GWD-R, GWD-I or GWD-C

Authors:
Sergio Andreozzi (editor), INFN
Stephen Burke, RAL
Felix Ehm, CERN
Laurence Field, CERN
Gerson Galang, SAPAC
Balazs Konya, Lund University
Maarten Litmaath, CERN
Paul Millar, Desy
JP Navarro, ANL

GLUE WG
http://forge.ogf.org/sf/sfmain/do/viewProject/projects.glue-wg

March 06, 2008

GLUE Specification v. 2.0 (draft 34)

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. Rendering to concrete data models such as XML Schema, LDAP and relational are provided.

Editorial To Do:

Check:

- 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
- Check authors/contributors list and verify addresses
- Rules for properties
  - Properties name all with first letter of each component word capitol
  - Added data types with suffix _t, capitol as properties
  - Decide if to use multiple in unit of measure
    - http://en.wikipedia.org/wiki/International_System_of_Units
    - http://en.wikipedia.org/wiki/SI_prefix
  - Enumeration values all small letters

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.

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].

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. Each GLUE entity has either an ID attribute or LocalID attribute. The ID is a global identifier, while the LocalID is an identifier local to a container entity which is specified in the definition.

Both ID and LocalID MUST not be interpreted by the user or the system as having any meaning other than as an identifier. In particular, there is no relationship between an ID and a network endpoint. The ID MUST be compliant with the syntax of a URI. The usage of URN (subset of URI) is recommended.

As regards unit of measure, for multiple of bytes we refer to the SI prefix (http://en.wikipedia.org/wiki/SI_prefix), therefore GB is 10^9 Bytes and not 2^30 Bytes (the latter are GibiBytes).

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.

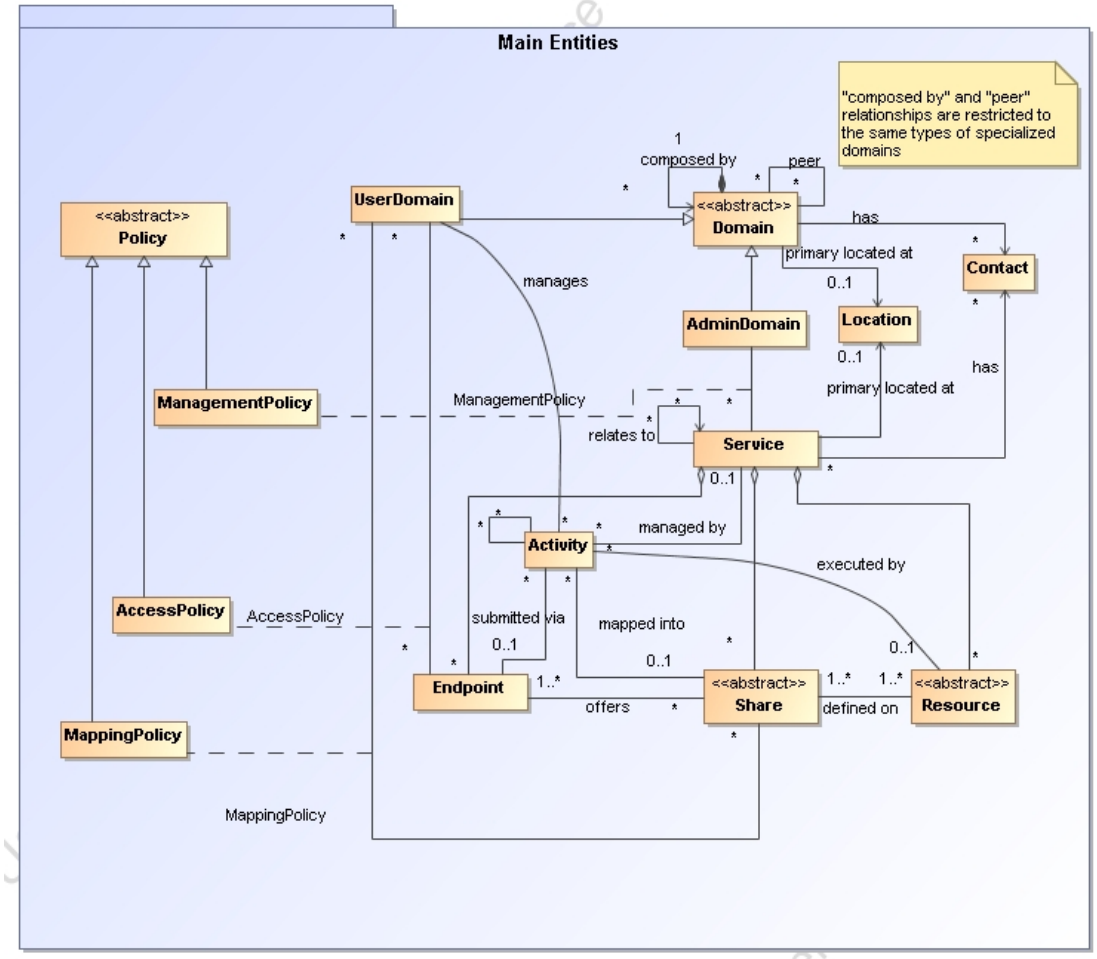

Figure 1 GLUE main entities and their relationships

4.1 Location

Entity	Inherits from			Description
Location				A geographical position
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	LocalID_t	1		An opaque identifier local to the associated Service or Domain
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	LocalID_t	1		An opaque identifier local to the associated Service or Domain
URL	URI	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

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	URI	*		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

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	*URI*	*		*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

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	*URI*	*		*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	UInt32	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

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.

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

4.4 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 zero 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	1..*		The provided capability 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	URI	*		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.

