution/Availability

Middleware stack: *UNICORE 6*

Standalone usability: *Yes*

Component dependencies: **UNICORE Atomic services**

External software dependencies: *none*

Component consumers:

Supported platforms: *All Java-based platforms*

Planned to be supported platforms: *None*

Interoperability/Standards

Standard-compliance: *non-umd (OGSA-BES)*

Interoperable components: *ARC A-REX and CREAM-BES*

Additional Standards: *Web Services Resource Framework (WS-RF)*

Cost estimates

Maintenance/support effort: *0.125*

Standard convergence effort: *0.125*

New development effort: *0.25*

Total effort: *0.5*

2.3.21 UNICORE: HILA

Name UNICORE High level API

Description

HiLA is a High Level API for Grid Applications, that allows simple development of clients with just a few lines of code for otherwise complex functionality. It can also be used to integrate UNICORE 6 in user applications to e.g. directly submit simulation jobs to Grid resources without using UNICOREs command line or graphical clients. HiLA provides a single interface with multiple implementations for UNICORE 5, UNICORE 6 and OGSA-BES. HiLA is used to integrate UNICORE 6 access into **DESHL** (including SAGA support), as part of the DEISA portal, and to connect **GAT** with UNICORE 6. The nature of the HiLA API leads to a concise coding toolkit for building collective tier Grid services and client interfaces. Following the UNICORE principle of seamlessness, the design of the API models the Grid through a object-oriented facade, presenting abstract representations of the underlying resources. Importantly, this includes encapsulating security configuration behind well defined interfaces, further enhancing the uncluttered API.

Basic information

Component type: *library*

Developer: *UNICORE SourceForge Community*

Documentation:

http://www.unicore.eu/unicore/architecture/client-layer.php#anchor_hila (2009-03)

Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/hila>

Development language: *Java*

Build environment: Apache Maven
Download: http://sourceforge.net/project/showfiles.php?group_id=102081&package_id=225019
Licence: BSD License

Status

Maturity: *pre-production*
Usage: *DEISA*
Interface maturity: *pre-production*
Support status: *long-term*
Support team: FZJ and the UNICORE SourceForge Community
Development plans: improve for production use, new features

Distribution/Availability

Middleware stack: *UNICORE 6*
Standalone usability: *Yes*
Component dependencis: UNICORE Atomic Services
External software dependencies:
Component consumers:
Supported platforms: *All Java-based Platforms*
Planned to be supported platforms: *None*

Library component information (*only relevant for a library*)

Consumers: *JavaGAT, DESHL*
Language binding: *Java*

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards: *WS-RF*

Cost estimates

Maintenance/support effort: *0.125*
Standard convergence effort: *0.125*
New development effort: *0.25*
Total effort: *0.5*

2.4 New development

To be added later.

3 Data functionality

3.1 Standards landscape

To be provided later...

Overview of what existing, followed data related relevant specifications..

Discussion whether there are already existing set of standards, which can be used to achieve std-based interoperability of UMD data components.

Recommendation of what an UMD data component should comply with

3.2 Overview table: data functionality

	Name	Type	Maturity	Standard compliance		
3.3.1	ARC: Classic SE	service	production	none		
3.3.2	ARC: libarcdata2	library	Pre-production	Non-umd		
3.3.3	ARC: ng* data management CLI	client	production	Non-umd		
3.3.4	ARC: arc* data management CLI	client	Pre-production	Non-umd		
3.3.5	ARC: Chelonia	service	Pre-production	none		
3.3.6	ARC: Chelonia client	client	Pre-production	none		
3.3.7	gLite: DPM	service	production	Non-umd		
3.3.8	gLite: lcg_utils	client	production	???		
3.3.9	gLite: htcp	client	production	???		
3.3.10	gLite: SRM client	client	production	???		
3.3.11	gLite: DPM client	client	production	???		
3.3.12	gLite: LFC client	client	production	???		
3.3.13	gLite: FTS client	client	production	???		
3.3.14	gLite: DPM/DICOM interface	client	???	???		
3.3.15	gLite: LCG File Catalogue (LFC)	service	production	???		
3.3.16	gLite: File Transfer Service (FTS)	service	production	???		
3.3.17	gLite: Hydra	service	production	???		
3.3.18	gLite: AMGA	service	production	???		
3.3.19	gLite: AMGA Client API	library	???	???		
3.3.20	UNICORE: UAS (Storage Management and File Transfer)	service	production	Non-umd		
3.3.21	Other: dCache	Service (?)	production	???		

3.3 Inventory of components

3.3.1 ARC: Classic SE

Name ARC Classic SE

Description

A Gridftp server developed using the Globus libraries. The Classic SE comes with powerful backend plugins presenting various information using FTP interface. Thos include: jobplugin for Grid job submission and management, gacplugin for storing ordinary files with assigned access policies in GACL language, fileplugin for ordinary FTP interface.

Basic information

Component type: {service, library, client}

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: {planned, development, pre-production, production}

Usage:

Interface maturity: {development, pre-production, production}

Support status: {abandoned, best effort, short term, long term}

Support team:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:

Aligned security:

Capabilities:

Library component information *(only for a library component)*

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}

Interoperable components:

Additional standards:

Cost estimates

Maintenance/support effort: 0.25

Standard convergence effort: 0

New development effort: 0
Total effort: 0.25

Other

Any other kind of information

3.3.2 ARC: libarcdata2

Name ARC data management library (libarcdata2)

Description

The component implements a general purpose data library with support for multiple data transfer and management protocols. Due to its pluggable architecture it is relatively easy to add support for new protocols. Currently supported protocols are: HTTP(S,G), (Grid)FTP, LDAP, LFC, RLS, SRM and Chelonia.

The libarcdata is targeted for grid developers and advanced application integrators and provides a general toolkit for high-level cross-middleware, multiplatform client side development.

The library provides the core of the data management layer of the highly successful and powerful, nevertheless lightweight ARC clients, grid portals and client-side job managers.

Basic information

Component type: library

Developer: Uppsala University and University of Oslo from NorduGrid Collaboration

Documentation:

<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/dmc/dmc.pdf>

Code repository:

<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/dmc>

<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/libs/data>

Development language: C++

Build environment: GNU autotools

Download: download.nordugrid.org

Licence: Apache License version 2.0

Status

Maturity: pre-production

Usage: as the underlying library of all ARC data management the library will be widely deployed

Interface maturity: pre-production

Support status: short term support by KnowARC project, long term support by NorduGrid collaboration

Support team: Uppsala University

Development plans: enhance maturity, decouple from gsoap

Distribution/Availability

Middleware stack: technology preview releases of ARC (0.9.x)

Standalone usability: no

Component dependencies: ARC:HED

External software dependencies: globus data libraries (globus-ftp-client, globus-rsl-client, LFC-client-libs), gsoap, openldap

Component consumers: ARC: arc* CLI

Supported platforms: all the major flavours of linuxes, mac-osx

Planned to be supported platforms: windows (most of the effort needed is to port the gobus dependencies)

Library component information *(only for a library component)*

Consumers: ARC: arc* CLI

Language binding: python, java

Interoperability/Standards

Standard-compliance: non-UMD

Interoperable components: any storage system with SRM interface

Additional standards: gridftp, HTTP(S), SRM storage services are supported

Cost estimates

Maintenance/support effort: 1

Standard convergence effort: 0.5

New development effort: 0.5

Total effort: 2

Other

The library is scheduled for inclusion into production ARC release for the 3rd quarter of this year.

3.3.3 ARC: ng* data management CLI

Name ng* data management CLI

Description

The ng* data management tools (ngls, ngcp, ngrm, ngtransfer, ngstage) are based on older version of libarcdata library and feature support for expandable set of data protocols. They are powerful command line tools support listing, removing and copying data.

Basic information

Component type: client

Developer: Uppsala University and Oslo University of NorduGrid Collaboration

Documentation:

Code repository:

Development language: c++

Build environment: autotools

Download: download.nordugrid.org

Licence: Apache License version 2.0

Status

Maturity: production

Usage: widely used as the client-side data handling utility

Interface maturity: production

Support status: long term

Support team: Oslo University

Development plans: none

Distribution/Availability

Middleware stack: production ARC releases (latest version 0.6.5)

Standalone usability: yes

Component dependencies: none

External software dependencies:

Component consumers: none, it is a standalone client

Supported platforms: all the major linux flavours

Planned to be supported platforms: none

Interoperability/Standards

Standard-compliance: non-UMD

Interoperable components: all the SRM, gridftp-based storage solutions

Additional standards: supports SRM, gridftp

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0
Total effort: 0.25

Other

This component is being gradually phased out by the next generation arc* data management clients.

3.3.4 ARC: arc* data management CLI

Name arc* data management client tools

Description

The arc* data management tools (arccp, xxx) are based on the libarcdata2 library and feature support for several data protocols. Additional protocols can be added by writing corresponding plugins. These command line tools support listing, removing and copying of data.

Comment [BK1]: List them

Basic information

Component type: client
Developer: Uppsala University and Oslo University from NorduGrid Collaboration
Documentation:
Code repository:
Development language: c++
Build environment: autotools
Download: download.nordugrid.org
Licence: Apache License version 2.0

Status

Maturity: pre-production
Usage: it will be widely used as the main client-side data handling utility replacing the ng* data CLI
Interface maturity: pre-production
Support status: short term support by KnowARC project, long term support by NorduGrid collaboration
Support team: Uppsala University
Development plans: support for third party transfers

Distribution/Availability

Middleware stack: technology preview releases of ARC (0.9.x)
Standalone usability: yes
Component dependencies: ARC: libarcdata2
External software dependencies:
Component consumers: none, it is a standalone client
Supported platforms: Linux, MacOS, Solaris
Planned to be supported platforms: windows

Interoperability/Standards

Standard-compliance: non-UMD
Interoperable components: any SRM-based storage system
Additional standards: supports SRM, gridftp, http(s)

Cost estimates

Maintenance/support effort: 0.5
Standard convergence effort: 0
New development effort: 0.5
Total effort: 1

Other

This component is gradually phasing out the old ng* data management clients. The tools are scheduled for inclusion into production ARC release for the 3rd quarter of the year.

