GWD-R, GWD-I or GWD-C
GLUE-WG

Sergio Andreozzi, INFN
Felix Ehm, CERN
Laurence Field, CERN
Balázs Kónya, Lund University
February 19, 2008

GLUE Specification v. 2.0 (draft 22)

Status of This Document

This document provides information to the Grid community regarding the specification of the GLUE information model. Distribution is unlimited. This document is a draft.

Abstract

The GLUE specification is an information model for Grid entities described in natural language enriched with a graphical representation using UML Class Diagrams. As a conceptual model, this is meant to be implementation-independent. Mapping to concrete data models such as XML Schema, LDAP and relational are provided.

Sergio Andreozzi, INFN
Balázs Kónya, Lund University
Add final date

Contents

1. Introduction

In this document, we present a conceptual information model for Grid entities described in natural language enriched with a graphical representation using UML Class Diagrams. As a conceptual model, this is meant to be implementation-independent. Mapping to concrete data models such as XML Schema, LDAP, relational and RDF are provided in the Appendix. From the semantic viewpoint, the concrete data model should represent the same concepts and relationships of the conceptual information model; nevertheless it can contain simplifications specific to the target data model in order to improve query performance or other aspects.

Commento [SA1]: To be added

This information model is based on the experience of several modeling approaches being used in current production Grid infrastructures (e.g., GLUE Schema 1.x [glue-1.x], NorduGrid schema [ng-schema], Naregi model [naregi-schema]). The proposed initial collection of entities is motivated also by the use cases document [glue-usecases].

Commento [SA2]: To be extended

2. Notational Conventions

Only include this section if applicable.

The key words ‘MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC 2119 [BRADNER1]

3. General

The Information Model and its renderings have to be consider case-sensitive

4. Conceptual Model of the Main Entities

This section introduces the main entities of the GLUE information model. They captures the core concepts that relevant in a Grid environment. The main entities SHOULD be used to derive specialized information models. In Figure 1, the classes and the related relationships are presented.

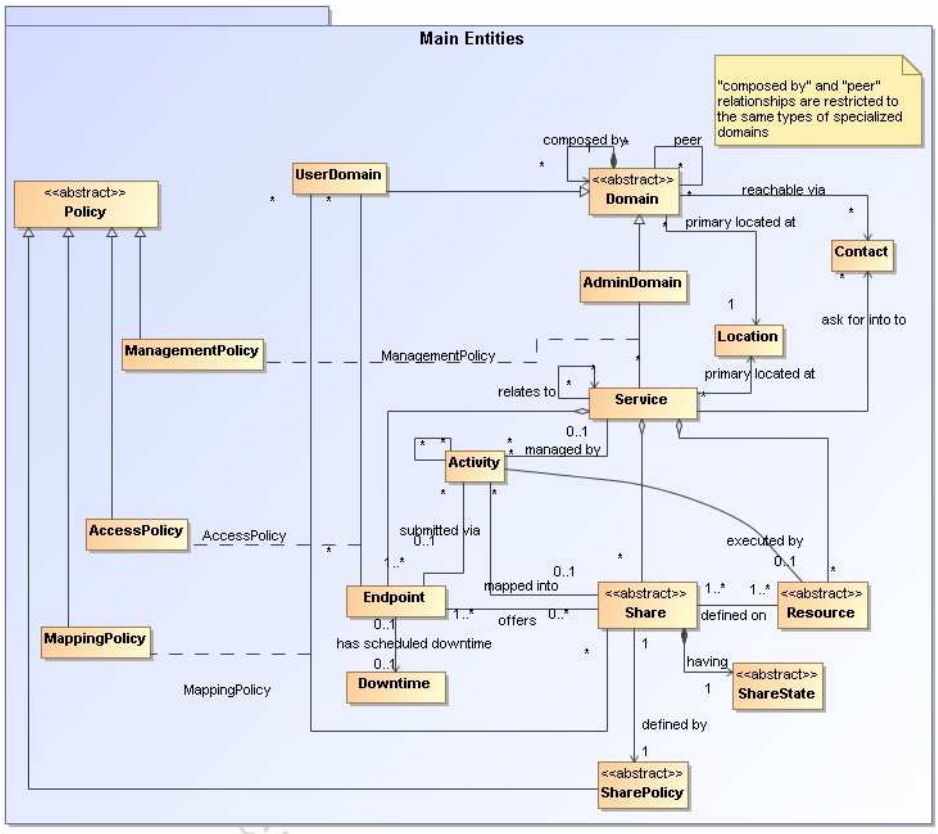


Figure 1 GLUE main entities and their relationships

4.1 Location

Entity	Inherits from			Description
Location				A geographical position
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	String	1		An opaque local identifier
Name	String	1		A human-readable name
Address	String	0..1		Street address
Place	String	0..1		Name of town/city
Country	String	0..1		Country name
PostCode	String	0..1		Postal code
Latitude	Real32	0..1	Degree	The position of a place north or south of the equator measured from -90° to +90° with positive values going north and negative values going south
Longitude	Real32	0..1	Degree	The position of a place east or west of Greenwich, England measured from -180° to +180° with positive values going east and negative values going west

The location entity is meant to be used for describing reference geographical positions of domains and services. They aim is to provide a simple way to express geographical information and is not intended to be used in complex geographical information systems. The accuracy of latitude and longitude should be defined in an interoperability profile.

4.2 Contact

Entity	Inherits from			Description
Contact				Information enabling to establish a communication with a person or group of persons part of a domain
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	String	1		An opaque local identifier
URL	URL	1		URL embedding the contact information. The syntax of URI depends on the communication channel
Type	ContactType_t	1		Type of contact
OtherInfo	String	*		Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax

Commento [SA3]: What about if an email address is used for usersupport and security? (multiple types or decoupling ID from contact info?)

This entity can be used to represent contact information for user support, security, sysadmin. The various types of contact are identified by the Type attribute. In case of time-depend contact information, the instances of this entity should represent only the active contact information.

For telephone and fax: http://www.ietf.org/rfc/rfc2806.txt
For email: http://www.ietf.org/rfc/rfc2368.txt
For irc: http://www.w3.org/Addressing/draft-mirashi-url-irc-01.txt

http://www.ietf.org/rfc/rfc2806.txt

4.3 Domain

Entity	Inherits from			Description
Domain				A collection of actors that can be assigned with roles and privileges to entities via policies. A domain may have relationships to other domains.
Property	**Type**	**Mult.**	**Unit**	**Description**
ID [key]	URI	1		A global unique ID
Name	String	0..1		Human-readable name
Description	String	0..1		A description of the domain
WWW	URL	*		The URL identifying a web page with more information about the domain
OtherInfo	String	*		Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax

Commento [SA4]: Add recommendation from Stephen Burke mentioned document

This is an abstract entity not meant to be instantiated. It SHOULD be used in order to derive specialized entities.