4.5 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	URI	1		Network location of the endpoint to contact the related service
Capability	EndpointCapability_t	1..*		The provided capability according to the OGSA architecture
Technology	EndpointTechnology_t	0..1		Technology used to implement the endpoint
InterfaceName	String	1		Name of the type of interface
InterfaceVersion	String	1		Version of the type of interface
WSDL	URI	*		URL of the WSDL document describing the offered interface (applies to Web Services endpoint)
SupportedProfile	URI	*		URI identifying a supported profile
Semantics	URI	*		URI of a document providing a human-readable description of the semantics of the endpoint functionalities
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)
QualityLevel	QualityLevel_t	1		Maturity of the service in terms of quality of the software components
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
StartTime	DateTime_t	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
DowntimeAnnounce	DateTime_t	0..1		The timestamp for the announcement of the next scheduled downtime
DowntimeStart	DateTime_t	0..1		The starting timestamp of the next scheduled downtime
DowntimeEnd	DateTime_t	0..1		The ending timestamp of the next scheduled downtime
DowntimeInfo	String	0..1		Description of the next scheduled downtime
Association End		**Mult.**	**Description**	
Association to UserDomain via Access Policy				

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)

4.6 Share

Entity	Inherits from			Description
Share				A utilization target for a set of resources offered via related endpoints defined by configuration parameters and characterized by status information
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID [key]	LocalID_t	1		An opaque identifier local to the associated Service
Name	String	0..1		Human-readable name
Description	String	0..1		Description of this share

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

4.7 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).

4.8 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 (named Computing Activities in GLUE) 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.

4.9 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.

4.9.1 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**

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.

4.9.2 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**
LocalID	LocalID_t	1		An opaque identifier local to the Service to which the associated entity belongs to
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

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.

4.9.3 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**
LocalID	LocalID_t	1		An opaque identifier local to the Service to which the associated entity belongs to
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

This entity can be used to express which UserDomains can consume a certain share of resources. Given a certain UserDomain and a certain Share, there MUST be at most one MappingPolicy instance which property Default is true.

Auxiliary 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.

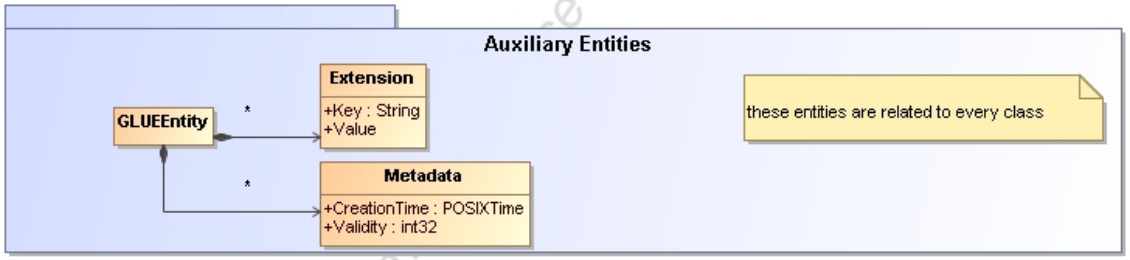

Figure 2 Auxiliary Entities

4.10 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

4.11 Metadata

Entity	Inherits from			Description
Metadata				
Property	**Type**	**Mult.**	**Unit**	**Description**
CreationTime	DateTime_t	1		Timestamp when the entity instance was generated
Validity	UInt64	1	s	The time period for how long the generated information is considered to be relevant by the information provider

5. 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.

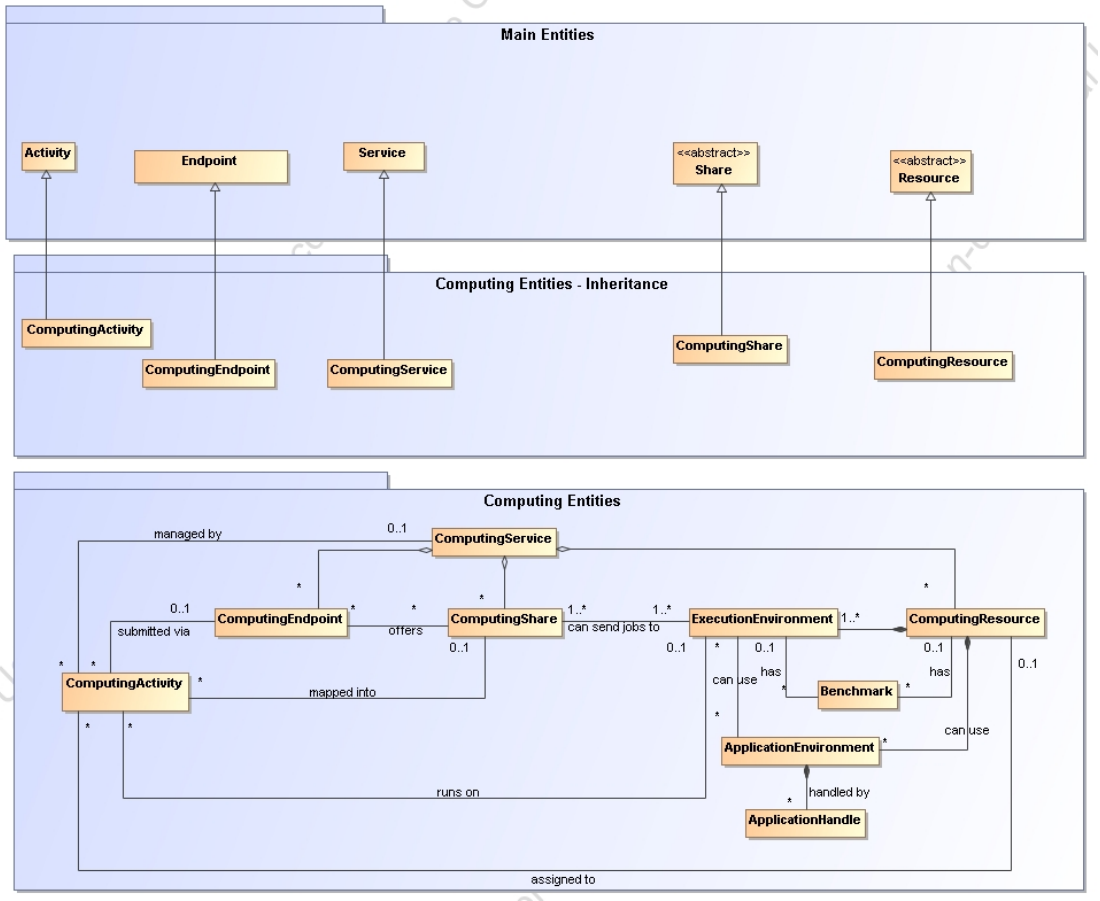

Figure 3 Entities and relationships for the Computing Service conceptual model

In the computing entities section, we extensively use the concept of slot. A slot is defined as a portion of executable time in an execution environment instance which can be consumed by a job. Usually, there is one slot per logical CPU. Jobs can consume several slots at the same time (e.g., MPI jobs).