3.3.5 ARC: Chelonia**Name ARC Chelonia storage system****Description**

Chelonia is a self-healing data storage system which is designed to avoid single points of failures and bottlenecks. The system shows a hierarchical namespace of files and collections (which can contain files and other collections) to the user. This namespace is global in the sense that each user of the system sees exactly the same tree hierarchy of files (maybe some parts are hidden for some users), it is very similar to the concept of a multi-user operating system with one local filesystem and multiple users. If a user changes something in the namespace of the storage, all the other users will see the changes immediately. The files are replicated and the broken replicas are repaired automatically. The files are referred by Logical Names (LNs), which are paths in the global namespace. It is possible to integrate the namespace of third-party storage systems into this namespace in a similar way as mounting remote filesystems into a local filesystem. For each file and collection there are access policies defined in which it can be specified in terms of individual users or Virtual Organizations who can access or modify them. Chelonia aims to be an easy-to-deploy, easy-to-operate nevertheless robust standalone storage solution.

Basic information

Component type: service
Developer: NIIF Institute, Uppsala University and Oslo University from NorduGrid Collaboration
Documentation:
http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/storage_system/arc-storage-documentation.pdf
Code repository:
<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/services/storage>
Development language: python
Build environment: autotools
Download: download.nordugrid.org
Licence: Apache License version 2.0

Status

Maturity: pre-production
Usage: first will be deployed in the Hungarian ClusterGrid
Interface maturity: production
Support status: long term
Support team: *NIIF Institute from NorduGrid Collaboration*
Development plans: performance and stability improvements, add support for SRM interface

Distribution/Availability

Middleware stack: technology preview releases of ARC (0.9.x)
Standalone usability: no
Component dependencies: ARC: libarcdata2; ARC: HED

External software dependencies:

Component consumers: ARC: arc* data cli
Supported platforms: Linux, Mac OS X, Solaris
Planned to be supported platforms: Windows

Service component information

Public Interfaces: propriatery WS-based interface
Aligned security: TLS/SSL, proxies are supported, VOMS attributes are supported
Capabilities: data.access.flatfile, data.management.replica, data.management.storage, data.transfer, data.naming.resolver

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional Standards: bytelO, HTTP(s)

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.5
New development effort: 0.5
Total effort: 1.25

Other

Chelonia is scheduled to be included into the production ARC releases for 1st quarter of 2010.

3.3.6 ARC: Chelonia client

Description

Client solutions for the Chelonia storage system, including a command line tool and a module for FUSE enabling transparent access to Chelonia-enabled Grid storage resources.

Basic information

Component type: client
Developer: NIIF Institue, Uppsala University and Oslo University from NorduGrid Collaboration
Documentation:
http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/storage_system/arc-storage-documentation.pdf
Code repository: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/services/storage>
Development language: python
Build environment: autotools
Download: download.nordugrid.org
Licence: Apache License version 2.0

Status

Maturity: pre-production
Usage: The client tools will be tested by the Hungarian ClusterGrid users
Interface maturity: production
Support status: long term
Support team: NIIF, Oslo from NorduGrid Collaboration
Development plans: increase user-friendliness and ease-of-use, GUI tools

Distribution/Availability

Middleware stack: technology preview releases of ARC (0.9.x)
Standalone usability: no
Component dependencies: ARC: libarcdata2, ARC:HED

External software dependencies:

Component consumers: none (end users)
Supported platforms: Linux, Mac OS X, Solaris
Planned to be supported platforms: Windows

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional standards: HTTP(s)

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0.25
Total effort: 0.5

Other

Chelonia clients are scheduled for ARC production release inclusion for 1st quarter of 2010.

3.3.7 gLite: DPM**Name** gLite Disk Pool Manager (DPM)**Description**

The light-weight Disk Pool Manager (DPM) offers a simple solution for a disk-only Storage Element. It is easy to install and configure and requires very low maintenance effort. The DPM consists of a set of servers with their own client interface. All the interfaces work only in secure mode using X509 certificates or Kerberos 5 tokens and can use VOMS based ACLs. The DPM is deployed in about 120 sites within EGEE.

Basic information

Component type: Service
Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team

Development plans:**Distribution/Availability**

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:
External software dependencies:
Component consumers:
Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security: X509 certificates or Kerberos 5 tokens and can use VOMS based ACLs

Capabilities:

Interoperability/Standards

Standard-compliance: non-umd

Interoperable components:

Additional Standards: SRM, rfi, GridFTP, HTTP(S), xrootd

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.8 gLite lcg_utils

Name gLite lcg_utils

Description

In the lcg_utils a mixture of GFAL and Globus functionality was used to create Grid equivalents of UNIX copy commands and registration that spans two storage systems and catalogues of choice (i.e. LFC, EDG). Other functions have been implemented, such as getting the list of file replicas from many sources, performing lookups on LFNs or GUIDs, changing the status of the file or finally removing it from the Grid.

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.9 gLite: htcp

Name gLite htcp

Description

htcp is a client to fetch files or directory listings from remote servers using HTTP or HTTPS, or to put or delete files or directories onto remote servers using HTTPS. htcp is similar to the scp command, but uses HTTP/HTTPS rather than ssh as its transfer protocol.

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:
New development effort:
Total effort:

3.3.10 gLite: SRM client

Name gLite SRM client

Description

Command line Client for the SRM interface

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.11 gLite: DPM client

Name gLite DPM client

Description

Command line Client for the DPM interface

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/i386

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.12 gLite: LFC client

Name gLite LFC client

Description

Command line Client for LFC

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.13 gLite: FTS client

Name gLite FTS client

Description

Command line Client for FTS

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.14 gLite: DPM/DICOM interface

Name gLite DPM/DICOM interface

Description

The Digital Imaging and Communications in Medicine (DICOM) is a standard medical image storage system used by Biomedical researchers, an important user community in EGEE. gLite includes an interface between DICOM and the DPM storage element. This DPM/DICOM interface is a command line client that takes anonymized medical images from a DICOM system, enters the image metadata into a metadata storage system such as AMGA, uses EDS and Hydra to encrypt the image, stores the resulting file in DPM and stores storage metadata in LFC. These encrypted files may then be analyzed or studied in a Grid environment. The planned support effort for DPM/DICOM is driven by the requests from the user community, currently the Biomedical researchers.

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity:

Usage:

Interface maturity:

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

3.3.15 gLite: LCG File Catalogue (LFC)

Name LCG File Catalogue (LFC)

Description

The LFC offers a hierarchical view of files to users and provides Logical File Name (LFN) to Storage URL (SURL) mappings (via a GUID) with authorization on its logical namespace. Read-only distributed catalog can be achieved by using Oracle streams. The LFC is deployed at most EGEE European and Asian sites, in addition to the CERN site. It supports several deployment scenarios; central, local and local readonly replica of global.

Basic information

Component type: Service
Developer: EGEE3 JRA1
Documentation:
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language:
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production
Usage: most EGEE European and Asian sites, in addition to the CERN site
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.16 gLite: File Transfer Service (FTS)

Name gLite File Transfer Service (FTS)

Description

The gLite File Transfer Service (FTS) is a low-level data movement service for transferring files between Storage Elements. In addition, it provides features for administration and monitoring of these transfers. The FTS exposes an interface to submit asynchronous bulk requests and performs the transfers using either third-party GridFTP or SRM Copy. The FTS servers are typically deployed at (large) sites where there are large amounts of data to be transferred. From the security point of view, the interactions with these external services (mainly SRMs and Storage Elements) always use the client proxy credentials either retrieved from MyProxy or delegated by the GridSite delegation components.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/i386
Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:
Aligned security:
Capabilities:

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

3.3.17 gLite: Hydra

Name gLite Hydra

Description

The symmetric encryption keys for encrypted data (files) storage are stored in a specific set of servers called "Hydra". Hydra provides controlled access to these keys (through certificate DN and VOMS attributes based ACLs) and secured communication to the requester. The Hydra service is a Java Web Service, which can be deployed in a J2EE container, such as Tomcat. It requires a database back-end, and communicates via the HTTPS protocol with its clients. In addition, Hydra exploits the Shamir secret-sharing scheme to improve security and reliability of this service. Shamir's scheme consists of splitting keys into N fragments stored in different places. Only $M < N$ fragments are needed to reconstruct a complete key. However, owning less than M key fragments, does not give any information on the complete key. Thus, the system is both resistant to attacks (at least M key stores need to be compromised for an attacker to be able to reconstruct a key) and reliable (the disconnection of a limited number of servers does not prevent the key reconstruction).

Basic information

Component type: Service
Developer: EGEE3 JRA1
Documentation:
Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>
Development language: Java
Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production
Usage:
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies: Tomcat, MySQL

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security: HTTPS, certificate DN and VOMS attributes based ACLs

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.18 gLite: AMGA

Name AMGA**Description**

AMGA is a metadata service which provides database access for Grid applications. AMGA is implemented as a thin layer between a Grid application and the underlying database and acts as an interface between them, providing a Grid style authentication mechanism. AMGA also provides functionality to make Grid applications interoperable with different database backends. Finally, AMGA's support for replication of relational data can be used to build scalable and reliable Grid applications.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

3.3.19 gLite: AMGA Client API

Name AMGA Client API

Description

Client API for contacting the AMGA service.