4.3.1 AdminDomain

Entity	Inherits from			Description
AdminDomain	Domain			A collection of actors that can be assigned with administrative roles and privileges to services via policies. An AdminDomain manages services that can be geographically distributed, nevertheless a primary location should be identified.
Inherited Property	**Type**	**Mult.**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
Description	*String*	*0..1*		*A description of the domain*
WWW	*URL*	*		*The URL identifying a web page with more information about the domain*
OtherInfo	*String*	*		*Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax*
Property	**Type**	**Mult.**	**Unit**	**Description**
Distributed	Boolean	0..1		True if the services managed by the admindomain are considered geographically distributed by the administrators themselves
Owner	String	*		Owner of the managed resources

Commento [SA5]: Add recommendation from Stephen Burke mentioned document

4.3.2 UserDomain

Entity	Inherits from			Description
UserDomain	Domain			A collection of actors that can be assigned with user roles and privileges to services or shares via policies
Inherited Property	**Type**	**Mult.**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
Description	*String*	*0..1*		*A description of the domain*
WWW	*URL*	*		*The URL identifying a web page with more information about the domain*
OtherInfo	*String*	*		*Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax*
Property	**Type**	**Mult.**	**Unit**	**Description**
Level	Int32	0..1		The number of hops to reach the root for hierarchically organized domains described by the “composed by” association (0 is for the root)
ManagerEndpoint	URI	*		The Endpoint ID managing the users part of the domain and the related attributes such as groups or roles

Commento [SA6]: Add recommendation from Stephen Burke mentioned document

In the GLUE Information Model, the Virtual Organization can be realized by using the concept of UserDomain. If the VO has an internal structure, this can be represented by using different domains related to each other. A Virtual Organization (VO) comprises a set of individuals and/or institutions having direct access to computers, software, data, and other resources for collaborative problem-solving or other purposes. Resources utilized by a VO are expected to be accessible via network endpoints and constrained by defining utilization targets called shares. The VO can exhibit the internal structure in terms of groups of individuals, each of them being a UserDomain. UserDomains can be hierarchically structured. This structure can be represented via the "composed by" association. A userDomain can be also related to other other userDomains via a "peer" relationship.

Commento [SA7]: We do no have use cases for instantiating the peer relationship; if we won't have, then we should remove it

As regards the ManagerEndpoint, a commonly used implementation is the VOMS.

Commento [SA8]: Add reference

4.4 Policy

Entity	Inherits from			Description
Policy				Statements, rules or assertions that specify the correct or expected behavior of an entity
Property	**Type**	**Mult.**	**Unit**	**Description**

This is an abstract entity not meant to be instantiated.

4.4.1 ManagementPolicy

Entity	Inherits from			Description
ManagementPolicy	Policy			Statements, rules or assertions that assign management capabilities to actors as regards a manageable entity
Property	**Type**	**Mult.**	**Unit**	**Description**

Commento [SA9]: Specify that this is added to have a consistent conceptual model; example implementation in LDAP/XML is parent-child relationship between AdminDomain and Service

The existence of relationship among an AdminDomain and a Service implies that an AdminDomain can manage a Service. Currently, there is no use cases for having attributes in this entity.

4.4.2 AccessPolicy

Entity	Inherits from			Description
AccessPolicy	Policy			Statements, rules or assertions that provide coarse-granularity information about the access by actors to an endpoint
Property	**Type**	**Mult.**	**Unit**	**Description**
Scheme	PolicyScheme_t	1		Scheme adopted to define the policy rules
Rule	String	*		A policy rule
TrustedCA	DN_t	*		Distinguished name of the trusted Certification Authority

Commento [SA10]: Add more clarification about why it is coarse-granular

Commento [SA11]: Add basic policy scheme with VO, VOMS FQAN, (ALLOW)/DENY

Commento [SA12]: Evaluate if trustedCA goes together with access control information

This entity can be used to express which UserDomains can access a certain service endpoint. The granularity of these policies should be coarse-grained and suitable for pre-selection of services. The actual decision on the service side is performed by an authorization component that can contain a finer-grained set of policy rules that in some case can contradict the published coarse-grained policy rules. Examples of actors involved in this entity are userDomains representing VOs or groups.

4.4.3 MappingPolicy

Entity	Inherits from	Description
MappingPolicy	Policy	Statements, rules or assertions that provide coarse-granularity information about the mapping of activities to a share

Property	Type	Mult.	Unit	Description
Scheme	PolicyScheme_t	1		Scheme adopted to define the policy rules
Rule	String	*		A policy rule
Default	Boolean	0..1		Default share to which the activity will be mapped if no preference are expressed by the user

Commento [SA13]: do we need this? Is it a special case of access policy or a different category?

Commento [SA14]: Add more clarification about why it is coarse-granular

Commento [SA15]: To be confirmed

This entity can be used to express which UserDomains can consume a certain share of resources.

4.5 Service

Entity	Inherits from	Description
Service		An abstracted, logical view of actual software components that participate in the creation of an entity providing one or more functionalities useful in a Grid environment. A service exposes one or more endpoints having well-defined interfaces, zero or more shares and zero or more resources. The service is autonomous and denotes a weak aggregation among endpoints, the exposed resources, and the defined shares. The service enables to identify the whole set of entities providing the functionality with a persistent name.

Property	Type	Mult.	Unit	Description
ID [key]	URI	1		A global unique ID
Name	String	0..1		Human-readable name
Capability	ServiceCapability_t	*		The capability provided by this service according to the OGSA architecture
Type	ServiceType_t	1		The type of service according to a middleware classification
QualityLevel	QualityLevel_t	1		Maturity of the service in terms of quality of the software components
StatusPage	URL	*		Web page providing additional information like monitoring aspects
Complexity	String	0..1		Human-readable summary description of the complexity in terms of the number of endpoint types, shares and resources. The syntax should be: endpointType=X, share=Y, resource=Z.
OtherInfo	String	*		Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax

The simplest Service is composed by one endpoint, no share and no resource (e.g. a metadata catalog service). In the context of a Service, the same resource part of it can be exposed via multiple endpoints based on defined shares. For instance, in the area of storage systems, SRMv1 and SRMv2.2 interfaces can expose the same resource via different endpoints offering different interface version; in the area of computing systems, the CREAM and GRAM endpoints can expose the same batch system. Endpoints, shares and resources can belong to only one service.

Commento [SA16]: To be verified by real-world use cases

4.6 Endpoint

Entity	Inherits from			Description
Endpoint				A network location having a well-defined interface and exposing the service functionalities
Property	**Type**	**Mult.**	**Unit**	
ID [key]	URI	1		A global unique ID
Name	String	0..1		Human-readable name
URL	URL	1		Network location of the endpoint to contact the related service
Capability	EndpointCapability_t	*		The capability exposed by this interface
Type	EndpointType_t	1		The type of endpoint according to a middleware classification
QualityLevel	QualityLevel_t	1		Maturity of the service in terms of quality of the software components
SpecificationName	String	0..1		Name of the interface specification
SpecificationVersion	String	0..1		Version of the interface
Implementor	String	0..1		Main organization implementing this software component
ImplementationName	String	0..1		Name of the implementation
ImplementationVersion	String	0..1		Version of the implementation (e.g., major version.minor version.pathcversion)
HealthState	EndpointHealthState_t	1		A state representing the health of the endpoint
HealthStateInfo	String	0..1		Textual explanation of the state endpoint
ServingState	ServingState_t	1		The serving state (production, draining, queueing, closed)
WSDL	URL	*		URL of the WSDL document describing the offered interface (applies to Web Services endpoint)
SupportedProfile	URI	*		URI identifying a supported profile
Semantics	URL	*		URL of a document providing a human-readable description of the semantics of the endpoint functionalities
StartTime	DateTime	0..1		The timestamp for the start time of the endpoint
IssuerCA	DN_t	0..1		Distinguished name of Certification Authority issuing the certificate for the endpoint
Association End		**Mult.**		**Description**
Association to UserDomain via Access Policy				

Commento [SA17]: To be verified if we keep both here and in service or only in one part

Commento [SA18]: Suggesti on to use URI for identifying categories; Donal will provide examples

Commento [SA19]: What is the relationship between values for this attribute and values for the service.qualityLevel?