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	*URI*	*		*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	UInt32	0..1	job	Number of total jobs
RunningJobs	UInt32	0..1	job	Number of running jobs
WaitingJobs	UInt32	0..1	job	Number of jobs waiting in the underlying LRMS's
StagingJobs	UInt32	0..1	job	Number of jobs that are staging files in/out
SuspendedJobs	UInt32	0..1	job	Number of jobs which started their execution, but are suspended (e.g., for preemption)
PreLRMSWaitingJobs	UInt32	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)

5.1 ComputingEndpoint

Entity	Inherits from			Description
ComputingEndpoint	Endpoint			Endpoint for creating, monitoring, and controlling computational activities called jobs; it can be used to expose also complementary capabilities (e.g., reservation, proxy manipulation)
Inherited Property	**Type**	**Mult**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
URL	*URI*	*1*		*Network location of the endpoint to contact the related service*
Capability	*EndpointCapability_t*	*1..**		*The provided capability according to the OGSA architecture*
Technology	*EndpointTechnology_t*	*0..1*		*Technology used to implement the endpoint*
InterfaceName	*String*	*1*		*Name of the type of interface*
InterfaceVersion	*String*	*1*		*Version of the type of interface*
WSDL	*URI*	*		*URL of the WSDL document describing the offered interface (applies to Web Services endpoint)*
SupportedProfile	*URI*	*		*URI identifying a supported profile*
Semantics	*URI*	*		*URI of a document providing a human-readable description of the semantics of the endpoint functionalities*
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)*
QualityLevel	*QualityLevel_t*	*1*		*Maturity of the service in terms of quality of the software components*
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*
StartTime	*DateTime_t*	*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*
DowntimeAnnounce	*DateTime_t*	*0..1*		*The timestamp for the announcement of the next scheduled downtime*
DowntimeStart	*DateTime_t*	*0..1*		*The starting timestamp of the next scheduled downtime*
DowntimeEnd	*DateTime_t*	*0..1*		*The ending timestamp of the next scheduled downtime*
DowntimeInfo	*String*	*0..1*		*Description of the next scheduled downtime*
Property	**Type**	**Mult.**	**Unit**	**Description**
Staging	Staging_t	0..1		Supported staging functionalities

5.2 ComputingShare

As regards CPU Time and Wall Time related properties, there is the need for a way to normalize them depending on the computing capacity of the execution environment. The approach proposed in GLUE is to add two attributes in the Execution Environment which refer to the scaling factor to be used to compute the CPU/Wall time that a job will get if it will be assigned to such an execution environment via a certain share. It is important that a job will get always at least the advertised CPU/Wall time. This means that the reference Execution Environment for the normalization should be always the fastest among those available in the whole Computing Service. For this Execution Environment, the scaling factor MUST be equal to 1. The CPU/Wall time values published by a share refer to the time that the job will get when mapped to this Execution Environment. For the other Execution Environments, the time should be normalized according to the defined scaling factors.

Entity	Inherits from			Description
ComputingShare				A utilization target for a set of computing resources defined by a set of configuration parameters and characterized by status information
Inherited Property	**Type**	**Mult**	**Unit**	**Description**
LocalID [key]	*LocalID_t*	*1*		*An opaque identifier local to the associated Service*
Name	*String*	*0..1*		*Human-readable name*
Description	*String*	*0..1*		*Description of this share*
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	UInt64	0..1	s	The maximum obtainable wall clock time that can be granted to the job upon user request (unormalized value)
MinWallTime	UInt64	0..1	s	The minimum Wall clock time for a job (unormalized value); if a job requests a lower time, than it can be rejected; if a job requests at least this value, but runs for a shorter time, than it might be accounted for this value
DefaultWallTime	UInt64	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 (unormalized value)
MaxCPUTime	UInt64	0..1	s	The maximum obtainable CPU time that can be granted to the job upon user request on a single CPU (unormalized value)
MaxCPUsTime	UInt64	0..1	s	The maximum obtainable CPU time that can be granted to the job upon user request across all assigned CPUs (unormalized value)
MinCPUTime	UInt64	0..1	s	The minimum CPU time for a job (unormalized value); if a job requests a lower time, than it can be rejected; if a job requests at least this value, but runs for a shorter time, than it might be accounted for this value
DefaultCPUTime	UInt64	0..1	s	The default CPU time allowed to each job by the LRMS if no limit is requested in the job submission description (unormalized value)
MaxTotalJobs	UInt32	0..1	job	The maximum allowed number of jobs in this share
MaxRunningJobs	UInt32	0..1	job	The maximum allowed number of jobs in running state in this share
MaxWaitingJobs	UInt32	0..1	job	The maximum allowed number of jobs in waiting state in this share
MaxPreLRMSWaitingJobs	UInt32	0..1	job	The maximum allowed number of jobs that are in the Grid layer waiting to be passed to the underlying LRMS for this share
MaxUserRunningJobs	UInt32	0..1	job	The maximum allowed number of jobs in running state per Grid user in this share
MaxSlotsPerJob	UInt32	0..1	slot	The maximum number of slots which could be allocated to a single job (defined to be 1 for a computing service accepting only single-slot jobs)
MaxStageInStreams	UInt32	0..1	stream	The maximum number of streams to stage in files
MaxStageOutStreams	UInt32	0..1	stream	The maximum number of streams to stage out files
SchedulingPolicy	SchedulingPolicy_t	0..1		Implied scheduling policy of the share
MaxMemory	UInt64	0..1	MB	The maximum RAM that a job can use
MaxDiskSpace	UInt64	0..1	GB	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
ServingState	ServingState_t	1		The share state
TotalJobs	UInt32	0..1	job	Number of total jobs in any state
RunningJobs	UInt32	0..1	job	Number of running jobs submitted via any type of interface (local and Grid)
LocalRunningJobs	UInt32	0..1	job	Number of running jobs submitted via a local interface
WaitingJobs	UInt32	0..1	job	Number of jobs waiting in the underlying LRMS's submitted via any type of interface (local and Grid)
LocalWaitingJobs	UInt32	0..1	job	Number of jobs waiting in the underlying LRMS's submitted via a local interface
StagingJobs	UInt32	0..1	job	Number of jobs that are staging files in/out
SuspendedJobs	UInt32	0..1	job	Number of jobs which started their execution, but are suspended (e.g., for preemption)
PreLRMSWaitingJobs	UInt32	0..1	job	Number of jobs that are in the Grid layer waiting to be passed to the underlying LRMS
EstimatedAverageWaitingTime	UInt64	0..1	s	Estimated time to last for a new job from the acceptance to the start of its execution
EstimatedWorstWaitingTime	UInt64	0..1	s	The estimated worst waiting time assuming that all jobs run for the maximum wall time
FreeSlots	UInt32	0..1	slot	Number of free slots
UsedSlots	UInt32	0..1	slot	Number of slots used by running jobs
RequestedSlots	UInt32	0..1	slot	Number of slots which are needed to execute all waiting and staging jobs
ReservationPolicy	ReservationPolicy_t	0..1		Type of reservation policy