Basic information

Component type: API

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity:

Usage:

Interface maturity:

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information**Consumers:****Language binding:****Interoperability/Standards****Standard-compliance:****Interoperable components:****Additional Standards:****Cost estimates****Maintenance/support effort:****Standard convergence effort:****New development effort:****Total effort:**

3.3.20 UNICORE: UAS (Storage Management and File Transfer)

Name The UNICORE Atomic Services (UAS) elements StorageManagement service and File transfer services

Description

The WS-RF-based UNICORE Atomic Services (UAS) are the proprietary Web service interfaces of the UNICORE middleware. Among other WS related to computation a various of services are related to storage management and file transfers. Firstly, the StorageManagement service acts as a representation of various types of storages and the various flavours of the file transfer services allow different types of transfers (e.g. plain HTTP, ByteIO). The UAS rely on the XNJS for optimal execution.

Basic information

Component type: *service*

Developer: UNICORE SourceForge Community

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_xnjs

Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/unicorex/trunk/uas-core/>

Development language: Java

Build environment: Apache Maven

Download:

<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>

Licence: BSD License

Status

Maturity: *production*

Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*

Interface maturity: *production*

Support status: *long-term*

Support team: UNICORE SourceForge Community

Development plans: Add and enhance support for various storage backends (e.g. iRODS, Apache Hadoop)

Distribution/Availability

Middleware stack: UNICORE 6.2

Standalone usability: **Yes**

Component dependencies: *UNICORE/X container*
External software dependencies: *none*
Component consumers: *UNICORE Atomic services, UNICORE OGSA-BES*
Supported platforms: *All Java-based platforms*
Planned to be supported platforms: *none*

Service component information

Public Interfaces: *Proprietary StorageManagement Service and File Transfer Services*
Aligned security: *TLS/SSL (proxies optional), SAML assertions in the SOAP header*
Capabilities: *data.management.storage; data.management.transfer*

Interoperability/Standards

Standard-compliance: *non-umd (HTTP FileTransfer Service)*
Interoperable components: *None*
Additional Standards: *OGSA ByteIO, WSRF*

Cost estimates

Maintenance/support effort: *0.25*
Standard convergence effort: *0.5*
New development effort: *1.0*
Total effort: *1.75*

3.3.21 other: dCache

Name dCache

Description

The goal of dCache is to provide a system for storing and retrieving huge amounts of data, distributed among a large number of heterogeneous server nodes, under a single virtual filesystem tree with a variety of standard access methods. Depending on the Persistency Model, dCache provides methods for exchanging data with backend (tertiary) Storage Systems as well as space management, pool attraction, dataset replication, hot spot determination and recovery from disk or node failures. Connected to a tertiary storage system, the cache simulates unlimited direct access storage space. Data exchanges to and from the underlying HSM are performed automatically and invisibly to the user. Filesystem namespace operations may be performed through a standard NFS(2) interface.

Basic information

Component type: {service, library, client}
Developer:
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: {planned, development, pre-production, production}
Usage:
Interface maturity: {development, pre-production, production}
Support status: {abandoned, best effort, short term, long term}
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Library component information *(only for a library component)*

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}
Interoperable components:
Additional standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

Other

Any other kind of information

3.4 New development

4 Information functionality

4.1 Standards landscape

The information system components of the three middlewares are still utilizing different information models: ARC relies on the NorduGrid schema, gLite makes use of the GLUE1.3 schema while Unicore relies on its own schema as well.

Comment [BK2]: Unicore to comment on it

Fortunately, the OGF GLUE working group after several years of intensive work has released the GLUE v2.0 specification which provides a common information model embraced by all the major middlewares and grid infrastructures. However, GLUE v2.0, as an information model, does not regulate how the information is obtained, queried or exchanged. There is no common such mechanisms and no ongoing relevant standardization work is addressing this requirement.

It is proposed that every information functionality UMD component should comply with the GLUE v2.0 specification. Concerning information publishing/query it is recommended that the currently used mechanisms are to be harmonized by agreeing on some best practices. To satisfy the GLUE v2.0 compliance requirement all the UMD candidate components require considerable development efforts invested into standard convergence.

4.2 Overview table: information area

	Name	Type	Maturity	Standard compliance		
4.3.1	ARC: Classic Infoserver	service	Production	None		
4.3.2	ARC: Classic Infoindex	Service	Production	None		
4.3.3	ARC: ALIS	service	Pre-production	Early-umd		
4.3.4	ARC: ISIS	service	Development	None		
4.3.5	ARC: libarcclient (infosys modules)	library	Pre-production	Early-umd		
4.3.6	ARC: Grid Monitor	client	production	None		
4.3.7	gLite: BDII	service	production	none		
4.3.8	gLite: lcg-info and lcg-infosites	client	production	none		
4.3.9	gLite: Service Discovery API	library	production	None		
4.3.10	UNICORE: CIS	service	Pre-production	Early-umd		
4.3.11	UNICORE: Service Registry	service	production	None		

4.3 Inventory of components

4.3.1 ARC: Classic Infoserver

Comment [BK3]: To be completed

Name ARC Classic Infoserver

Description

A short paragraph describing the component

Basic information

Component type: service

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: production

Usage:

Interface maturity: production

Support status: long term

Support team:

Development plans: none

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Interoperability/Standards

Standard-compliance: none
Interoperable components:
Additional standards: LDAP

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0
Total effort: 0.25

Other

Recently replaced the old globus GRIS-ldap modul dependency by native ldap and BDII

4.3.2 **ARC: Classic Infoindex**

Comment [BK4]: To be filled

Name ARC Classic Infoindex

Description

A short paragraph describing the component

Basic information

Component type: service
Developer:
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: production
Usage:
Interface maturity: production
Support status: long term
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:

Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Interoperability/Standards

Standard-compliance: none
Interoperable components:
Additional standards: LDAP

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0
New development effort: 0
Total effort: 0.25

Other

Recently replaced the old globus GIIS ldap backend with native ldap

4.3.3 ARC: ALIS

Name ARC Local Information System (ALIS)

Description

The ALIS is a component implementing the "local information system" functionality for ARC services. It is based upon a HED library which provides WSRF interface for information query (service and client side) and XML container for storing information. It also features registration functionality for registering services to information collectors/indexing services. Its internal design is made as much independent of external interfaces. That makes it easy to extend supported interfaces without significant changes in services and client tools.

Basic information

Component type: library
Developer: Oslo University, Lund University, NIIF from NorduGrid Collaboration
Documentation:
https://www.knowarc.eu/wiki/images/9/90/Knowarc_D1.2-1_07.pdf (Section 2)
<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/ARC1-API.pdf>
(Arc::InfoRegisterContainer and related classes for API reference)
Code repository: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed/libs/infosys>
Development language: C++
Build environment: GNU autotools
Download: download.nordugrid.org
Licence: Apache License Version 2.0

Status

Maturity: pre-production
Usage: used by every ARC services which are implemented in the HED framework
Available Documentation: https://www.knowarc.eu/wiki/images/9/90/Knowarc_D1.2-1_07.pdf
(Section 2) and <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/ARC1-API.pdf>
Interface maturity: development
Support status: short term support by KnowARC project, long term support by NorduGrid collaboration
Support team: NIIF and Oslo University from NorduGrid

Development plans: enhancing maturity and interoperability of implementation, adding caching functionality

Distribution/Availability

Middleware stack: technology preview releases of ARC (version 0.9.x)

Standalone usability: no

Component dependencies: ARC:HED

External software dependencies: libxml

Component consumers: ARC:A-REX, ARC:Chelonia (ARC services use this library to provide information about themselves)

Supported platforms: Linux, MacOS, Solaris, Windows (early support)

Planned to be supported platforms: Windows

Service component information

Public Interfaces: WSRF (Selected functionality), proprietary registration interface

Aligned security: Evaluation of ARC policies embedded into informational document

Capabilities:

Library component information

Consumers: ARC services and client tools

Language binding: C++, Python, Java (not tested)

Interoperability/Standards

Standard-compliance: early-umd (preliminary GLUE v2.0 support)

Interoperable components: None

Additional Standards: WSRF selected functionality

Cost estimates

Maintenance/support effort: 0.25

Standard convergence effort: 0.5

New development effort: 1

Total effort: 1.75

4.3.4 ARC: ISIS

Name ARC Information System Indexing Service (ISIS)

Description

The ISIS is a service which stores information pushed to it by another services and serves as resource registration and discovery point. Multiple ISIS services form peer-to-peer network to provide fault-tolerant information system.

Basic information

Component type: service

Developer: NIIF from NorduGrid Collaboration

Documentation:

<http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/doc/infosys/TechnicalHandbook/ISIS%20-%20TechnicalHandbook.pdf>

Code repository: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/services/isis>

Development language: C++

Build environment: GNU autotools

Download: <http://download.nordugrid.org/software/nordugrid-arc1/releases/latest/>

Licence: Apache License Version 2.0

Status

Maturity: development

Usage: once completed it'll be a key component of ARC-based grids as the ISIS will form the information backbone connecting resources grid-enabled by ARC components

Interface maturity: pre-production

Support status: short term support by KnowARC project, long term support by NorduGrid collaboration

Support team: NIIF

Development plans: enhancing maturity of implementation.

Distribution/Availability

Middleware stack: technology preview releases of ARC (version 0.9.x)

Standalone usability: no

Component dependencies: ARC:HED

External software dependencies: libxml, Glibm

Component consumers: ARC:libarcclient, ARC:A-REX, ARC: Chelonia, etc..

Supported platforms: Linux, MacOS, Solaris, Windows (early)

Planned to be supported platforms: Windows

Service component information

Public Interfaces: WSRF (Selected functionality), proprietary registration interface

Aligned security: Authorization of registering and neighbor services, evaluation of ARC policies embedded into informational document

Capabilities:

Interoperability/Standards

Standard-compliance: none

Interoperable components: None

Additional Standards: WSRF selected functionality

Cost estimates

Maintenance/support effort: 0.25

Standard convergence effort: 0

New development effort: 0.75

Total effort: 1

Other

ISIS will gradually replace the ARC: Classic Infoindex component.