Commento [SA20]: Verify if a single value is enough

Commento [SA21]: to be extended, should capture what is currently called AccessControlBaseRule in GLUE 1.x

For Grid services requiring a richer set of properties for the endpoint, specific models can be derived by specializing from the Endpoint entity and adding new properties or relationships. The current proposal contains the ComputingEndpoint specialization (see Section)

Commento [SA22]: add section reference

4.7 Downtime

Entity	Inherits from			Description
Downtime				A description of a scheduled downtime event
Property	**Type**	**Mult.**	**Unit**	
DowntimeAnnounce	DateTime	0..1		The timestamp for the announcement of the next scheduled downtime
DowntimeStart	DateTime	1		The starting timestamp of the next scheduled downtime
DowntimeEnd	DateTime	0..1		The ending timestamp of the next scheduled downtime
DowntimeInfo	String	0..1		Description of the next scheduled downtime

4.8 Share

Entity	Inherits from	Description
Share		A utilization target for a set of resources offered via related endpoints defined by policies and characterized by status information

Property		Type	Mult.	Unit	Description
LocalID	[key]	String	1		An opaque local identifier
Name		String	0..1		Human-readable name
Description		String	0..1		Description of this share

Commento [SA23]: shares can be related to each other for instance via hierarchy

This is an abstract entity not meant to be instantiated. It SHOULD be used in order to derive specialized entities.

4.8.1 ShareState

Entity	Inherits from	Description
ShareState		State information for a share

Property	Type	Mult.	Unit	Description

This is an abstract entity not meant to be instantiated.

4.8.2 SharePolicy

Entity	Inherits from	Description
SharePolicy		Statements, rules or assertions that specify the correct or expected behavior of a share

Property	Type	Mult.	Unit	Description

This is an abstract entity not meant to be instantiated. It SHOULD be used in order to derive specialized entities.

4.9 Resource

Entity	Inherits from	Description
Resource		An entity useful in a Grid environment part of a logical service, reachable via one or more endpoints and having one or more shares defined on it. A resource usually represents aggregated information

Property		Type	Mult.	Unit	Description
ID	[key]	URI	1		A global unique ID
Name		String	0..1		Human-readable name

This is an abstract entity not meant to be instantiated. For Grid resources requiring a richer set of properties, specific models can be defined by specializing from the Resource entity and adding new properties or relationships. The current proposal contains the Computing Resource specialization (see Section).

Commento [SA24]: add section reference

4.10 Activity

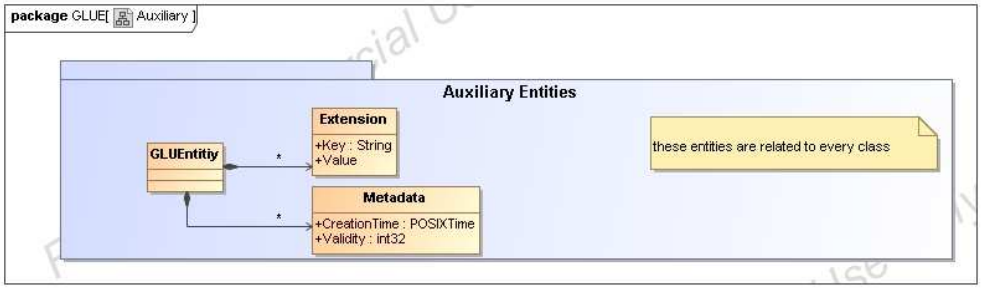
Entity	Inherits from	Description
Activity		An activity is a unit of work managed by a service and submitted via an endpoint; an activity can have relationships to other activities being managed by different services, therefore it shares a common context.

Property		Type	Mult.	Unit	Description
ID	[key]	URI	1		A global unique ID
Type		ActivityType_t	1		The type of this activity

Grid jobs are example of activities for a Computing Service. An interesting type of relationship for jobs derives from its propagation through several services. For instance, a broker service submits a Grid job to a selected execution service, upon completion the execution service submits a logging record to an accounting service. Each of these services will have associated an instance of a Grid job related to the lifecycle of the job within the service. All instances refer to the same conceptual job submitted by the user.

5. Auxiliar Entities

The auxiliary entities currently provides extensibility mechanisms and metadata applicable to all GLUE entities. Widely used extensions will be considered for addition in future GLUE information model revision as primary properties.



5.1 Extension

Entity	Inherits from			Description
Extension				A key,value pair providing extra information not captured in the current model
Property	**Type**	**Mult.**	**Unit**	**Description**
Key	String	1		A local ID, typically an attribute name that could be added in future info model revisions
Value	String	*		A value for the attribute

5.2 Metadata

Entity	Inherits from			Description
Metadata				
Property	**Type**	**Mult.**	**Unit**	**Description**
CreationTime	DateTime	1		Timestamp when the entity instance was generated
Validity	Int32	1	s	The time period for how long the generated information is considered to be relevant by the information provider

6. Conceptual Model of the Computing Service

The conceptual model of the Computing Service is based upon the main entities and uses specializations of Service, Resource, Share, Endpoint and Activity entities. Further computing related concepts such as Execution Environment and Application Environment are introduced.

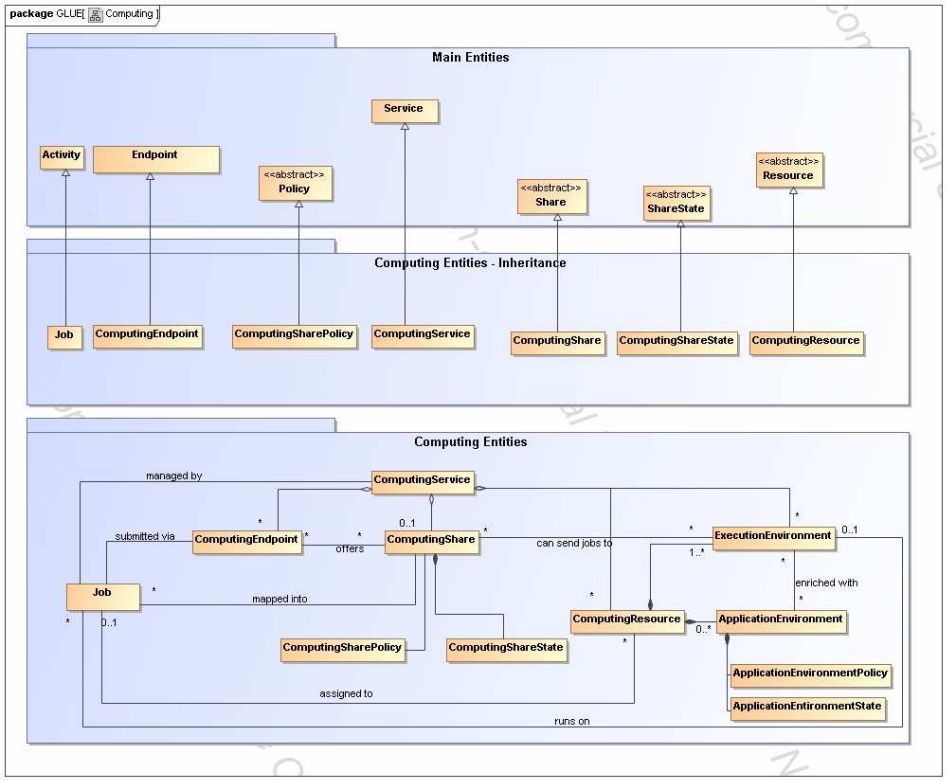


Figure 2 Entities and relationships for the Computing Service conceptual model

6.1 ComputingService

Entity	Inherits from			Description
ComputingService	Service			An abstracted, logical view of actual software components that participate in the creation of a computational capacity in a Grid environment. A computing service exposes one or more endpoints having well-defined interfaces, one or more computing shares and one or more computing resource. The service is autonomous and denotes a weak aggregation among endpoints, the exposed computing resources, and the defined computing shares. The service enables to identify the whole set of entities providing the computing functionality with a persistent name.
Inherited Property	**Type**	**Mult**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
Capability	*ServiceCapability_t*	*		*The capability provided by this service according to the OGSA architecture*
Type	*ServiceType_t*	*1*		*The type of service according to a middleware classification*
QualityLevel	*QualityLevel_t*	*1*		*Maturity of the service in terms of quality of the software components*
StatusPage	*URL*	*		*Web page providing additional information like monitoring aspects*
Complexity	*String*	*0..1*		*Human-readable summary description of the complexity in terms of the number of endpoint types, shares and resources. The syntax should be: endpointType=X, share=Y, resource=Z.*
OtherInfo	*String*	*		*Placeholder to publish info that does not fit in any other attribute. Free-form string, comma-separated tags, (name, value) pair are example of syntax*
Property	**Type**	**Mult**	**Unit**	**Description**
TotalJobs	int32	0..1	job	Number of total jobs
RunningJobs	int32	0..1	job	Number of running jobs
WaitingJobs	int32	0..1	job	Number of jobs waiting in the underlying LRMS's
StagingJobs	int32	0..1	job	Number of jobs that are staging files in/out
SuspendedJobs	Int32	0..1	job	Number of jobs which started their execution, but are suspended (e.g., for preemption)
PreLRMSWaitingJobs	int32	0..1	job	Number of jobs that are in the Grid layer waiting to be passed to the underlying LRMS

The simplest computing service is formed by a computing endpoint exposing an interface for job submission and control, a computing share and a computing resource. In case of a single computing resource exposed by multiple computing endpoints, such computing endpoints have to be considered part of the computing service. In case of a computing endpoint exposing many computing resources, then these computing resources are part of the computing service.