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.

5.3 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
LRMSReservation	Boolean	0..1		True if the LRMS supports advance reservation
LRMSBulkSubmission	Boolean	0..1		True if the LRMS supports the bulk submission
TotalSlots	UInt32	0..1	slot	Number of managed slots
SlotsUsedByLocalJobs	UInt32	0..1	slot	Number of slots used by jobs submitted via local interface
SlotsUsedByGridJobs	UInt32	0..1	slot	Number of slots used by jobs submitted via a Grid interface
TotalPhysicalCPUs	UInt32	0..1	Ph.CPU	Number of managed physical CPUs accessible via any of the available endpoints (there is one

				physical CPU per socket)
TotalLogicalCPUs	UInt32	0..1	Log.CPU	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	UInt64	0..1	GB	Total shared disk space allocated in the computing resource available to Grid jobs
GridAreaFree	UInt64	0..1	GB	Free shared disk space allocated in the computing resource available to Grid jobs
GridAreaLifeTime	UInt64	0..1	s	Lifetime of the Grid job directory after the end of the jobs
CacheTotal	UInt64	0..1	GB	Total disk space allocated for caching files of Grid jobs
CacheFree	UInt64	0..1	GB	Free disk space allocated for caching files of Grid jobs

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.

5.4 Benchmark

Entity	Inherits from			Description
Benchmark				Benchmark information about a computing entity
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	LocalID_t	1		An opaque identifier local to the Computing Service
Type	Benchmark_t	1		Type of benchmark
Value	Real32	1		Value

5.5 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**
LocalID	LocalID_t	1		An opaque identifier local to the Computing Service
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	UInt32	0..1		Number of execution environment instances
UsedInstances	UInt32	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	UInt32	0..1		Number of unavailable execution environment instances because of failures or maintenance
PhysicalCPUs	UInt32	0..1		Number of physical CPUs in an execution environment instance (counted by socket)
LogicalCPUs	UInt32	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	UInt32	0..1	MHz	CPU nominal clock speed
CPUTimeScalingFactor	Real32	0..1		Factor used by the LRMS to scale the the CPU time (CPU Time divided by CPUTimeScalingFactor); for the reference execution environment, use 1;
WallTimeScalingFactor	Real32	0..1		Factor used by the LRMS to scale the the Wall time (Wall Time divided by WallTimeScalingFactor)
MainMemorySize	UInt64	1	MB	Amount of RAM (if many jobs run in the same execution environment, they compete for the total RAM)
VirtualMemorySize	UInt64	0..1	MB	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	*		Type of internal network available among the execution environments

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.

5.6 ApplicationEnvironment

Entity	Inherits from			Description
ApplicationEnvironment				Description of the application software environment available within one or more execution environments
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	LocalID_t	1		An opaque identifier local to the Computing Service
Name	String	1		Name
Version	String	0..1		Version
State	AppEnvState_t	0..1		State about the installation
LifeTime	UInt64	0..1	s	Time left before removal
License	License_t	0..1		The type of license
Description	String	0..1		The description of this application environment
ParallelType	ParallelType_t	0..1		The type of supported parallel execution
MaxSlots	UInt32	0..1	slot	Maximum number of slots that can run jobs using the application environment at the same time
MaxJobs	UInt32	0..1	job	Maximum number of jobs that can use the application environment at the same time
MaxUserSeats	UInt32	0..1	user seat	Maximum number of user seats that can use the application environment at the same time
FreeSlots	UInt32	0..1	slot	Available number slots that can run jobs using the application environment at the same time
FreeJobs	UInt32	0..1	slot	Number of new jobs that could start their execution and use the application environment at the same time
FreeUserSeats	UInt32	0..1	user seat	Free seats for new users that can use the application environment at the same time

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.