4.3.5 **ARC: libarcclient (infosys modules)**

Comment [BK5]: Fill it

Name The information system modules of libarcclient

Description

The next generation general purpose grid library (libarcclient) offers transparent access to grid resources through an intuitive and easy-to-use Multilanguage API. The libarcclient comes loaded with plenty of power features including resource discovery, information system interrogation and resource selection and brokering as well. It features a pluggable architecture which allows it to implement cross-grid information system query. The current implementation comes with plugins for ARC Classic Infoserver and Infoindex (ldap-based pre-WS solution), A-REX (WSRF-based solution), Unicore (WSRF-based solution) and CREAM (mixture of ldap-based BDII and a proprietary WS-based solution).

Libarcclient is targeted for grid developers and advanced application integrators and provides a general toolkit for high-level cross-middleware, multiplatform client side development.

The library is the core of the highly successful and powerful, nevertheless lightweight ARC clients, grid portals and client-side job managers.

Basic information

Component type: library
Developer:
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: pre-production
Usage:
Interface maturity: pre-production
Support status: {abandoned, best effort, short term, long term}
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Library component information *(only for a library component)*

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance: early-UMD (preliminary GLUE v2.0 support)
Interoperable components: CREAM, BDII, Unicore service
Additional standards: LDAP, WSRF

Cost estimates

Maintenance/support effort: 0.5
Standard convergence effort: 0.5
New development effort: 0.5
Total effort: 1.5

Other

The libarcclient development library is scheduled to be part of the next production ARC release scheduled for 3rd quarter of this year as a replacement of the aging previous generation ARCLIB.

4.3.6 ARC: Grid Monitor

Name ARC Grid Monitor**Description**

The Grid Monitor is a Web client tool for the ARC Information System, allowing browsing all the published information about the system. It makes use of the hierarchical information organization and the PHP LDAP module to provide a real-time monitoring and primary debugging for ARC-

based grids.

Basic information

Component type: Client
Developer: Lund University from NorduGrid Collaboration
Documentation: "The Grid Monitor: Usage Manual". [NORDUGRID-MANUAL-5] (March 2009)
Code repository: <http://svn.nordugrid.org/arc0/trunk/monitor>
Development language: PHP
Build environment: N/A
Download: download.nordugrid.org
Licence: Apache License Version 2.0

Status

Maturity: Production
Usage: deployed for monitoring as part of the ARC-based grids
Interface maturity: Production
Support status: Best effort
Support team: Lund University from NorduGrid
Development plans: Has to be adapted for GLUE-based information systems and WS interfaces

Distribution/Availability

Middleware stack: ARC production releases (latest v0.6.5)
Standalone usability: YES
Component dependencies: none
External software dependencies: ldap, php
Component consumers: none, it is an end-user tool
Supported platforms: All
Planned to be supported platforms: none

Interoperability/Standards

Standard-compliance: none
Interoperable components: none
Additional Standards: LDAP

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.25
New development effort: 0.25
Total effort: 0.75

Other

Is localizable in any human language (currently, 10 languages are available)

4.3.7 gLite: BDII

Name gLite Berkeley Database Information Index (BDII)

Description

The BDII is a core component of the EGEE information system. It is used to index information sources and provides an up-to-date cache of the information from these sources. It is used at the resource, site and top level in the infrastructure.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/i386

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance: none

Interoperable components:

Additional Standards: LDAP, GLUE v1.x

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

4.3.8 gLite: lcg-info and lcg-infosites

Name gLite: lcg-info and lcg-infosites

Description

lcg-info and lcg-infosites are two command line clients which are used to query the information system.

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS
Download: <http://glite.web.cern.ch/glite/packages/R3.1/>
Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/i386

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance: none

Interoperable components:

Additional Standards: LDAP

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

4.3.9 gLite: Service Discovery API

Name gLite Service Discovery API

Description

Service Discovery is a set of libraries that provide an abstraction for finding Grid services and discovering their basic attributes, independently of the underlying Information System infrastructure.

Basic information

Component type: library

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/i386

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

4.3.10 UNICORE: CIS

Name UNICORE Common Information Service (CIS)

Description

The Common Information Service (CIS) is the information service of UNICORE 6. It gathers both static and dynamic information from all connected UNICORE sites, which are then displayed either in raw XML or human-readable text form in some GUI. For instance, as longitude and latitude information is also stored, a Google maps view allows a geographical representation of the Grid infrastructure. CIS is based on the GLUE 2.0 standard from OGF. Like with the UNICORE registry, a central CIS is necessary in an operational UNICORE infrastructure.

Basic information

Component type: *service*

Developer: FZJ and UNICORE SourceForge Community

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_cis

Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/CommonInformationService/>

Development language: Java

Build environment: Apache Maven

Download: http://sourceforge.net/project/showfiles.php?group_id=102081&package_id=269054

Licence: BSD License

Status

Maturity: pre-production

Usage: *D-Mon as part of D-Grid*

Interface maturity: production

Support status: *long-term*

Support team: UNICORE SourceForge Community
Development plans: Full GLUE2 support

Distribution/Availability

Middleware stack: UNICORE 6
Standalone usability: **Yes**
Component dependencies: *CIS Infoprotector deployed on UNICORE sites*
External software dependencies: *none*
Component consumers: *None*
Supported platforms: *All Java-based platforms*
Planned to be supported platforms: *None*

Service component information

Public Interfaces: CIS proprietary interface
Aligned security: *TLS/SSL*
Capabilities: *information.provenance*

Interoperability/Standards

Standard-compliance: *early-umd (GLUE2)*
Interoperable components: *None*
Additional Standards: *None*

Cost estimates

Maintenance/support effort: *0.25*
Standard convergence effort: *0.25*
New development effort: *0.5*
Total effort: *1*

4.3.11 UNICORE: Service Registry

Name UNICORE Service Registry

Description

Like many service-oriented environments, a global service registry is available, where the different services can register once they are started. A single service registry is necessary to build-up and operate a distributed UNICORE infrastructure. This service registry is contacted by the clients in order to "connect to the Grid". Therefore it is the central service of a UNICORE 6 Grid. Like the XNJS, the service registry runs in UNICORE's WS-RF hosting environment.

Basic information

Component type: *Web service*
Developer: UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_registry
Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/unicore/trunk/uas-core/>
Development language: *Java*
Build environment: *Apache Maven*
Download:
http://downloads.sourceforge.net/unicore/unicore-quickstart-6.1.3-p1.tgz?use_mirror=ovh
Licence: *BSD License*

Status

Maturity: *production*
Usage: *DEISA, D-Grid, SKIF-Grid, and commercial partners*
Available Documentation:
http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_registry (2009-03)

Interface maturity: production
Support status: *long-term and supported by UNICORE SourceForge Community*
Development plans: None

Distribution/Availability

Middleware stack: UNICORE 6
Standalone usability: **Yes**
Component dependencies: *UNICORE/X container*
External software dependencies: none
Component consumers: UNICORE clients, UNICORE atomic services
Supported platforms: *All Java-based platforms*
Planned to be supported platforms: *None*

Service component information

Public Interfaces: WS-RF ServiceGroups
Aligned security: *TLS/SSL*
Capabilities: *information.provenance*

Interoperability/Standards

Standard-compliance: *None*
Interoperable components: *none*
Additional Standards: WS-RF ServiceGroups

Cost estimates

Maintenance/support effort: *0.125*
Standard convergence effort:
New development effort:
Total effort: *0.125*

4.4 New development

5 Security functionality

5.1 Standards landscape

The standards landscape in terms of middleware security can be basically differentiated between authentication and authorization.

5.1.1 Authentication (AuthN)

Authentication of end-users refers to the check whether certificates of end-users have been signed by a trusted CA, are valid (date), and not revoked by Certificate Revocation List (CRL) mechanisms. In terms of authentication the middleware should support commonly used schemes that use certificate-based identity information from the transport layer security (TLS)/Secure Socket Layer(SSL) protocols. In this context, TLS connections can be initiated using full X.509 end-entity certificates [SEC1] or X.509 proxy certificates [SEC2].

We emphasize that UMD components have to either support one or the other! Because of modern handler designs it is possible to even combine different solutions without breaking the other correspondent other model in interoperability use cases.

5.1.2 Authorization (AuthZ)

Authorization can be differentiated in basically two approaches. The purely identity-based

authorization uses information of X.509 certificates to grant end-users access to the systems. However, in terms of UMD, we require a more flexible solution and raise the demand to support attribute-based authorization.

In this context, attributes indicate the position of an end-user within a virtual organization (VO) and group membership as well as role possession. These attributes are released from so-called Attribute Authorities (AA), which in terms of UMD are typically represented by either the Virtual Organization Membership Service (VOMS) or the UNICORE Virtual Organization Service (UVOS). To ensure the correctness of attributes a signing mechanism must be used and over time we have seen two approaches of signing and then conveying these attributes to middleware systems.

The first approach uses Attribute Certificates (AC) [SEC3] that are typically conveyed within the extension of X.509 certificates. This can be done with full end-entity X.509 certificates or X.509 proxies on the transport level (i.e. authentication). One AA that supports this approach is the non Web service-based 'classic VOMS' system.