The computing service always aggregate computing endpoints, shares and resources forming a connected set. In other words, Endpoint A exposing resource A via share A and Endpoint B exposing Resource B via share B form two different computing services. On the other side, Endpoint A exposing Resource A via a share and Endpoint B exposing Resource A and B via another share form a computing service.

Properties from previous schemas: nordugrid-cluster-localse (similar to Glue.CESEBind.SEUniqueID)

Commento [SA25]: To be investigated when we have more mature version of Storage Entities schema

6.2 ComputingResource

Entity	Inherits from			Description
ComputingResource	Resource			Grouping concept for a set of different types of execution environments offered through computing endpoint(s). The computing resource usually represents aggregated information. The aggregation is defined by the common local management scope.
Inherited Property	**Type**	**Mult**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*string*	*0..1*		*Human-readable name*
Property	**Type**	**Mult.**	**Unit**	**Description**
LRMSType	LRMSType_t	1		Type of the underlying local resource management system
LRMSVersion	string	0..1		Version of the underlying local resource management system
LRMSOtherInfo	string	0..1		Additional information about the LRMS
TotalJobSlots	int32	0..1		Number of managed job slots
SlotsUsedByLocalJobs	int32	0..1		Number of slots used by jobs submitted via local interface
SlotsUsedByGridJobs	int32	0..1		Number of slots used by jobs submitted via a Grid interface
TotalPhysicalCPUs	int32	0..1		Number of managed physical CPUs accessible via any of the available endpoints (there is one physical CPU per socket)
TotalLogicalCPUs	int32	0..1		Number of managed logical CPUs accessible via any of the available endpoints (a logical CPU corresponds to a CPU visible to the operating system)
TmpDir	string	0..1		
ScratchDir	string	0..1		
DataDir	string	0..1		
Homogeneity	boolean	0..1		True if the computing resource manages only one type of execution environment
NetworkInfo	NetworkInfo_t	0..1		Type of internal network available among the execution environments
LogicalCPUDistribution	string	0..1		Syntax: X*1*:Y*1*, …, X*n*:Y*n* where X*i* is the number of logical CPUs and Y*i* is the number of boxes for the execution environment *i*
GridAreaTotal	int32	0..1	GB	Total shared disk space allocated in the computing resource available to Grid jobs
GridAreaFree	int32	0..1	GB	Free shared disk space allocated in the computing resource available to Grid jobs
GridAreaLifeTime	int32	0..1	min	Lifetime of the Grid job directory after the end of the jobs
CacheTotal	int32	0..1	GB	Total disk space allocated for caching files of Grid jobs
CacheFree	int32	0..1	GB	Free disk space allocated for caching files of Grid jobs

Commento [SA26]: Verify if there are use cases where same worker nodes are managed by different LRMS's

Commento [SA27]: To clarifiy relationship between execEnv (total,used) and CPUs (physical,logical)

Commento [SA28]: Check with GIN work if they are needed

Commento [SA29]: If they are confirmed to stay here, check if they are needed also in the ExecutionEnvironment

Commento [SA30]: Add type in appendix

Commento [SA31]: Evaluate if the 5 attributes Grid* + Cache* have to be moved in the computingEndpoint

A local resource management system like a batch system is an example of aggregation scope. The Operating System can be the simplest case of LRMS.

Entity	Inherits from			Description
Benchmark				Benchmark information about a computing entity
Property	**Type**	**Mult.**	**Unit**	**Description**
Type	Benchmark_t	1		Type of benchmark
Value	int32	1		Value

6.3 ExecutionEnvironment

Entity	Inherits from			Description
ExecutionEnvironment				A description of hardware and software characteristics that defines the environment available to and requestable by a Grid job when submitted to a Computing Service via a Computing Endpoint; the description also includes information about the total/available/used instances of the execution environment
Property	**Type**	**Mult.**	**Unit**	**Description**
ID [key]	URI	1		A global unique ID
PlatformType	platform_t	1		The type of platform running the execution environment instance
VirtualMachine	boolean	0..1		True if the execution environment is based on a virtual machine (in this case, the values of the other attributes are related to the virtualized environment and not to the hosting environment)
TotalInstances	int32	0..1		Number of execution environment instances
UsedInstances	int32	0..1		Number of used execution environment instances (an instance is used when, according to the policies of the LRMS, it cannot accept new jobs because it already runs the maximum number of jobs)
UnavailableInstances	int32	0..1		Number of unavailable execution environment instances because of failures or maintenance
PhysicalCPUs	int32	0..1		Number of physical CPUs in an execution environment instance (counted by socket)
LogicalCPUs	int32	0..1		Number of logical CPUs in an execution environment instance as showed by the operating system
CPUMultiplicity	CPUMultiplicity_t	0..1		Multiplicity of the CPU
CPUVendor	string	0..1		Name of the CPU vendor
CPUModel	string	0..1		CPU model as defined by the vendor
CPUVersion	string	0..1		CPU version as defined by the vendor
CPUClockSpeed	Int32	0..1	MHz	CPU nominal clock speed
MainMemorySize	Int64	1	byte	Amount of RAM (if many jobs run in the same execution environment, they compete for the total RAM)
VirtualMemorySize	Int64	0..1	byte	The amount of Virtual Memory (RAM+Swap)
OSFamily	OSFamily_t	1		Family of the operating system
OSName	OSName_t	0..1		Name of the operating system
OSVersion	string	0..1		Version of the operating system
ConnectivityIn	boolean	1		Permission for direct inbound connectivity, even if limited
ConnectivityOut	boolean	1		Permission for direct outbound connectivity, even if limited
NetworkInfo	NetworkInfo_t	0..1		Type of internal network available among the execution environments

Commento [SA32]: do we need global ID?

Commento [SA33]: Re-evaluate if to use Mega/GigaB or just bytes

An execution environment can be realized in several ways. Examples are a computing node or a virtual machine image that can be requested by a job (different virtual machine images can coexist on the same node). The description about individual software packages is considered by the ApplicationEnvironment class.

6.4 ApplicationEnvironment

Entity	Inherits from	Description
ApplicationEnvironment		Description of the application software environment available within one or more execution environments

Property		Type	Mult.	Unit	Description
ID	[key]	URI	1		A global unique ID
Name		string	1		Name
Version		string	0..1		Version
State		appEnvState_t	0..1		State about the installation
LifeTime		Int32	0..1	s	Time left before removal
License		license_t	0..1		The type of license
InstalledRoot		string	0..1		The directory where the software is installed on the file system
SetupMethod		string	0..1		Method for setting the environment
SetupKey		string	0..1		Fully qualified script for the setting of the application environment
Description		string	0..1		The description of this application environment
MaxCPUs		Int32	0..1		Maximum number of CPUs that can run the application environment at the same time
MaxJobSeats		Int32	0..1		Maximum number of jobs that can use the application environment at the same time
MaxUserSeats		Int32	0..1		Maximum number of users that can use the application environment at the same time
FreeCPUs		Int32	0..1		Available number of CPUs that can run the application environment at the same time
FreeJobSeats		Int32	0..1		Available number of jobs that can use the application environment at the same time
FreeUserSeats		Int32	0..1		Available number of users that can use the application environment at the same time

Commento [SA34]: to be investigated

Commento [SA35]: do we need global ID?