5.7 ApplicationHandle

Entity	Inherits from			Description
ApplicationHandle				Technique for accessing the application
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	LocalID_t	1		An opaque identifier local to the Computing Service
Type	ApplicationHandle_t	1		(module, softenv, executable, path)
Value	String	1		Description for the technique

5.8 ComputingActivity

Entity	Inherits from			Description
ComputingActivity	Activity			An activity managed by an OGSA execution capability service (the computing activity is traditionally called job)
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**
LRMSID	String	0..1		The job ID as assigned by the LRMS
Name	String	0..1		The job name as specified by the user in the job description document
State	ComputingActivityState_t	1		The state of the job according to

				the Grid state model for jobs
RestartState	ComputingActivityState_t	0..1		The state from which a failed job can restart upon a client request
ExitCode	Int32	0..1		The exit code as returned by the executable of the job
LRMSExitCode	String	0..1		The exit code provided by the batch system
Error	String	*		Error messages as provided by the software components involved in the management of the job
LRMSWaitingPosition	UInt32	0..1		For a waiting in the underlying LRMS, the position in the queue
UserDomain	String	0..1		Selected user domain by the job owner (an owner can belong to several user domains, it should decide which one to choose when submitting a job)
Owner	String	1		The Grid identity of the job's owner; in case of anonymity is required, the value CONFIDENTIAL should be advertised
LocalOwner	String	0..1		The local user name to which the job's owner is mapped
RequestedWallTime	UInt64	0..1	s	The wall clock time requested by the job
RequestedCPUTime	UInt64	0..1	s	The CPU time requested by the job
RequestedApplicationEnvironment	String	*		The name of the requested ApplicationEnvironment (the value should match the name property of the ApplicationEnvironment)
RequestedCPUs	UInt32	0..1	Log. CPU	The number of requested logical CPUs
StdIn	String	0..1		The name of the file which is used as the standard input of the job
StdOut	String	0..1		The name of the file which contains the standard output of the job
StdErr	String	0..1		The name of the file which contains the standard error of the job
LogDir	String	0..1		The name of the directory which contains the logs related to the job generated by the Grid layer (usually the directory is private to the job)
ExecutionNode	String	*		Hostname of a cluster node which is running the job (multi-node jobs are described by several instances of this attribute)
QueueName	String	0..1		The name of the LRMS queue to which this job was queued
UsedWallTime	UInt64	0..1	s	The consumed wall clock time of the job
UsedCPUTime	UInt64	0..1	s	The consumed CPU time of the job (in case of multi-CPU jobs, this value refers to the sum of all CPU times)
UsedMainMemory	UInt64	0..1	MB	The RAM used by the job
SubmissionTime	DateTime_t	0..1		Time when the job was submitted to a computing endpoint
LRMSSubmissionTime	DateTime_t	0..1		Time when the job was submitted to the LRMS by the Grid layer
StartTime	DateTime_t	0..1		Time when the job entered in the LRMS running state
LRMSEndTime	DateTime_t	0..1		Time when the job entered its final LRMS state
EndTime	DateTime_t	0..1		Time when the job entered its final Grid state
GridAreaEraseTime	DateTime_t	0..1		The time when the dedicated Grid job area will be removed
ProxyExpirationTime	DateTime_t	0..1		The expiration time of the proxy

				related to the job
SubmissionHost	String	0..1		The name of the host from which the job was submitted (e.g., IP address, port and host name)
SubmissionClientName	String	0..1		The name of the client software which was used to submit the job
OtherMessages	String	*		Optional job messages provided by either the Grid Layer or the LRMS

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.

6. Conceptual Model of the Storage Service

Like the Computing Service, the conceptual model of the Storage Service is based upon the main entities and uses specializations for those entities. Further on, storage related concepts such as StorageCapacity, StorageMappingPolicy, StorageEnvironment and StorageAccessProtocol are introduced.