The second approach uses Security Assertion Markup Language (SAML) [SEC4] assertions that are typically conveyed within the SOAP header during Web service calls (e.g. OGSA-BES systems). This can be also done with using either full end-entity X.509 certificates or X.509 proxies on the transport level (i.e. authentication). One AA that supports this approach is the Web service-based 'OMII-Europe/EGEE-III VOMS system' and UVOS. Although this approach seems to be limited in terms of using it only with Web service technologies there is a trend towards the use of this approach since it is much more flexible than the first approach. For instance, the second approach allows for the transport of n different SAML assertions maybe obtained from n different AAs. But most notably, the second approach encapsulates the transport level (i.e. TLS/SSL) from the transport of attributes (SOAP header instead of extension within a certificate used for TLS/SSL).

We emphasize that UMD components have to either support one or the other transport mechanism! Because of modern handler designs it is possible to even combine these different approaches without breaking the other correspondent other model in interoperability use cases. Also, many systems, like the 'classic VOMS' and 'OMII-Europe/EGEE-III VOMS' are based on the same VO database and thus release the same attributes in a different manner.

However, since at the end, only the attributes matter in terms of authorization decisions it is important that within UMD an agreement is achieved upon the security attributes and their semantics. The common UMD security semantics maybe based on Extensible Access Control Markup Language (XACML) standards [SEC5] policies or using rather proprietary mechanisms. Work should be focused on an agreement of the attribute semantics and their structure. This can be done in close collaboration with the OGF Production Grid Infrastructure Working Group that works on this topic.

An UMD security compliant component should be able to 1) validate users' credentials either in full X509 certificate or X509 proxy certificate format; 2) fetch attributes (roles) of the users either from well-defined VOMS extensions or from SAML assertions within the SOAP header of the messages. 3) be able to understand the semantics of the common UMD set of security attributes. As a result of the above three requirements UMD compliant components will be able to base their security decision (accept/reject the request) on the gathered information. In order to fulfill the requirements considerable amount of work is necessary in terms of both standardization and code development.

5.2 Overview table: Security area

	Name	Type	Maturity	Standards compliance		
--	------	------	----------	----------------------	--	--

5.3.1	ARC: HED security API	library	Pre-production			
5.3.2	ARC: Shib Bridge	service	development			
5.3.3	ARC: arcproxy	client	Pre-production			
5.3.4	ARC: Charon Authorization Service	service	development			
5.3.5	ARC: Proxy Store	service	development			
5.3.6	ARC: Fruitfly credential service	service	development			
5.3.7	gLite: VOMS	service	production	Non-umd		
5.3.8	gLite: VOMS-Admin	service	production	???		
5.3.9	gLite: voms-proxy-*	client	production	???		
5.3.10	gLite: Authorization Service	service	development	Non-umd		
5.3.11	gLite: Short Lived Credential Service	service	Pre-production	???		
5.3.12	gLite: SCAS	service	Pre-production	???		
5.3.13	gLite: LCAS	library	production	???		
5.3.14	gLite: LCMAPS	library	production	???		
5.3.15	gLite: gLite-Exec (glExec)	client	production	???		
5.3.16	gLite: gLite-Delegation Client	client	production	???		
5.3.17	gLite: Shibboleth Short Lived Credential Service Client	client	production	???		
5.3.18	gLite: Gridsite	library	production	???		
5.3.19	gLite: CGSI_gSoap	library	production	???		
5.3.20	gLite: delegation-Java	library	production	???		
5.3.21	gLite: Trustmanager	Library	production	???		
5.3.22	UNICORE: Gateway	service	production	Non-umd		
5.3.23	UNICORE: XUADB	service	production	None		
5.3.24	UNICORE: XACML Entity	other	production	None		
5.3.25	UNICORE: UVOS	service	production	Non-umd		
5.3.26	other: MyProxy	service	production	???		

5.3 Inventory of components

5.3.1 ARC: HED security API

Comment [BK6]: Fill it

Name

HED security API

Description

Libraries for service, client, etc.. developers (libarcsecurity, libarcdelegation, libarccredential, libxmlsec, libws-security)

Basic information

Component type: library

Developer:

Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: pre-production
Usage:
Interface maturity: {development, pre-production, production}
Support status: {abandoned, best effort, short term, long term}
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Library component information *(only for a library component)*

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}
Interoperable components:
Additional standards:

Cost estimates

Maintenance/support effort: 0.25
Standard convergence effort: 0.5
New development effort: 0.25
Total effort:1

Other

Any other kind of information

5.3.2 ARC: Shibbridge

Name Shibbridge

Description

A short paragraph describing the component

Basic information

Component type: {service, library, client}
Developer:
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: {planned, development, pre-production, production}
Usage:
Interface maturity: {development, pre-production, production}
Support status: {abandoned, best effort, short term, long term}
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Library component information *(only for a library component)*

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}
Interoperable components:
Additional standards:

Cost estimates

Maintenance/support effort: 0.1
Standard convergence effort: 0.1
New development effort: 0.25
Total effort: 0.45

Other

Any other kind of information

5.3.3 ARC: arcproxy

Name arcproxy

Description

A short paragraph describing the component

Basic information

Component type: client
Developer:
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence:

Status

Maturity: pre-production
Usage:
Interface maturity: {development, pre-production, production}
Support status: {abandoned, best effort, short term, long term}
Support team:
Development plans:

Distribution/Availability

Middleware stack:
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms:
Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:
Aligned security:
Capabilities:

Library component information *(only for a library component)*

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}
Interoperable components:
Additional standards:

Cost estimates

Maintenance/support effort: 0.1
Standard convergence effort: 0
New development effort: 0
Total effort: 0.1

Other

Any other kind of information

5.3.4 ARC: Charon Authorization Service

Name Charon Authorization Service

Description

A short paragraph describing the component

Basic information

Component type: service

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: {planned, development, pre-production, production}

Usage:

Interface maturity: {development, pre-production, production}

Support status: {abandoned, best effort, short term, long term}

Support team:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:

Aligned security:

Capabilities:

Library component information *(only for a library component)*

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}

Interoperable components:

Additional standards:

Cost estimates

Maintenance/support effort: 0.1

Standard convergence effort: 0.1

New development effort: 0.25

Total effort: 0.45

Other

Any other kind of information

5.3.5 ARC: Proxy Store

Name Proxy Store

Description

A short paragraph describing the component

Basic information

Component type: service

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: {planned, development, pre-production, production}

Usage:

Interface maturity: {development, pre-production, production}

Support status: {abandoned, best effort, short term, long term}

Support team:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:

Aligned security:

Capabilities:

Library component information *(only for a library component)*

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}

Interoperable components:

Additional standards:

Cost estimates

Maintenance/support effort: 0.1

Standard convergence effort: 0

New development effort: 0.25

Total effort: 0.35

Other

Any other kind of information

5.3.6 ARC: Fruitfly credential service

Name ARC: Fruitfly credential service

Description

Short Lived Credential Service *A short paragraph describing the component*

Basic information

Component type: service with simple client

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity: {planned, development, pre-production, production}

Usage:

Interface maturity: {development, pre-production, production}

Support status: {abandoned, best effort, short term, long term}

Support team:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:

Aligned security:

Capabilities:

Library component information *(only for a library component)*

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}

Interoperable components:

Additional standards:

Cost estimates

Maintenance/support effort: 0.1

Standard convergence effort: 0.2

New development effort: 0

Total effort: 0.3

Other

Any other kind of information

5.3.7 gLite: VOMS

Name gLite Virtual Organization Membership Service (VOMS)

Description

VOMS is an Attribute Authority service that issues attributes (in the form of signed assertions) expressing membership information of a subject within a VO. VOMS allows VO users to be partitioned in groups and to assign them roles and other free-form attributes that are then used to drive authorization decisions.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

VOMS Communication Protocol: <https://twiki.cern.ch/twiki/view/VOMS/VOMSProtocol>,

<https://twiki.cern.ch/twiki/view/VOMS/WebHome?>

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language: C++

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage: all EGEE installations, plus other major Grid infrastructures worldwide

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability: Yes

Component dependencies:

External software dependencies:

Component consumers: voms-proxy-* command line tools, gLite VOMS-admin, gLite WMS, gLite WMS UI, gLite CREAM among others

Supported platforms: SLC4/x86, other Linux distributions

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces: Legacy, SAML

Aligned security:

Capabilities:

Library component information

Consumers:

Language binding: C/C++/Java

Interoperability/Standards

Standard-compliance: non-umd (SAML)

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.8 gLite: VOMS-Admin

Name VOMS-Admin

Description

VOMS-Admin is the administrative application that manages and populates the VOMS database with VO-specific membership information. It is implemented as a Web Service that can be accessed either through a web interface or using a command line client.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation: <https://twiki.cfnf.infn.it/cgi-bin/twiki/view/VOMS/WebHome?>

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability: No

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.9 gLite: voms-proxy-*

Name gLite voms-proxy-*

Description

Commands to generate and manipulate X509 proxy certificates containing Attribute Certificates ACs by contacting a VOMS server.

Basic information

Component type: Client
Developer: EGEE3 JRA1
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence: Apache License

Status

Maturity: Production
Usage:
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability: No
Component dependencies: gLite VOMS
External software dependencies:
Component consumers:
Supported platforms: SLC4/i386
Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.10 gLite: Authorization Service

Name gLite Authorization Service

Description

The goal of the Authorization Service is to provide a consistent means of distributing, evaluating, and enforcing authorization policies across grid services. Policies, which are stored and transported as XACML documents, may be authored by various parties (e.g. VO administrators, site administrators) and combined to form an effective policy. It offers a consistent approach to authorization so that all components employ the same code, offer the same functionality and evaluate policies in the same manner. The Authorization Service is composed of four components: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Administration Point (PAP), and Execution Environment Service (EES).