Commento [SA36]: directory _t?

Commento [SA37]: filesystem path type?

Commento [SA38]: Verify naming Seats vs. slots vs CPU; avoid different names for same concepts

Commento [SA39]: To be confirmed

The Application Environment is suggested to be used also for describing application software in terms of a simple tag. In this case, the Name property should be used.

6.5 ComputingEndpoint

Entity	Inherits from	Description
ComputingEndpoint	Endpoint	Endpoint for creating, monitoring, and controlling computational activities called jobs

Inherited Property		Type	Mult	Unit	Description
ID	*[key]*	*URI*	*1*		*A global unique ID*
Name		*String*	*0..1*		*Human-readable name*
URL		*URL*	*1*		*Network location of the endpoint to contact the related service*
Capability		*EndpointCapability_t*	*		*The capability exposed by this interface*
Type		*EndpointType_t*	*1*		*The type of endpoint according to a middleware classification*
QualityLevel		*QualityLevel_t*	*1*		*Maturity of the service in terms of quality of the software components*
SpecificationName		*String*	*0..1*		*Name of the interface specification*
SpecificationVersion		*String*	*0..1*		*Version of the interface*
Implementor		*String*	*0..1*		*Main organization implementing this software component*
ImplementationName		*String*	*0..1*		*Name of the implementation*
ImplementationVersion		*String*	*0..1*		*Version of the implementation (e.g., major version.minor version.pathcversion)*
HealthState		*EndpointHealthState_t*	*1*		*A state representing the health of the endpoint*
HealthStateInfo		*String*	*0..1*		*Textual explanation of the state endpoint*
ServingState		*ServingState_t*	*1*		*The serving state (production, draining, queueing, closed)*
WSDL		*URL*	*		*URL of the WSDL document describing the offered interface (applies to Web Services endpoint)*
SupportedProfile		*URI*	*		*URI identifying a supported profile*

Commento [SA40]: To be verified if we keep both here and in service or only in one part

Commento [SA41]: Suggestion to use URI for identifying categories; Donal will provide examples

Commento [SA42]: What is the relationship between values for this attribute and values for the service.qualityLevel?

Commento [SA43]: Verify if a single value is enough

Semantics	*URL*	*	*URL of a document providing a human-readable description of the semantics of the endpoint functionalities*
StartTime	*DateTime*	*0..1*	*The timestamp for the start time of the endpoint*
IssuerCA	*DN_t*	*0..1*	*Distinguished name of Certification Authority issuing the certificate for the endpoint*

Property	Type	Mult.	Unit	Description
Staging	staging_t	0..1		Supported staging functionalities

6.6 ComputingShare

Entity	Inherits from	Description
ComputingShare		A utilization target for a set of computing resources defined by policies and characterized by status information

Inherited Property	Type	Mult	Unit	Description
LocalID [key]	*String*	*1*		*An opaque local identifier*
Name	*String*	*0..1*		*Human-readable name*
Description	*String*	*0..1*		*Description of this share*

Property	Type	Mult.	Unit	Description

In a computing resource describing a batch system, a typical implementation of a computing share is via a batch queue with the associated policies and status information. The same computing share can be implemented using different batch system configuration strategies. In complex batch systems, it is possible to define different set of policies for the same batch queue, this will imply a share for each set of policies. A computing share can be implemented by virtual machine management systems. The model supports heterogeneity by being able to represent different execution environments associated to the same computing share.

Commento [SA44]: to be extended

6.6.1 ComputingSharePolicy

Entity	Inherits from	Description
ComputingSharePolicy	SharePolicy	Statements, rules or assertions that specify the correct or expected behavior of a computing share

Inherited Property	Type	Mult	Unit	Description

Property	Type	Mult.	Unit	Description
MappingQueue	string	0..1		Name of a queue available in the underlying LRMS where jobs of this share are submitted (different shares can be mapped to the same queue; it is not foreseen that a single share can be mapped to many queues)
MaxWallTime	Int64	0..1	s	The maximum obtainable wall clock time that can be granted to the job upon user request
MinWallTime	Int64	0..1		The minimum Wall clock time for a job
DefaultWallTime	Int64	0..1	s	The default wall clock time allowed to each job by the LRMS if no limit is requested in the job submission description. Once this time has expired the job will most likely be killed or removed from the queue
MaxCPUTime	Int64	0..1	s	The maximum obtainable CPU time that can be granted to the job upon user request

MinCPUTime	Int64	0..1		The minimum CPU time for a job
DefaultCPUTime	Int64	0..1	s	The default CPU time allowed to each job by the LRMS if no limit is requested in the job submission description
MaxTotalJobs	Int64	0..1		The maximum allowed number of jobs in this share
MaxRunningJobs	Int64	0..1		The maximum allowed number of jobs in running state in this share
MaxWaitingJobs	Int64	0..1		The maximum allowed number of jobs in waiting state in this share
MaxPreLRMSWaitingJobs		0..1		The maximum allowed number of jobs that are in the Grid layer waiting to be passed to the underlying LRMS for this share
MaxUserRunningJobs	Int64	0..1		The maximum allowed number of jobs in running state per Grid user in this share
MaxSlotsPerJob	Int64	0..1		The maximum number of slots which could be allocated to a single job (defined to be 1 for a computing service accepting only standard jobs)
MaxStageInStreams	Int64	0..1		The maximum number of streams to stage in files
MaxStageOutStreams	Int64	0..1		The maximum number of streams to stage out files
SchedulingPolicy	schedulingPolicy_t	0..1		Implied scheduling policy of the share
MaxMemory	Int64	0..1	Byte	The maximum RAM that a job can use
MaxDiskSpace	Int64	0..1	Byte	The maximum disk space that a job can use excluding shared area such as cache
DefaultStorageService	URI	0..1		ID of the default Storage Service to be used to store files from jobs in case where no destination Storage Service is explicitly stated
Preemption	boolean	0..1		If true, the computing resource enables preemption of jobs; a preempted job is supposed to be automatically resumed
AssociationEnd		Mult.	Description	
Association to UserDomain via Mapping Policy				

6.6.2 ComputingShareState

Entity	Inherits from			Description
ComputingShareState	ShareState			Set of attributes describing the dynamic state of a computing share
Inherited Property	Type	Mult	Unit	Description

Property	Type	Mult.	Unit	Description
ServingState	ServingState_t	1		The share state (production, draining, queueing, closed)
TotalJobs	int32	0..1	job	Number of total jobs
RunningJobs	int32	0..1	job	Number of running jobs submitted via any type of interface (local and Grid)
LocalRunningJobs	Int32	0..1	job	Number of running jobs submitted via a local interface
WaitingJobs	int32	0..1	job	Number of jobs waiting in the underlying LRMS's submitted via any type of interface (local and Grid)
LocalWaitingJobs	Int32	0..1	job	Number of jobs waiting in the underlying LRMS's submitted via a local interface
StagingJobs	int32	0..1	job	Number of jobs that are staging files in/out
SuspendedJobs	Int32	0..1	job	Number of jobs which started their execution, but

				are suspended (e.g., for preemption)
PreLRMSWaitingJobs	int32	0..1	job	Number of jobs that are in the Grid layer waiting to be passed to the underlying LRMS
EstimatedAverageWaitingTime	Int64	0..1	S	Estimated time to last for a new job from the acceptance to the start of its execution
EstimatedWorstWaitingTime	Int64	0..1	S	The estimated worst waiting time assuming that all jobs run for the maximum wall time
FreeJobSlots	Int64	0..1		Number of single-processor jobs which could be started if no other jobs are submitted and no jobs finish in the interim