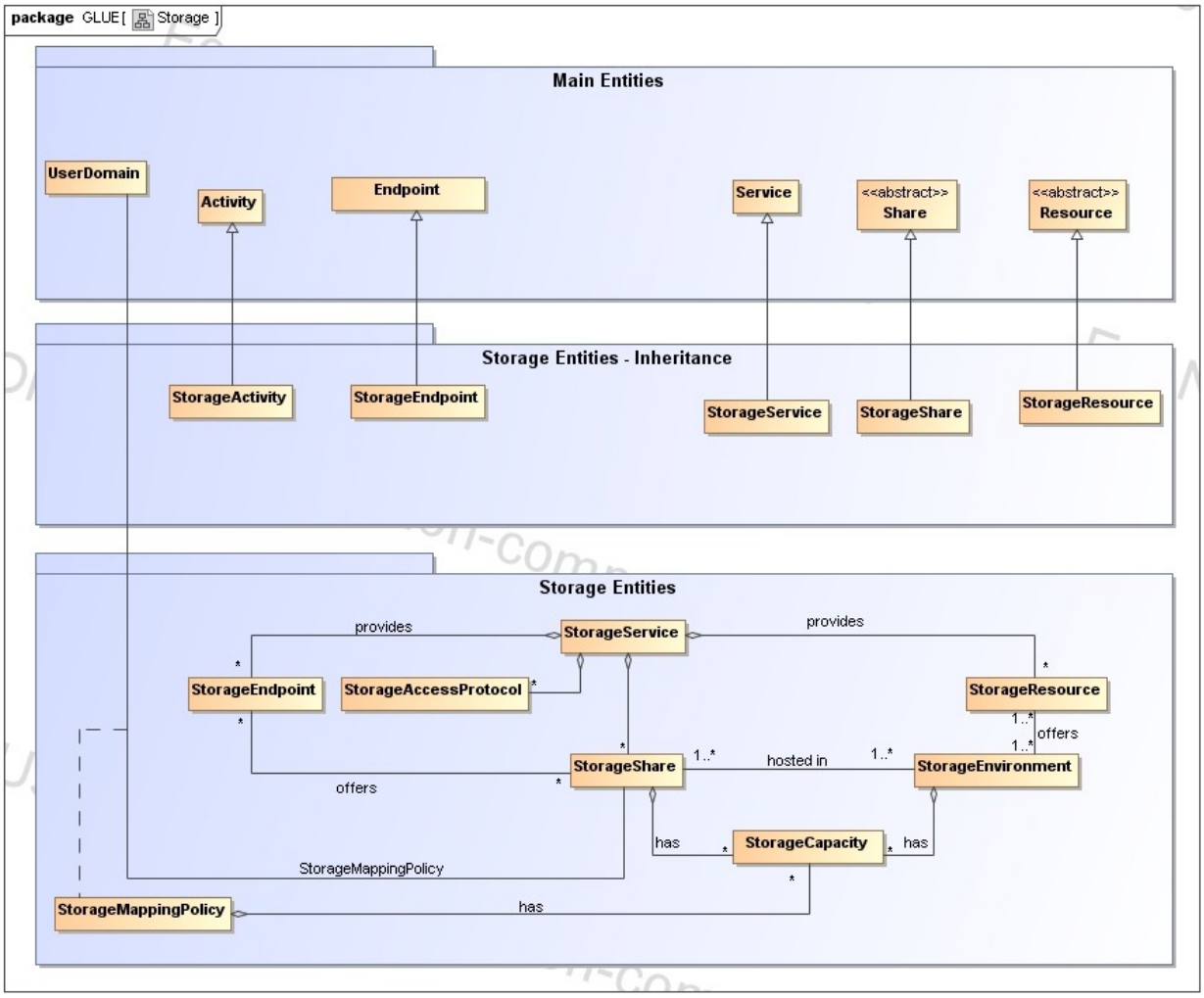

Figure 4 Entities and relationships for the Storage Element model

6.1 StorageService

Entity	Inherits from			Description
StorageService	Service			An abstracted, logical view of actual software components that participate in the creation of a storage capacity in a Grid environment. A storage service exposes one or more endpoints having well-defined interfaces and one or more storage shares. The service is autonomous and denotes a weak aggregation among endpoints and the defined storage shares. The service enables to identify the whole set of entities providing the storage 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	*URI*	*		*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**

The storage service is formed by storage endpoints offering interfaces to the service and storage shares which represent allocated storage capacity on the service which can be utilized for storage activities. The access to the endpoint and shares is controlled by a mapping policy instance.

A storage service is instantiated when it offers at least one endpoint. It may have zero or more shares. A storage service without a storage share does not offer any storage capabilities.

6.2 StorageEndpoint

Entity	Inherits from			Description
StorageEndpoint	Endpoint, Downtime			Endpoint for accessing and controlling storage activities.
Inherited Property	**Type**	**Mult**	**Unit**	**Description**
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
URL	*URI*	*1*		*Network location of the endpoint to contact the related service*
Capability	*EndpointCapability_t*	*1..**		*The provided capability according to the OGSA architecture*
Technology	*EndpointTechnology_t*	*0..1*		*Technology used to implement the endpoint*
InterfaceName	*String*	*1*		*Name of the type of interface*
InterfaceVersion	*String*	*1*		*Version of the type of interface*
WSDL	*URI*	*		*URL of the WSDL document describing the offered interface (applies to Web Services endpoint)*
SupportedProfile	*URI*	*		*URI identifying a supported profile*
Semantics	*URI*	*		*URI of a document providing a human-readable description of the semantics of the endpoint functionalities*
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)*
QualityLevel	*QualityLevel_t*	*1*		*Maturity of the service in terms of quality of the software components*
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*
StartTime	*DateTime_t*	*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*
DowntimeAnnounce	*DateTime_t*	*0..1*		*The timestamp for the announcement of the next scheduled downtime*
DowntimeStart	*DateTime_t*	*0..1*		*The starting timestamp of the next scheduled downtime*
DowntimeEnd	*DateTime_t*	*0..1*		*The ending timestamp of the next scheduled downtime*
DowntimeInfo	*String*	*0..1*		*Description of the next scheduled downtime*
Property	**Type**	**Mult.**	**Unit**	**Description**
Capability	String	*		Other information regarding this Endpoint

A StorageEndpoint exposes one interface of how a storage service can be contacted. It gives information about the control protocol and its status as well as possible downtimes.

A storage endpoint is linked to storage shares and thereby knows which shares it gives access to. The Capability field can be used to specify other restrictions such as WAN read-only/LAN read-write.

6.3 StorageShare

Entity	Inherits from			Description
StorageShare	Share			A utilization target for a set of storage resources defined by a set of configuration parameters and characterized by status information
Inherited Property	*Type*	*Mult*	*Unit*	*Description*
LocalID [key]	*LocalID_t*	*1*		*An opaque identifier local to the Storage Service*
Name	*String*	*0..1*		*Human-readable name*
Property	Type	Mult.	Unit	Description
Path	String	0..1		A namespace where files are logically assigned to when they are stored into this share.
ExpirationMode	ExpirationMode_t	*		The expiration modes which this Share supports for files (neverExpire, warnWhenExpired, releaseWhenExpired)
Tag	String	*		A user defined tag for additional information
State	ServingState_t	1		The serving state

A storage share represents allocated logical storage space within a storage environment and can be accessed through the service's endpoint(s).
The access of a user domain to storage shares is described by storage mapping policies. A share may have more than one environment for which it may offer storage capacity information. This capacity information should reflect the share's environment in the type attribute.

6.4 StorageResource

Entity	Inherits from			Description
StorageResource	Resource			Grouping concept for a set of different types of storage environments offered through storage endpoint(s). The storage 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
ImplementationName	String	0..1		The name of the software offering storage environment(s) through the associated storage service endpoints.
ImplementationVersion	String	0..1		The version of the software offering storage environment(s) through the associated storage service endpoints.
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

A storage resource may be instantiated if it offers at least one storage environment.

6.5 StorageEnvironment

Entity	Inherits from			Description
StorageEnvironment				A description of a storage sub-system with homogeneous characteristics that defines the environment where storage shares can be created
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID [key]	LocalID_t	1		An opaque identifier local to the Storage Service
AccessLatency	AccessLatency_t	0..1		(Online, Nearline, Offline)
RetentionPolicy	RetentionPolicy_t	0..1		(Custodial, Output, Replica)

6.6 StorageAccessProtocol

Entity	Inherits from			Description
StorageAccessProtocol				Describes the access protocols of a Service.
Property	**Type**	**Mult.**	**Unit**	**Description**
LocalID	LocalID_t	1		An opaque identifier local to the Storage Service
Type	StorageAccessProtocol_T	1		The name of the protocol
Version	String	1		The version of the protocol
MaxStreams	UInt32	0..1	stream	The number of parallel streams this protocol supports

6.7 StorageCapacity

Entity	Inherits from			Description
StorageCapacity				Describes size and state of an homogenous storage extent
Property	**Type**	**Mult.**	**Unit**	**Description**
Type	StorageSpace_t	1		Type of storage space (e.g., online, nearline,…)
FreeSize	UInt64	0..1	GB	The free space left
UsedSize	UInt64	0..1	GB	The used space
TotalSize	UInt64	0..1	GB	The total size
ReservedSize	UInt64	0..1	GB	The reserved

The storage capacity entity may only be specified if it is aggregated into a storage environment, storage share or storage mapping policy. It must not be given as an own entity.
The type of the storage capacity may reflect a descriptive name for the related entity for which it gives size information. A share for example, may have two types of storage space determined by its related storage environments. For each type a storage capacity entity need to be instantiated (e.g. nearline and online).