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence: Apache License

Status

Maturity: Development

Usage:

Interface maturity: Development

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: none at the moment (the component is in development)

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance: non-umd (XACML)

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.11 gLite: Short Lived Credential Service

Name Short Lived Credential Service (SLCS)

Description

The Short Lived Credential Service (SLCS) is a service that issues short lived X.509 credentials based upon successful authentication at a Shibboleth Identity Provider. This service presents from the user's point of view an X.509 certificate factory that issues him/her a certificate with which he/she can access grid resources. The certificate is accepted by the international Grid Policy Management Authorities (IGTF), and therefore the SLCS certificates can be used on most existing Grid infrastructures. The SLCS server relies on Shibboleth to do the user authorization. It uses the Shibboleth attributes of the user to generate the required X.509 subject DN and extensions. An external online certificate authority server is used as a backend to sign the certificate signing requests sent by the client. The server also enforces the policies defined in the SLCS CP/CPS.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence: Apache License

Status

Maturity: Pre-Production

Usage:

Interface maturity: Pre-Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:
Total effort:

5.3.12 gLite: SCAS

Name gLite Site Central Authorization Service (SCAS)

Description

A centralized Local Centre Authorization Service (LCAS) implemented as a server front-end. This will allow LCAS to be invoked by any node other service in a site, providing centralized site-wide authorization policies.

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence: Apache License

Status

Maturity: Pre-Production

Usage:

Interface maturity: Pre-Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.13 gLite: LCAS

Name gLite Local Centre for Authorization Service (LCAS)

Description

The Local Centre Authorization Service (LCAS) is a framework that makes a binary authorization decision based on the credentials provided by the Grid user and is dynamically loaded by services for each incoming request. It provides authorization decisions in a fast way before getting to the mapping stage (LCMAPS). The LCAS framework does not need to communicate with any other service.

Basic information

Component type: Library

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment: ETICS

Download:

Licence: Apache License

Status

Maturity: Production

Usage: EGEE installations

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers: gLexec

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.14 gLite: LCMAPS

Name gLite Local Credential MAPPING Service (LCMAPS)

Description

The Local Credential MAPPING Service (LCMAPS) is a framework implemented as a library which is loaded dynamically by a service needing to map the Grid credentials presented by a client into UNIX UID and GIDs. The LCMAPS framework also executes some sub-tasks in order to optimize the execution of these evaluations by providing internally the certificate chain of the users, the JDL, a list of VOMS data and a placeholder for storing data such as the gathered UNIX UID and GIDs. This information can be used by the plugins directly from LCMAPS in order to reduce calls to the VOMS API and/or the GridSite core library.

Basic information

Component type: Library

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence: Apache License

Status

Maturity: Production

Usage: EGEE installations

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers: gLexec

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.15 gLite: gLite-Exec (glExec)

Name gLite-Exec (glExec)

Description

glExec (gLite Exec) is a component that switches the identity of Grid user processes based on the Fully Qualified Attributes presented by the user credential (usually a VOMS proxy). It was initially designed to operate on a Computing Element to run the specific LRMS command with appropriate rights. Lately its scope has been extended to operate also on a Worker Node, to enable jobs to run other processes under different identities, with the purpose of safely supporting job management frameworks based on pilot jobs. The development of glExec has been forked from gsexec, which in turn was forked from the suexec program from the Apache project. This development decision was made since the predecessor programs have a proven history to be safe.

The glExec component operates with strong security checking in mind, in that a valid user proxy certificate chain must be presented for a mapping to take place. glExec itself cannot determine the mapping from Grid user to uid and gid itself and relies on the LCAS/LCMAPS system to do this.

In order to do this, LCAS/LCMAPS needs to have a valid user proxy containing the full certificate chain and the private key generated in the last delegation step to the machine running glExec. glExec checks the ownership and file permissions of this certificate before it calls out to LCAS/LCMAPS. If glExec determines that the conditions are incorrect, it will halt, stopping the whole identity switching process. Since changing uid and gid usually requires root privileges, glExec comes as a binary executable program with the suid bit set.

Basic information

Component type: Client

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment: ETICS

Download:

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability: **Yes**

Component dependencies: LCAS, LCMAPS

External software dependencies:

Component consumers: BLAH, gLite CREAM and others

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.16 gLite: gLite-Delegation Client

Name gLite-Delegation Client

Description

This component also provides command-line utilities to manage (delegate, check and destroy) credentials on a service implementing the GridSite delegation protocol. The delegation client component depends on Service Discovery, gridsite-core and CGSI_gSOAP.

Basic information

Component type: Client
Developer: EGEE3 JRA1

Documentation:
Code repository:
Development language:
Build environment: ETICS
Download:
Licence: Apache License

Status

Maturity: Production
Usage: EGEE installations
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers: gLite CREAM, gLite WMPProxy
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.17 gLite: Shibboleth Short Lived Credential Service Client

Name Shibboleth Short Lived Credential Service Client

Description

Two shell commands are available as part of the SLCS client package. The main command is slcs-init, which authenticates the user with the given IdP, generates locally a private key and certificate signing request, then receives the resulting short lived X.509 certificate from the SLCS server. The information command slcs-info gives details about the configured Shibboleth identity providers.

Basic information

Component type: Client
Developer: EGEE3 JRA1
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence: Apache License

Status

Maturity: Production
Usage:
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability:
Component dependencies:
External software dependencies:
Component consumers:
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.18 gLite: Gridsite

Name gLite Gridsite

Description

GridSite delegation is a proxy certificate delegation protocol, based on standard Web Service interfaces. The development of this tool was fostered by the need of replacing the GSI-specific HTTPG protocol by the standard HTTPS one, in order that off-the-shelf components (such as client libraries and service containers) could be used more easily for Grid development. The simple Web Services-based interface is adjusted to match the functionality of the WSRF-based Globus delegation protocol, thus making the transition between these systems easier for developers. This package also provides the SQL schema definitions for MySQL and Oracle implementations, which serve as an interface between the delegation service and other services using the delegated credentials. It also provides a programming language independent documentation of the file and database based back-ends for service implementers who wish to use the delegated credentials.

Basic information

Component type: library
Developer: EGEE3 JRA1
Documentation:
Code repository:
Development language:
Build environment:
Download:
Licence: Apache License

Status

Maturity: Production
Usage:
Interface maturity: Production
Support status: short term (until the end of the EGEE3 project)
Support team: EGEE3/JRA1 team
Development plans:

Distribution/Availability

Middleware stack: gLite 3.1
Standalone usability:
Component dependencies:
External software dependencies:
Component dependencies:
Supported platforms: SLC4/x86
Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

5.3.19 gLite: CGSI_gSoap

Name gLite CGSI_gSoap

Description

CGSI-gSOAP is a library which allows gSOAP clients and services to use the Globus Security Infrastructure (GSI) authentication. By making minimal modifications to the C or C++ source code, a developer can enable the usage of HTTPS and HTTPG protocols via the GSI libraries. The server-side version of the CGSI-gSOAP library also aids the usage of VOMS credentials with the help of the VOMS library.

Basic information

Component type: Library

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.20 gLite: delegation-Java

Name gLite delegation-Java

Description

The delegation-java component is a server-side Java library that implements a GridSite delegation service. It provides implementation for both the file and database based back-ends. The delegation-java library can be used by service developers who wish to add proxy delegation functionality to their Java-based web service. The delegation-service-java component is a packaging of the delegation-java library into a Java Web Service, which can be deployed in a J2EE container such as Tomcat. It is possible to use a database back-end. The delegation-service-java communicates via the HTTPS protocol with its clients. The delegation-java component depends on Util-java and Trustmanager. The delegation-service-java component depends on Tomcat.

Basic information

Component type: Library

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language: Java

Build environment:

Download:

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.21 gLite: Trustmanager

Name gLite Trustmanager

Description

The Trustmanager is a Java library to check the validity of an X509 certificate chain and is interfaced with the Java extensible SSL engine. It contains the machinery to configure the Java SSL engine, to read the certificates in the format used in Grids, to support Certificate Revocation Lists (CRL) and to automatically reload at runtime the CRLs or the user credentials when they change. The package also contains plugins for integration with Tomcat web server and Axis web service clients. Trustmanager uses the bouncycastle Java cryptographic routines and depends on the presence of a servlet container.

Basic information

Component type: Library

Developer: EGEE3 JRA1

Documentation:

Code repository:

Development language: Java

Build environment:

Download:

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project)

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack: gLite 3.1

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers: gLite CREAM

Supported platforms: SLC4/x86

Planned to be supported platforms: SLC5/x86_64

Library component information

Consumers:

Language binding:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

5.3.22 UNICORE: Gateway

Name UNICORE Gateway

Description

The Gateway component acts as the entry point to a UNICORE site and performs the authentication of all incoming requests. From the UNICORE perspective it is the "door to the outside world" in a site firewall and may serve several resources/target systems behind it. In more detail the UNICORE Gateway checks whether a certificate is signed by a trusted CA and valid (date check) as well as not revoked using Certificate Revocation List (CRL) principles. The UNICORE Gateway performs purely identity-based authentication using the certificate information from the transport level connection (i.e. TLS/SSL), optionally accepting Globus proxy certificates. It provides basic HTTP proxy capabilities, i.e. it will forward HTTP GET, PUT and POST requests to the target sites behind it.