Commento [SA45]: investigate if we need a more complex structure, see NorduGrid approach

6.7 ComputingActivity

Entity	Inherits from	Description
ComputingActivity	Activity	An activity managed by an OGSA execution capability service (the computing activity is traditionally called job)

Commento [SA46]: Evaluate which attributes can be moved to the Activity class

Inherited Property	*Type*	*Mult*	*Unit*	*Description*
ID [key]	*URI*	*1*		*A global unique ID*
Type	*ActivityType_t*	*1*		*The type of this activity*

Property	Type	Mult.	Unit	Description
LocalID	string	1		
Name	string	1		
State		1		
ReRunable				
ExitCode				
LRMSExitCode				
Errors				
WaitingPosition				
Owner	string	1		
LocalOwner	string			
RequestedWallTime				
RequestedCPUTime				
RequestedApplicationEnvirnment				
RequestedCPUs				
StdOut				
StdErr				
StdIn				
LogDir				
Comment				
LRMSComment				
ExecutionNodes				
ExecutionCluster				
ExecutionQueue				
UsedWallTime				
UsedMemory				
UsedCPUTime				
NodeCount				
CompletionTime				
SessionDirEraseTime				
ProxyExpirationTime				
SubmissionTime				
SubmissionClient				
SubmissionClientSoftware				

Commento [SA47]: define state model

Commento [SA48]: check consistency with OGF Usage records specs, JSDL and BES

A Job is typically described by an XML document compliant to the JSDL specification. In this specification, the Job is related to a single processor job. Other job types such “collection of jobs” and workflows will be considered in a future revision.

7. Conceptual Model of the Storage Service

See sub-document at the following URL:
To be merged when stable

8. Relationship to OGF Reference Model

In this section, we describe the integration of the GLUE information model with the OGF Reference Model.

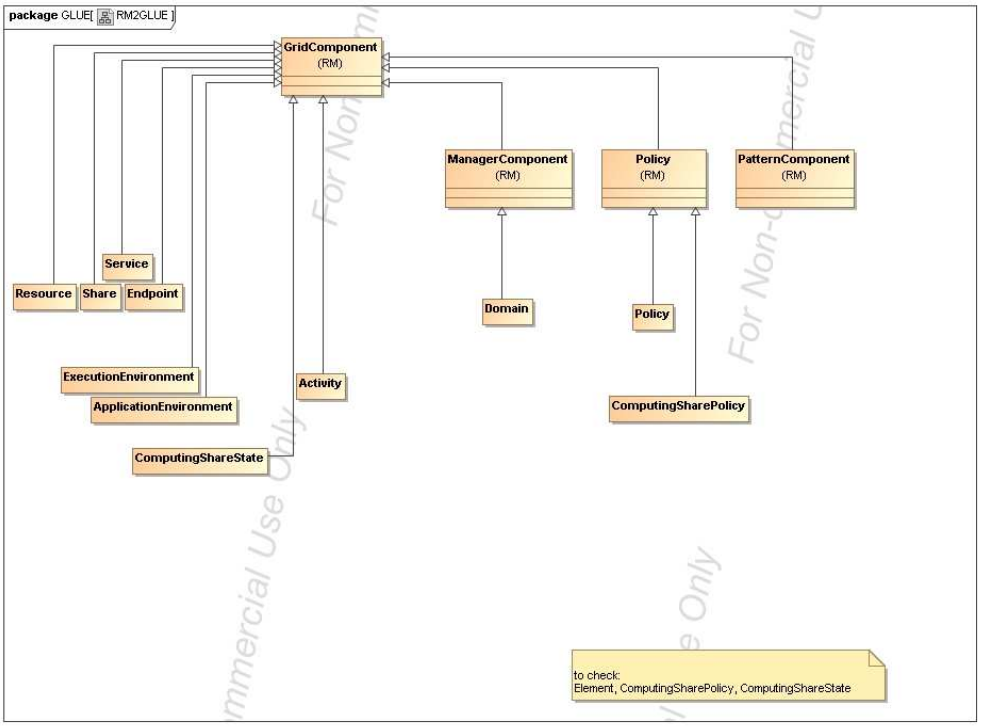


Figure 3 GLUE and Reference Model integration (draft)

9. Template

Commento [SA49]: Describe template

Entity	Inherits from			Description
Property	**Type**	**Mult.**	**Unit**	**Description**

10. Security Considerations

Please refer to RFC 3552 [RESCORLA] for guidance on writing a security considerations section. This section is required in all documents, and should not just say "there are no security considerations." Quoting from the RFC:

> "Most people speak of security as if it were a single monolithic property of a protocol or system, however, upon reflection, one realizes that it is clearly not true. Rather, security is a series of related but somewhat independent properties. Not all of these properties are required for every application.
>
> We can loosely divide security goals into those related to protecting communications (COMMUNICATION SECURITY, also known as COMSEC) and those relating to protecting systems (ADMINISTRATIVE SECURITY or SYSTEM SECURITY). Since communications are carried out by systems and access to systems is through communications channels, these goals obviously interlock, but they can also be independently provided."

11. Author Information

Contact information for authors.

The actual Authors (or Editors) listed on the title page are those committed to taking permanent stewardship for this document – receiving communication in the future and otherwise being responsive to its content. The GFSG recommends at most three Author/Editors be listed on the title page, unless there are compelling reasons to list more.

12. Contributors & Acknowledgements

We gratefully acknowledge the contributions made to this document (in no particular order) by

13. Glossary

Recommended but not required.

14. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

15. Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

16. Full Copyright Notice

Commento [HK50]: I don't think it is just "current year." For example, a document wad started to create from 2003, it should be "2003, 2004, 2005" or "2003-2005."

17. References

Note that only permanent documents should be cited as references. Other items, such as Web pages or working groups, should be cited inline (i.e., see the Open Grid Forum, http://www.ogf.org). References should conform to a standard such as used by IEEE/ACM, MLA, Chicago or similar. Include an author, year, title, publisher, place of publication. For online materials, also add a URL. It is acceptable to separate out "normative references," as IETF documents typically do. Some sample citations:

[glue-wg] The Glue Working Group of OGF, *https://forge.gridforum.org/sf/projects/glue-wg*
[glue-usecases] Glue 2.0 Use Cases (early draft), *https://forge.gridforum.org/sf/go/doc14621*
[glue-1.x] The Glue Schema 1.3, *https://forge.gridforum.org/sf/go/doc14185*
[ng-schema] The NorduGrid/ARC Information System, NORDUGRID-TECH 4, *https://forge.gridforum.org/sf/go/doc14273*
[naregi-schema] NAREGI information and data model, *https://forge.gridforum.org/sf/go/doc14300*
[ogf-ts] Technical Strategy for the Open Grid Forum 2007-2010. GFD-I.113. http://www.ogf.org/documents/GFD.113.pdf
[omii-jra2-djra2.1] Sergio Andreozzi, Antonia Ghiselli, Chunming Hu, Jinlei Jiang, Balazs Konya, Morris Riedel, Davy Virdee, Li Zha. D:JRA2.0 Report on Grid Activities relevant to the identification of new services http://omii-europe.org/OMII-Europe/News/DJRA20.pdf

18. Appendix A: Place-holder values for unknown data

Updated to v.1.3 – to be formatted

Introduction

Whilst people endeavour to provide accurate information, there may be situations where specific GLUE attributes may be assigned place-holder (or dummy) values. These place-holder values carry some additional semantic meaning; specifically, that the correct value is currently unknown and the presented value should be ignored. This appendix describes a set of such place-holder values.

Some attributes within the GLUE schema are required whilst others are optional. If the attribute is optional and the corresponding information is unavailable, the information provider must either publish a place-holder or not to publish the attribute. If the attribute is required, then the information must either publish a place-holder value or refrain from publishing the GLUE object.