6.8 StorageMappingPolicy

Entity	Inherits from		Description
StorageMappingPolicy	MappingPolicy		Statements, rules or assertions that specify which instantiation of a Domain may use the associated StorageShare..
Inherited Property			*Description*
LocalID	*LocalID_t*	*1*	*An opaque identifier local to the Storage Service*
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*

Property	Type	Mult.	Unit	Description
Name	String	1		An descriptive name for this Policy
Path	String	0..1		VO specific namespace to when utilizing an associated Share
Tag	String	0..1		A user defined tag for this policy

The storage mapping policy describes the relationship of a user domain to a storage share and keeps further finer-grained information of how the user domain may utilize the storage share. The path attribute defines a different namespace from the associated storage share in order to allow more fine-grained control for organizing VO specific data in different paths on the same share.

6.9 SSCSLink

Entity	Inherits from			Description
SSCSLink	ServiceServiceLink			Statements that describe network link quality of a storage service to a computing service.
Inherited Property	Type			Mult
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
Property	Type	Mult.	Unit	Description
SS_ID	URI	1		The ID of the StorageService
CE_ID	URI	1		The ID of the linked ComputingService
ProtocolLocalID	LocalID_t	0..1		The local identified of the access protocol of the associated storage service
NetworkInfo	NetworkInfo_t	0..1		Type of network available among the storage service and computing service environments
Bandwidth	UInt32	0..1	MBit/sec	The technical available bandwidth between the storage and computing service

6.10 CSSSLink

Entity	Inherits from			Description
CSSSLink	ServiceServiceLink			Statements that describe how a computing service can make the associated storage service locally available.
Inherited Property	Type			Mult
ID [key]	*URI*	*1*		*A global unique ID*
Name	*String*	*0..1*		*Human-readable name*
Property	Type	Mult.	Unit	Description
SS_ID	URI	1		The ID of the StorageService
CE_ID	URI	1		The ID of the linked ComputingService
SSPath	String	1		The default path of the Storage Service
CSPath	String	1		The path of the storage service as seen from the computing service

7. Relationship to OGF Reference Model

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

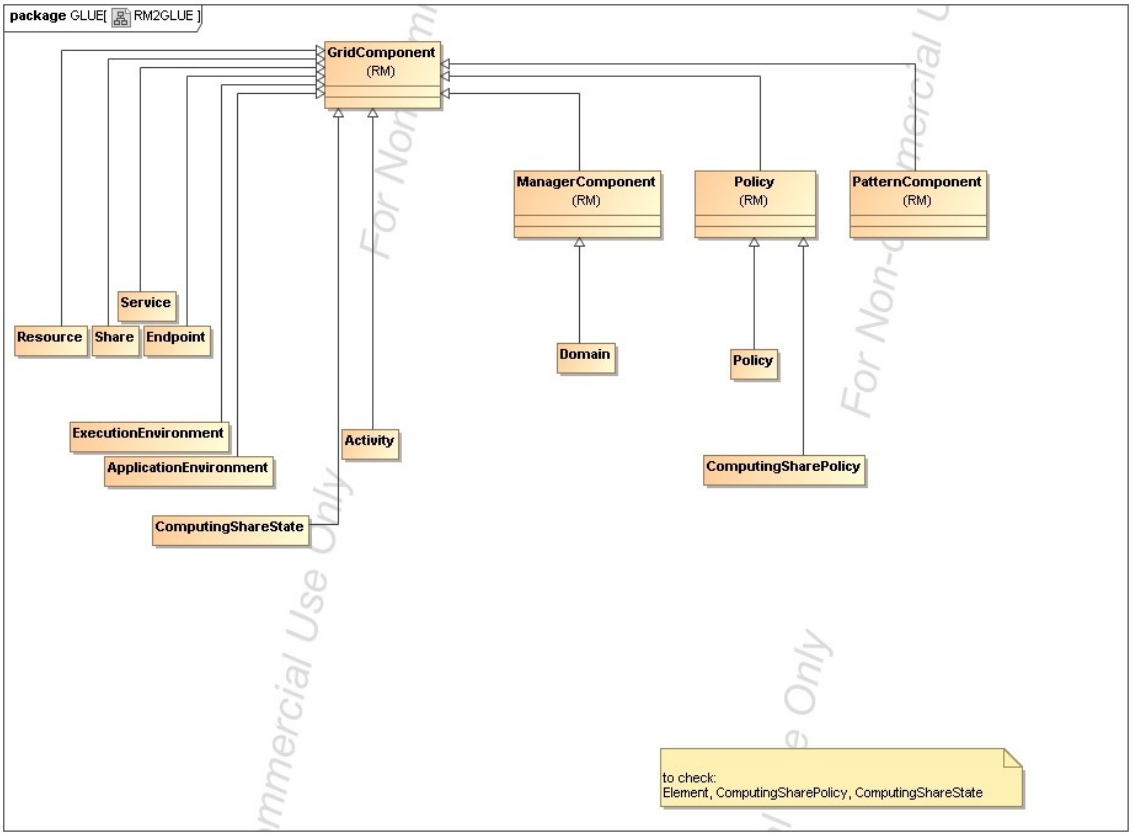

Figure 5 GLUE and Reference Model integration (draft)

8. Template

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

9. 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.”

10. Author Information

Sergio Andreozzi, INFN
Stephen Burke, RAL
Felix Ehm, CERN
Laurence Field, CERN
Gerson Galang,
Balazs Konya, Lund University,
Maarten Litmaath, CERN
Paul Millar, Desy
JP Navarro

11. Contributors & Acknowledgements

We gratefully acknowledge the contributions made to this document (in no particular order) by Shiraz Memon, Matt Viljonen and Steve Traylen.