Basic information

Component type: *service*

Developer: UNICORE SourceForge Community

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_gateway

Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/gateway/>

Development language: Java

Build environment: Apache Maven

Download:

<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>

Licence: BSD License

Status

Maturity: *production*

Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*

Interface maturity: *production*

Support status: *long-term*

Support team: supported by UNICORE SourceForge Community

Development plans: Full HTTP support

Distribution/Availability

Middleware stack: UNICORE Core Server Bundle Version 6.2

Standalone usability: *yes*

Component dependencies: *None*

External software dependencies: *None*

Component consumers: *None*

Supported platforms: *All Java-based platforms*

Planned to be supported platforms: *None*

Service component information

Public Interfaces: *purely SSL (i.e. https)*

Aligned security: *TLS/SSL (optional proxy support)*

Capabilities: *security.authentication*

Interoperability/Standards

Standard-compliance: *non-umd (TLS/SSL, proxy certificates)*

Interoperable components: *None*

Additional Standards: *HTTP*

Cost estimates

Maintenance/support effort: *0.25*

Standard convergence effort:
New development effort: 0.25
Total effort: 0.5

5.3.23 UNICORE: XUADB

Name UNICORE enhanced UNICORE User Database (XUADB)

Description

For mainly identity-based authorization of users the XNJS uses the XUADB user database to perform the mapping from X.509 certificates to the actual users' logins and roles. The XUADB component is a Web service in itself, so it can be used from multiple UNICORE installations, e.g. within the same computing centre. Full X.509 certificates are used as base line, while the access control is based on XACML policies. Proxy certificates are optionally supported in UNICORE 6 e.g. to use GridFTP.

Basic information

Component type: *service*

Developer: UNICORE SourceForge Community

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_xuadb

Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/xuadb/>

Development language: Java

Build environment: Apache Maven

Download:

<http://downloads.sourceforge.net/unicore/unicore-servers-6.2.0-p1.tgz>

Licence: BSD License

Status

Maturity: *production*

Usage: *DEISA, D-Grid, SKIF-Grid, commercial partners*

Interface maturity: *production*

Support status: *long-term*

Support team: supported by UNICORE SourceForge Community

Development plans: *None*

Distribution/Availability

Middleware stack: UNICORE 6.2

Standalone usability: *yes*

Component dependencies: *None*

External software dependencies: *None*

Component consumers: UNICORE Atomic Services, UNICORE OGSA-BES

Supported platforms: *All Java-based platforms*

Planned to be supported platforms: *None*

Service component information

Public Interfaces: proprietary XUADB WS interface

Aligned security: *TLS/SSL and optional proxies, identity-based authorization PIP*

Capabilities: *security.authorization; security.identitymapping*

Interoperability/Standards

Standard-compliance: *None*

Interoperable components: *None*

Additional Standards: **WS**

Cost estimates

Maintenance/support effort: 0.125
Standard convergence effort: n/a
New development effort: 0.125
Total effort: 0.25

5.3.24 UNICORE: XACML Entity

Name UNICORE Extensible Access Control Markup Language (XACML) Entity

Description

The XACML Entity in UNICORE performs authorization decisions based on XACML policy files. This is used as an PDP in UNICORE that allows for identity-based authorization in conjunction with the XUADB PIP and attribute-based authorization decisions if, and only if, attributes have been used in conjunction with UNICORE (e.g. using UVOS).

Although this component is not directly a full UNICORE component it is essential in understanding the architecture of UNICORE with respect to security.

Basic information

Component type: **other (non Web service functionality)**

Developer: UNICORE SourceForge Community

Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_xuadb

Code repository: <http://unicore.svn.sourceforge.net/viewvc/unicore/xnjs/>

Development language: Java

Build environment: Apache Maven

Download:

http://downloads.sourceforge.net/unicore/unicore-quickstart-6.1.3-p1.tgz?use_mirror=fastbull

Licence: BSD License

Status

Maturity: production

Usage: DEISA, D-Grid, SKIF-Grid, commercial partners

Interface maturity: production

Support status: long-term

Support team: UNICORE SourceForge Community

Development plans: None

Distribution/Availability

Middleware stack: UNICORE Core Server Bundle Version 6.1.3

Standalone usability: No

Component dependencies: None

Component consumers: UNICORE Atomic Services, UNICORE OGSA-BES

Supported platforms: *All Java-based platforms*

Planned to be supported platforms: *None*

Service component information

Public Interfaces: None

Aligned security: certificate-based identity information and security attributes are used to perform authorization decisions

Capabilities: security.authorization

Library component information

Consumers: None

Language binding: None

Interoperability/Standards

Standard-compliance: None
Interoperable components: None
Additional Standards: XACML

Cost estimates

Maintenance/support effort: *0.08*
Standard convergence effort: *0*
New development effort: *0.08*
Total effort: *0.16*

5.3.25 UNICORE: UVOS

Name UNICORE Virtual Organisation Service

Description

As an alternative or as an addition to the XUADB UVOS, this Virtual Organisation (VO) Service using the SAML standard can be used to authorize users. It acts as an attribute-authority (AA) and releases signed attributes for end-users stating the position of an end-user within a VO as well as group membership or role possession. Released attributes are later used in conjunction with the UNICORE XACML entity to perform attribute-based authorization decisions.

Basic information

Component type: service
Developer: UNICORE SourceForge Community
Documentation: http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_uvos
Code repository: <http://unicore.svn.sourceforge.net/svnroot/unicore/uvos> (currently being migrated)
Development language: Java
Build environment: Apache Maven
Download:
<http://downloads.sourceforge.net/unicore/VOManager-1.3.2-linux.gtk.x86.tar.gz>
Licence: BSD License

Status

Maturity: production
Usage: Chemomentum project
Interface maturity: production
Support status: long term
Support team: ICM Warsaw and UNICORE SourceForge Community
Development plans:

Distribution/Availability

Middleware stack: UNICORE VO Service 1.3.2
Standalone usability: yes
Component dependencies: *None*
External software dependencies: None
Component consumers: UNICORE atomic services, UNICORE OGSA-BES
Supported platforms: All Java-based platforms
Planned to be supported platforms: None

Service component information

Public Interfaces: SAML
Aligned security: Username/Password, TLS/SSL, attributes of end-users conveyed via SAML assertions

Capabilities: security.attributeauthority

Interoperability/Standards

Standard-compliance: non-umd (SAML)

Interoperable components: *None*

Additional Standards: None

Cost estimates

Maintenance/support effort: 0.25

Standard convergence effort: 0.25

New development effort: 0.25

Total effort: 0.75

5.3.26 other: MyProxy

Name MyProxy

Description

MyProxy is used for managing X.509 Public Key Infrastructure (PKI) security credentials (certificates and private keys). Within EGEE this service is used to overcome the problem of short-lived proxies and long jobs. It allows the user to create and store a long-term proxy in a dedicated server (a MyProxy server). The WMS, CREAM CE, VOBOX and FTS will then be able to use this long-term proxy to periodically renew the proxy for a submitted job before it expires and until the job ends (or the long-term proxy expires).

Basic information

Component type: Service

Developer: EGEE3 JRA1

Documentation:

Code repository: gLite code repository <http://jra1mw.cvs.cern.ch:8180/>

Development language:

Build environment: ETICS

Download: <http://glite.web.cern.ch/glite/packages/R3.1/>

Licence: Apache License

Status

Maturity: Production

Usage:

Interface maturity: Production

Support status: short term (until the end of the EGEE3 project),

Support team: EGEE3/JRA1 team

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information *(only for a service component)*

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance: {none, non-UMD, early-UMD, full-UMD}

Interoperable components:

Additional standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

Other

Any other kind of information

5.4 New development

6 Accounting functionality

6.1 Standards landscape

There are currently two OGF specifications (still in their draft stage) dealing with representation and manipulation of accounting information. These specifications are the Usage Record (UR) [5] and Resource Usage Service (RUS) [6]. The Usage Record specification describes an XML notation for describing resource usage information, while the Resource Usage Service specification describes the Web Service interface for a standards-compliant accounting system, which enables grid resource usage auditing and accounting as well as grid economic model. The RUS service is needed to provide interface for UR insertion to accounting database and to provide query interface for information about resource usage to a variety of Grid entities: service manager who wishes to examine utilization across their resources; a service that wishes to charge for the use of the consumed resources; and a virtual organization that wishes to monitor resource activity within their Grid. Usage information is made available in UR format.

The main limitations of the RUS/UR specifications, which make them unsuitable for production use, are their immature status and the fact that the RUS interface does not allow aggregate resource usage information to be accessed. This means that, for example, it is not possible to get the average, per-VO utilization of a given computational resource over a given time span. Only individual accounting records can be provided and the client must then aggregate the data. This is highly inefficient, as the RUS service must provide large amount of information to clients, with serious performance problems.

Unfortunately, neither RUS nor the UR in its current form is ready for production usage. Currently the RUS specification is being improved by the OGF RUS-WG. Either a new version will be available in a reasonable timescale (~1 year) so that it can be evaluated and eventually implemented in production Grids or the existing RUS specification needs to be heavily profiled to remove its limitations and be used in production.

The currently available accounting-related standards do not provide useful recommendation for public accounting interfaces and associated data exchanged, nor is such a common standard expected to emerge in a reasonable timescale. However, given that accounting is usually very middleware-specific (and often also VO-specific), it is acceptable that each accounting system will maintain a different interface as long as these interfaces are open and properly published as well as well documented. Thus, within UMD it is suggested that accounting components should come

with properly documented open interfaces.

6.2 Overview table: Accounting area

	Name	Type	Maturity	Standard compliance		
6.3.1	ARC: JURA	client	development	Non-umd		
6.3.2	gLite: Accounting Service	service	???	None (?)		
6.3.3	gLite: APEL	client (?)	???	None (?)		
6.3.4	other: SGAS	service	production	Non-umd		

6.3 Inventory of components

6.3.1 ARC: JURA

Name Job Usage Reporter of ARC (JURA)

Description

The Job Usage Reporter of ARC (JURA) is an agent for reporting resource usage of jobs to accounting services. Its input consists of job log files created by the execution manager from metering data provided by the LRMS. These job logs are parsed, transformed into standard Usage Record format, and then submitted to accounting services for persistent recording. Currently ARC Job Usage Reporter interacts with the SGAS Logging and Usage Tracking Service.