If a place-holder value is published, it must conform to the scheme described in this appendix. This is to increase the likelihood that software will understand the nature of the information it receives.

This appendix describes place-holder values that have be chosen so they are obvious "wrong" to humans, unlikely to occur under normal operation and valid within the attribute type. This also allows for detection of failing information provider components.

Use-cases:

There are two principle use-cases for place-holder values, although others may exist.

Scenario 1. a static value has no good default value and has not been configured for a particular site.

Some provisions for GLUE Schema provide templates. These templates may contain attributes that have no good default value; for example, supplying the correct value may require site-specific knowledge. Whilst it is expected that these attributes be configured, it is possible that this does not happen, so exposing the attributes' default values.

Scenario 2. information provider is unable to obtain a dynamic value.

A dynamic value is provided by an information provider by querying the underlying grid resources. This query will use a number of ancillary resources (e.g., DNS, network hardware) that might fail; the grid services might also fail. If an attribute is required and the current value is unobtainable, a place-holder value must be used.

Place-holder values:

This section describes a number of values that can be represented within a given address space (e.g., Strings/UTF-8, Integers, FQDNs, IPv4 address space). Each of the different types are introduced along with the place-holder value and a brief discussion on usage, rational and any other considerations.

1. Simple strings (ASCII/UTF-8) should use "UNDEFINEDVALUE" or should start "UNDEFINEDVALUE:"

Upper-case letters make it easier to spot and a single word avoids any white-space issues.

A short error message can be incorporated into the message by appending the message after the colon.

Examples:
UNDEFINEDVALUE
UNDEFINEDVALUE: unable to contact torque daemon.

Using UNDEFINEDVALUE is a default option for strings that have no widely-known structure. If a value is of a more restrictive sub-type (e.g., FQDNs, FQANs, URIs) described below, then the rules for more restrictive form must be used.

2. Fully qualified domain names: must use a hostname ending either "example.org" for scenario 1, or "invalid" for scenario 2.

RFC 2606 defines two second-level domains: "example.org" and "example.com". These domains have the advantage of ending with a recognisable TLD, so are recognisable as a DNS name. Default configuration (scenario 1, above) must use DNS names that end "example.org"

RFC 2606 also reserves the "invalid" Top-Level-Domain (TLD) as always invalid and clearly so. For dynamic information gathering, a value ending "invalid" must be used.

In both cases, additional information may be included by specifying a prefix to "example.org" or "invalid". This may be used to specify the class of machine that should be present. For dynamic infomation, if the class of machine is not published then the FQDN "unknown.invalid" must be used.

Examples:
www.example.org
your-CE.example.org
unknown.invalid
site-local-BDII.invalid

3. IPv4 addr: must use 192.0.2.250

There are several portions of IPv4 addresses that should not appear on a network, but none that are reserved for documentation or to specify a non-existent address. Using any address leads to the risk of side-effects, should this value be used.

The best option is an IP address from the 192.0.2.0/24 subnet. This subnet is defined in RFC 3330 as "TEST-NET" for use in documentation and example code. For consistency, the value 192.0.2.250 must be used.

4. IPv6 addr: must use 2001:DB8::FFFF

There is no documented undefined IPv6 address. RFC 3849 reserves the address prefix 2001:DB8::/32 for documentation. For consistency, the address 2001:DB8::FFFF must be used.

5. Integers: must use "all nines"
For uint32/int32 this is 999,999,999
For uint64/int64 this is 999,999,999,999,999,999

For integers, all numbers expressible within the encoding (int32/uint32/etc.) are valid so there is no safe choice.

If an unsigned integer is encoded as a signed integer, it is possible to use negative numbers safely. However, these numbers will be unrepresentable if the number is stored as an unsigned integer. For this reason a negative number place-holder must not be used.

The number was chosen for three reasons. First, attribute scales are often chosen to reduce the likelihood of overflow: numbers towards MAXINT (the large number representable in an integer domain) are less likely to appear. Second, repeated numbers stand out more clearly to humans. Finally, the statistical frequency of measured values often follows Benford's law, which indicates that numbers starting with "1" occur far more frequently than those starting with "9" (about six times more likely). For these reasons, information providers must use all-nines to indicate an unknown value.

6. Filepath: must start either "/UNDEFINEDPATH" or "\UNDEFINEDPATH".

As with the simple string, a single upper-case word is recommended. The initial slash indicates that the value is a path. Implementations must use whichever slash is most appropriate for the underlying system (Unix-like systems use a forward-slash). Software should accept either value as an unknown-value place-holder.

Additional information can be encoded as data beyond the initial UNDEFINEDPATH, separated by the same slash as started the value. Additional comments should not use any of the following characters:

\ [ ] ; = " : | , * .

Examples:
/UNDEFINEDPATH
\UNDEFINEDPATH
/UNDEFINEDPATH/Path to storage area
/UNDEFINEDPATH/Broker unavailable

7. Email addresses: must use an undefined FQDN for the domain.

RFC 2822 defines emails addresses to have the form:
<local-part> '@' <domain>

The <domain> must be an undefined FQDN; see above for a complete description. For email addresses, information providers should use "example.org" for scenario 1. and "unknown.invalid" for scenario 2.

The <local-part> may be used to encode a small amount of additional information; for example, it may indicate the class of user to whom the email address should be delivered. If no such information is to be encoded the value "user" must be used.

Examples:
user@example.org
user@unknown.invalid
site-local-contact@example.org
local-admin@example.org

8. Uniform Resource Identifier (URI): schema-specific

RFC 3986 defines URIs as a "federated and extensible naming system." All URIs start with a schema-name part (e.g., "http") and no schema-name has been reserved for undefined or documenting example values.

For any given URI schema ("http", for example), it may be possible to define an unknown value within that name-space. If a GLUE value has only one valid schema, the undefined value must be taken from that schema. If several schemata are possible, one must be chosen from the available options. This should be the most commonly used.

Take care with the URI encoding. All unknown URI values must be valid URIs. If additional information is included, it must be encoded so the resulting URI is valid.

For schemata that may include a FQDN (e.g., a reference to an Internet host), an undefined URI must use an undefined FQDN; see above for details on undefined FQDNs.

URI schemata that reference a remote file (e.g., "http", "ftp", "https"), additional information may be included as the path. The FQDN indicates that the value is a place-holder, indicating an unknown value, so information providers should not specify "UNDEFINEDPATH".

For "file" URIs, the path part must identify the value as unknown and must use the forward-slash variant; see above for details on undefined paths.

For "mailto" URIs [RFC 2368] encapsulates valid email addresses with additional information (such as email headers and message body). Unknown mailto URIs must use an unknown email address (see above). Any additional information must be included in the email body.

There may be other schemata in use that are not explicitly covered in this section. A place-holder value should be agreed upon within whichever domain such schemata are used. This place-holder value should be in the spirit of the place-holder values described so far.

Examples:
http://www.example.org/
httpg://your-CE.example.org/path/to/end-point
httpg://unknown.invalid/User%20certificate%20has%20expired
mailto:site-admin@example.org
mailto:user@maildomain.invalid?body=Problem%20connecting%20to%20WLMS
file:///UNDEFINEDPATH
file:///UNDEFINEDPATH/path%20to%20some%20directory

9. X509 Distinguished Names: must start /O=Grid/CN=UNDEFINEDUSER

X509 uses a X500 namespace, represented as several Relative Domain-Names (RDNs) concatenated by forward-slashes. The final RDN is usually a single common name (CN), although multiple CNs are allowed.

Unknown DN values must have at least two entries: an initial O=Grid followed immediately by CN=UNDEFINEDUSER.

Additional information can be encoded using extra CN entries. These must come after CN=UNDEFINEDUSER.

Examples:
/O=Grid/CN=UNDEFINEDUSER
/O=Grid/CN=UNDEFINEDUSER/CN=Your Grid certificate DN here
/O=Grid/CN=UNDEFINEDUSER/CN=Cannot access SE

10. Fully Qualified Attribute Name (FQAN): must use a VO of "vo.example.org" (for scenario 1.) or "unknown.invalid" (for scenario 2).

The "VOMS Credential Format" document,

http://edg-wp2.web.cern.ch/edg-wp2/security/voms/edg-voms-credential.pdf

states that FQANs must have the form:

/VO[/group[/subgroup(s)]][/Role=role][/Capability=cap]

Where VO is a well-formed DNS name. Unlike DNS names, VO names must be lower-case. The unknown place-holder value for FQAN is derived from the unknown DNS name (see above). It must have no subgroup(s) or Capability specified.

Any additional information must be encoded within a single Role name. Care should be taken that only valid characters (A-Z, a-z, 0-9 and dash) are included.

Examples:
/vo.example.org
/vo.example.org/Role=Replace-this-example-with-your-FQAN
/unknown.invalid
/unknown.invalid/Role=Unable-to-contact-CE-Error-42

11. Geographic locations: must use longitude 0 degrees, latitude 0 degrees.

Meridians of longitude are taken from (-180,180] degrees, whilst parallels of latitude are taken from [-90,90] degrees. For a place-holder value to be a valid location, it must also be taken from these ranges.

By a happy coincidence, the (0,0) location is within the Atlantic Ocean, some 380 miles (611 kilometers) south of the nearest country (Ghana). Since this location is unlikely to be used and repeated numbers are easier for humans to spot, (0,0) must be used to specify an unknown location.

19. Appendix B: Data Types

19.1 ContactType_t

Open enumeration

Value	Description
security	
sysadmin	
usersupport	
general	

19.2 PolicyScheme_t

19.3 DateTime

Extended ISO 8061 format: [-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]
This data type maps the XSD dateTime simple type.
We restrict this syntax to GMT timezone: yyyy '-' mm '-' dd 'T' hh ':' mm ':' ss Z

Commento [SA51]: Ad examples or more description

19.4 ServiceCapability_t

List of values initially drafted from [omii-jra2-djra2.1]. To be refined by examples. Open enumeration.

Value	Description
Security.Authentication	Capacity of providing authentication mechanisms for Grid users machine and services
Security.CredentialStorage	Capacity of providing an online credential repository that allows users to securely obtain credentials when and where needed
Security.Delegation	capacity for a user to give a service the authority to undertake specific activities or decisions on its behalf
Security.Authorization	capacity of handling authorization aspects, making authorization decisions about the subject and the requested mode of access based upon combining information from a number of distinct sources
Security.IdentyMapping	capacity of mapping Grid-level credentials to local level credentials (e.g., mapping a user X.509 certificate into a UNIX account).
Security.AttributeAuthority	capacity of associating a user with a set of attributes in a trusted manner to a relying party, by way of digitally signed assertions
Security.Accounting	capacity of systematically recording, reporting, and analyzing the usage of resources
Data.Transfer	capacity of moving a file from one network location to another. It refers to the actual transfer (e.g., as performed by protocols like FTP, GridFTP, or HTTP)
Data.Management.Transfer	capacity of managing a transfer of files from the start to the completion
Data.Management.Replica	capacity of managing the creation of file replicas upon request
Data.Management.Storage	capacity of managing a storage resource, from simple systems like disk-servers to complex hierarchical systems
Data.Naming.Resolver	capacity of resolving one name to another (for example, search the associated abstract name to a certain human-oriented name)
Data.Naming.Scheme	capacity of attaching names to data resources. (To evaluate if it should moved to the main category infrastructure instead of data). In OGSA, a three-level naming scheme is defined: (1) human-oriented name, (2) abstract name and (3) address
Data.Access.Relational	capacity of providing access to a relational data source

Data.Access.XML	capacity of providing access to an XML data source
Data.Access.FlatFiles	capacity of providing access to a flat file
Information.Model	capacity of modelling resources based on a community accepted definition
Information.Discovery	capacity of locating unknown resources or services, possibly satisfying a set of requirements
Information.Logging	capacity of recording data, often chronologically
Information.Monitoring	capacity of periodically observing measurements, transform them and make available to users or other applications
Information.Provenance	capacity of providing long-term storage of information related to Grid activity and to let this information be accessed by users or other applications.
ExecMan.BES	capacity of executing a job or set of jobs.
ExecMan.JobDescription	capacity of letting users be able to describe a job submission request based on a machine-processable language
ExecMan.JobManager	capacity of managing the execution of a job or set of jobs from start to finish
ExecMan.ExecutionAndPlanning	capacity of building schedules for jobs, that is, the capability of defining mappings between services and resources, possibly with time constraints
ExecMan.CandidateSetGenerator	capacity of determining the set of resources on which a nit of workcan execute
ExecMan.Reservation	capacity of managing reservation of resources for future usage

19.5 ServiceType_t

Every item should start with org.MIDDLEWARENAME. Open enumeration.

Value	Description
org.glite.wms	
org.glite.lb	
…	

19.6 QualityLevel_t

Closed enumeration

Value	Description
development	
testing	
pre-production	
production	

19.7 EndpointCapability_t

The initial set of values is drafted from [omii-jra2-djra2.1]. At the moment, we use the same of ServiceCapability_t. Open enumeration

19.8 EndpointState_t

Closed enumeration

Value	Description
ok	
warning	
critical	
unknown	
other	

19.9 DN_t

19.10 License_t

Closed enumeration

Value	Description
Opensource	
Commercial	
Unknown	

19.11 CPUMultiplicity_t

Open enumeration:

Value	Description
singlecpu-singlecore	The execution environment is run by a single CPU with a single core
singlecpu-multicore	The execution environment is run by a single CPU with multiple cores
multicpu-singlecore	The execution environment is run by multiple CPUs with a single core each
multicpu-multicore	The execution environment is run by multiple CPUs with a multiple cores each

19.12 platform_t

Open enumeration:

Value	Description
IA32	
IA64	

19.13 OSFamily_t

Open enumeration:

Value	Description
linux	
macos	
windows	
solaris	

19.14 appEnvState_t

Open enumeration:

Value	Description
tested	
installed	
dynamic	
toberemoved	

19.15 schedulingPolicy_t

Open enumeration:

Value	Description
fairshare	Statistically guarantees the allocated share
fifo	First-In First-Out
random	Random choice

19.16 staging_t

Open enumeration:

Value	Description
none	No staging of files supported
stagingin	Automatic staging in of files supported
stagingout	Automatic staging out of files supported
stagginginout	Automatic staging in and out of files supported

19.17 OSName_t

Open enumeration:

Value	Description
scientific linux cern	
scientific linux	
windows xp	
windows vista	
ubuntu	
debian	
centos	
leopard	

In the final section, this page will contain the XML Schema rendering of GLUE 2.0. Meanwhile, the draft schema can be located at the following page:

http://forge.ogf.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/GLUE2XMLSchema

20. Appendix C: XML Schema Rendering

21. Appendix D: LDAP Rendering

In the final section, this page will contain the LDAP rendering of GLUE 2.0 (both schema and Directory Information Tree description). Meanwhile, the draft schema can be located at the following page:

http://forge.ogf.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/GLUE2LDAP

22. Appendix E: Relational Rendering

In the final section, this page will contain the Relational Schema rendering of GLUE 2.0. Meanwhile, the draft schema can be located at the following page:

http://forge.ogf.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/GLUE2Relational

Editorial Notes:

- Format Paul contribution

Checks before final submission

- In each table, verify that “Inherited Properties” are consistent with original
- All attributes having type, mult and description
- All data type being defined in appendix
- Consistency between main entities and derived models
- All comments answered and removed
- Verify the rules to follow for authors vs. contributors for the GLUE context in order to give credits