12. Glossary

Recommended but not required.

13. 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.

14. 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.

15. Full Copyright Notice

16. 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

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

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.

17.1 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.

17.2 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.

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.

17.2.1 Fully qualified domain names

They 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

17.2.2 IPv4 address

It 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.

17.2.3 IPv6 addr

It 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.

17.2.4 Integers

It 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.

17.2.5 File path

It 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

17.2.6 Email addresses

It 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

17.2.7 Uniform Resource Identifier (URI)

It is 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

17.2.8 X509 Distinguished Names

It 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

17.2.9 Fully Qualified Attribute Name (FQAN)

It 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

17.2.10 Geographic locations

It 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.

18. Appendix B: Data Types

18.1 LocalID_t

The base type is the string with the following restrictions: the characters , / = are not allowed. Alphanumerical + ':' +'_' + '.' + …

18.2 ContactType_t

Open enumeration

Value	Description
security	
sysadmin	
usersupport	
general	

18.3 PolicyScheme_t

Value	Description

18.4 DN_t

Distinguished Name as defined by RFC 4514 (http://www.rfc-editor.org/rfc/rfc4514.txt).

18.5 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.
executionmanagement.jobexecution	capacity of executing a job or set of jobs.
executionmanagement.jobdescription	capacity of letting users be able to describe a job submission request based on a machine-processable language
executionmanagement.jobmanager	capacity of managing the execution of a job or set of jobs from start to finish
executionmanagement.executionandplanning	capacity of building schedules for jobs, that is, the capability of defining mappings between services and resources, possibly with time constraints
executionmanagement.candidatesetgenerator	capacity of determining the set of resources on which a nit of workcan execute
executionmanagement.reservation	capacity of managing reservation of resources for future usage
executionmanagement.dynamicvmdeploy	capacity of dynamically deploying a virtual machine image in a worker node

18.6 ServiceType_t

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

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

18.7 QualityLevel_t

Closed enumeration

Value	Description
development	
testing	
pre-production	
production	

18.8 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

18.9 EndpointTechnology_t

Open enumeration.

Value	Description
webservice	

jndi	
legacy	

18.10 EndpointHealthState_t

Closed enumeration

Value	Description
ok	It was possible to check the state of the endpoint and it appeared to be functioning properly
warning	It was possible to check the state of the endpoint, but it appeared to be above some "warning" threshold or did not appear to be working properly
critical	It was possible to check the state of the endpoint and either it was not running or it was above some "critical" threshold
unknown	It was not possible to check the state of the endpoint
other	It was possible to check the state of the endpoint, but this is not covered by the defined states

18.11 ServingState_t

Closed enumeration

Value	Description
production	The endpoint is both accepting and serving requests
draining	The endpoint is not accepting requests, but is serving requests in the queue
queueing	The endpoint is accepting requests, but is not serving them
closed	The endpoint is not accepting request nor is serving them

18.12 ActivityType_t

Open enumeration

Value	Description
computing	

18.13 DateTime_t

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

18.14 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
staginginout	Automatic staging in and out of files supported

18.15 SchedulingPolicy_t

Open enumeration:

Value	Description

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

18.16 ReservationPolicy_t

Closed enumeration:

Value	Description
none	No reservation is supported
mandatory	Jobs must be submitted only via advance reservation
optional	Jobs can be submitted via advance reservation, but this is not required

18.17 LRMSType_t

Open enumeration:

Value	Description
openpbs	
lsf	

18.18 NetworkInfo_t

Open enumeration

Value	Description
gigabitethernet	
myrinet	
infiniband	

18.19 Benchmark_t

Open enumeration

Value	Description
specint2000	
specfp2000	
bogomips	

18.20 platform_t

Open enumeration:

Value	Description
i386	
amd64	
powerpc	

18.21 CPUMultiplicity_t

Closed 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

18.22 OSFamily_t

Open enumeration:

Value	Description
linux	
macos	
windows	
solaris	

18.23 ParallelType_t

Open enumeration:

Value	Description
mpi	Parallel execution based on mpi library
openmp	Parallel execution based on openmp library
none	No supported parallel execution

18.24 ApplicationHandle_t

Open enumeration:

Value	Description
module	Access based on loading modules via Environment Modules [REF]
softenv	Access based on loading SoftEnv
path	Access based on using an explicit path where the software is installed on the file system
executable	Access based on running directly the main executable of the application (this may require set-up of the environment)

18.25 OSName_t

Open enumeration:

Value	Description
scientificlinuxcern	
scientificlinux	
windowsxp	
windowsvista	
ubuntu	

debian	
centos	
leopard	

18.26 AppEnvState_t

Open enumeration:

Value	Description
tested	
installed	
dynamic	
toberemoved	

18.27 License_t

Closed enumeration

Value	Description
opensource	
commercial	
unknown	

18.28 SetupMethod_t

Closed enumeration

Value	Description
default	
setenv	
…	

18.29 ComputingActivityState_t

Open enumeration:

Value	Description

18.30 ExpirationMode_t

Closed enumeration:

Value	Description
neverExpire	The file will never be automatically deleted by the storage system. It must be removed by an authorized source.
warnWhenExpired	The storage system can only remove expired files if it keeps an archived copy of those.
releaseWhenExpired	The storage system is responsible for removing the expired files.

18.31 StorageAccessProtocol_t

Open enumeration:

Value	Description

gsiftp	
nfs	
afs	
rfio	
gsirfio	
dcap	
gsidcap	
root	
https	
other	

18.32 StorageEnvironmentType_t

Closed enumeration:

Value	Description
volatile	
durable	
permanent	

18.33 AccessLatency_t

Closed enumeration:

Value	Description
online	
nearline	
offline	

18.34 RetentionPolicy_t

Closed enumeration:

Value	Description
Custodial	
Output	
Replica	

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

19. Appendix C: XML Schema Rendering

20. 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

21. 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