Basic information

Component type: client

Developer: Lund University & NIIF of NorduGrid

Documentation:

<http://svn.nordugrid.org/repos/nordugrid/arc1/trunk/doc/jura/jura-tech-draft.pdf>

Code repository:

<http://svn.nordugrid.org/repos/nordugrid/arc1/trunk/src/clients/jura>

Development language: c++

Build environment: autotools

Download: download.nordugrid.org

Licence: Apache Licence version 2.0

Status

Maturity: development

Usage: Hungarian Clustergrid, Nordic Grid deployments

Interface maturity: production

Support status: short term provided by the KnowARC project

Support team: NIIF

Development plans: add support for other accounting services such as the gLite Accounting Service once their interfaces are well-defined and documented.

Distribution/Availability

Middleware stack: will be part of the next technology preview release of ARC (0.9.4)

Standalone usability: none

Component dependencies: ARC A-REX, ARC HED

External software dependencies: libxml, Glibm, openssl

Component consumers: ARC A-REX, SGAS

Supported platforms: all flavors of linuxes

Planned to be supported platforms: windows, mac-osx

Interoperability/Standards

Standard-compliance: non-UMD

Interoperable components: none

Additional standards: makes use of OGF UR to report to SGAS

Cost estimates

Maintenance/support effort: 0.1

Standard convergence effort: 0.15

New development effort: 0.5

Total effort: 0.75

Other

The JURA component is an important module undergoing rapid development and expected to be deployed in production as a replacement of the currently used SGAS JARM module already around the 3rd quarter of this year.

6.3.2 gLite: Accounting Service

Name gLite Accounting Service

Description

The accounting information is sent to a central accounting database where it is processed to generate statistical summaries that are available through the EGEE/WLCG Accounting Portal. These statistics are then available for use, in different views, by: all scientific communities using the named Grid Infrastructures; the owners and administrators of the constituent resources, and the management of the Grid Infrastructures. For example, they include the total CPU consumed by each VO.

Accounting portal: http://www3.egee.cesga.es/gridsite/accounting/CESGA/egee_view.php

Basic information

Component type: service

Developer:

Documentation:

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity:

Usage:

Interface maturity:

Support status:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information

Public Interfaces:

Aligned security:

Capabilities:

Interoperability/Standards

Standard-compliance:

Interoperable components:

Additional Standards:

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort:

6.3.3 gLite: APEL

Name APEL (Accounting Processor for Event Logs)

Description

APEL is a component of EGEE CPU accounting distributed as part of the gLite middleware. APEL parses batch system and Grid gatekeeper logs at each Grid site to generate CPU usage records and publishes them into a centralised repository at a GOC (Grid Operations Centre).

Basic information

Component type:

Developer:

Documentation: <https://edms.cern.ch/document/726137>

Code repository:

Development language:

Build environment:

Download:

Licence:

Status

Maturity:

Usage:

Available Documentation:

Interface maturity:

Support status:

Development plans:

Distribution/Availability

Middleware stack:

Standalone usability:

Component dependencies:

External software dependencies:

Component consumers:

Supported platforms:

Planned to be supported platforms:

Service component information

Public Interfaces:
Aligned security:
Capabilities:

Library component information

Consumers:
Language binding:

Interoperability/Standards

Standard-compliance:
Interoperable components:
Additional Standards:

Cost estimates

Maintenance/support effort:
Standard convergence effort:
New development effort:
Total effort:

6.3.4 other: SGAS

Name SweGrid Accounting System (SGAS)

Description

The SweGrid Accounting System (SGAS) is a Java implementation of a usage tracking and credit allocation enforcement system for the Grid. Services included are the Logging and Usage Tracking Service (LUTS) for recording usage data in an XML database after execution, and the Bank service for usage credit management. For these services, client applications called Job Account Reservation Manager (JARM) components are available for the computing element of an old version of the production ARC middleware and for Globus GRAM.

Basic information

Component type: collection of services, corresponding clients and management utilities
Developer: Umea University, NDGF
Documentation: <http://www.cs.umu.se/research/grid/sgas/>
Code repository: <svn://svn.cs.umu.se/proj/gird/sgas/trunk>
Development language: java
Build environment: recently changed to Apache buildr from Maven1
Download: <http://www.cs.umu.se/research/grid/sgas/releases/sgas-2.2.0.tar.gz>
Licence: Apache License Version 2.0

Status

Maturity: production
Usage: NDGF sites
Interface maturity: production
Support status: long term
Support team: Umea University
Development plans: replacement of the database backend, interface cleanup and optional standardization, add support for aggregations

Distribution/Availability

Middleware stack: SGAS distribution version 2.0
Standalone usability: yes
Component dependencies: none
External software dependencies: java, globus java-ws-core 4.0.8

Component consumers: ARC Usage Reporter
Supported platforms: linuxes
Planned to be supported platforms: none

Service component information *(only for a service component)*

Public Interfaces: WSRF Resource Properties-based custom interface to add/query usage records

Aligned security: TLS/SSL proxies are supported, VOMS attributes are not understood, file-based and XACML-based access control policies within the servers

Capabilities: security.accounting

Interoperability/Standards

Standard-compliance: non-UMD

Interoperable components:

Additional standards: incompletely supports an early version of OGF Usage Record

Cost estimates

Maintenance/support effort:

Standard convergence effort:

New development effort:

Total effort: 1.5

Other

SGAS is currently undergoing a major redesign and reimplementation.

6.4 New development

To be identified and added later.

7 Other functionality

7.1 Overview table: Other components

	Name	Type	Maturity	Standard conformance		
7.2.1	ARC: HED	library	Pre-production	Non-umd		

7.2 Inventory of components

7.2.1 ARC: HED

Name ARC Hosting Environment (Daemon) Framework (HED)

Description

The ARC HED is a collection of libraries and plugins for building lightweight client/service applications. It is XML based and has minimalistic SOAP support. One of the most important feature of HED is it's extremely flexible architecture due to pluggable functionality. Current implementation of HED comes with modules for developing SOAP and Web Services. It serves as base infrastructure for all ARC components being developed.

Basic information

Component type: library (accompanied by plugins and a hosting executable)

Developer: NIIF, Oslo University from NorduGrid
Documentation: http://www.nordugrid.org/documents/ARCHED_article.pdf
Code repository: <http://svn.nordugrid.org/trac/nordugrid/browser/arc1/trunk/src/hed>
Development language: C++
Build environment: GNU autotools
Download: download.nordugrid.org
Licence: Apache License Version 2.0

Status

Maturity: pre-production
Usage: HED is a central component being part of most ARC deployments either client or server-side
Available Documentation: http://www.nordugrid.org/documents/ARCHED_article.pdf
Interface maturity: production (current features stable, new features added in compatible way)
Support status: short term support by KnowARC project, long term support by NorduGrid collaboration
Support team: Oslo University
Development plans: continuously adding new plugins. implementing new features

Distribution/Availability

Middleware stack: technology preview releases of ARC (v 0.9.x) and to be part of the production ARC release starting from the coming v 0.8.0 scheduled for May 2009
Standalone usability: Yes
Component dependencies: none
External software dependencies: libxml, Glibm, openssl, plus several optional dependencies
Component consumers: all the newer ARC components
Supported platforms: Linux, MacOS, Solaris, Windows (preliminary)
Planned to be supported platforms: improve support for windows

Library component information

Consumers: recently developed ARC services and client tools
Language binding: C++, Python, Java (not tested)

Interoperability/Standards

Standard-compliance: non-umd (plugins for handling TCP, TLS, GSI, SOAP, WSRF selected functionality, WS-Security selected functionality, etc.
Interoperable components: None
Additional Standards: see above

Cost estimates

Maintenance/support effort: 1
Standard convergence effort: 1
New development effort: 0.5
Total effort: 2.5

Other

HED is scheduled to be included in the next production release of ARC (due May 2009) together with the first HED-based service, A-REX.

8 References

[1] Ali Anjomshoaa, Fred Brisard, Michel Drescher, Donal Fellows, An Ly, Stephen McGough, Darren Pulsipher, and Adreas Savva. Job Submission Description Language (JSDL) Specification, Version 1.0, November 1 2005. OGF Specification GFD-R.056,

<http://www.gridforum.org/documents/GFD.56.pdf>.

[2] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer. OGSA Basic Execution Service Version 1.0, August 2007. OGF Specification GFD.108, <http://www.ogf.org/documents/GFD.108.pdf>.

[3] Blair Dillaway, Marty Humphrey, Chris Smith, Marvin Theimer, and Glenn Wasson. HPC Basic Profile, Version 1.0, August 28 2007. OGF Specification GFD-R-P.114, <http://www.ogf.org/documents/GFD.114.pdf>.

[4] Marty Humphrey, Chris Smith, Marvin Theimer, Glenn Wasson, JSDL HPC Profile Application Extension, Version 1.0.

[5] Usage Record Working Group: <https://forge.gridforum.org/projects/ur-wg/>

[6] Resource Usage Service Working Group: <https://forge.gridforum.org/projects/rus-wg>

[SEC1] R. Housley, W. Ford, W. Polk, D.Solo: Internet X.509 Public Key Infrastructure (PKI) Certificate and CRL Profile, January 1999.
<http://www.ietf.org/rfc/rfc2459.txt>

[SEC2] S. Tuecke, V. Welch, D. Engert, L. Pearlman, M. Thompson: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, Internet Engineering Task Force, June 2004.
<http://www.ietf.org/rfc/rfc3820.txt>

[SEC3] S. Farrel, R. Housley: An Internet Attribute Certificate Profile for Authorization, Internet Engineering Task Force, April 2002. <http://www.ietf.org/rfc/rfc3281>

[SEC4] OASIS Security Services (SAML) Technical Committee,
<http://www.oasis-open.org/committees/security/>

[SEC5] OASIS XACML Technical Committee,
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml