EGI-InSPIRE

EGI PROFILE FOR THE USE OF THE GLUE 2.0 INFORMATION SCHEMA

Document identifier:	EGI GLUE 2 profile
Date:	**21/06/2013**
Document Status:	**v 1.0**
Dissemination Level:	**PUBLIC**
Document Link:	https://documents.egi.eu/document/1324

Authors

Stephen Burke, egi.eu

Abstract

This document specifies how the GLUE 2.0 information schema should be used in EGI. It gives detailed guidance on what should be published, how the information should be interpreted, what kinds of uses are likely, and how the information may be validated to ensure accuracy.

DOCUMENT LOG

Issue	Date	Comment	Author
1.0	21-06-2013	First release	**Stephen Burke**

TABLE OF CONTENTS

1 INTRODUCTION

This document defines the usage of the GLUE 2.0 schema in the EGI Grid. It extends the schema specification document [R1] with more detailed semantics for particular attributes, specifies conditions under which objects and attributes should and should not be published, and defines some additional information to be published in OtherInfo attributes. It also includes guidelines for validating the accuracy of the published information.

The schema contains many enumerated types defined as open, meaning that new types can be added freely. All such types should be recorded centrally in a way defined by the GLUE OGF working group, e.g. on the group wiki page [R2], and hence are not defined as part of this profile as they are effectively part of the schema itself, albeit easily extendable. It is intended that the lists of enumerated types will also be available in a downloadable format [R3]. Errata are also recorded on the GLUE wiki page.

GLUE 2 usage is still in its infancy, and this document will therefore evolve in the light of experience. The document is versioned, and information providers should publish the version with which they comply as described below. All substantive changes will be recorded in the Change Log.

The primary information system in EGI is currently based on LDAP, and this document is therefore currently targeted at the LDAP rendering of the schema, but the majority of the profile is technology-independent, although future evolution of the information system technology may necessitate a revision. Appendices briefly describe the current information system technology (BDII) and also the GOC DB and EMIR, and the implications for the use of the schema in each case. However, this document does not specify any details of the information system technology or implementation, only the semantics of the published information itself.

In general this document does not specify the frequency at which information should be updated or the latency for it to appear in the information system, as these depend on the underlying technology. Each attribute has a natural lifetime related to the rate at which the underlying property changes, which may vary from a few seconds to several years. This is often loosely classified as dynamic or static information according to whether it relates to properties which change autonomously (dynamic) or only when reconfigured by a system administrator (static). It is likely that there will be a minimum latency – for the BDII technology this is several minutes – which is considerably longer than the underlying rate of change of the fastest-changing attributes. The future evolution of the information system may involve using different technologies for different kinds of information to match the desired update rate.

In most cases it is not possible to make general statements about what information should be present in the information system, as publication may be prevented by downtime of the underlying services and may vary depending on the detailed configuration of the services. However, there are two general constraints. Firstly, EGI has a general operational concept of a Grid site which is reflected in many operational tools, with a name which is defined in the GOC DB. In the information system this corresponds to the existence of an **AdminDomain** object with an ID attribute identical to the name of the site, and this object is required to exist if any information for that site is published. Secondly, the **Service** class aggregates the information from a number of dependent objects, for example **Endpoint**s and **Share**s. The existence of the **Service** object is therefore required if any of the dependent objects exist. Other more detailed constraints on object existence are mentioned in the descriptions of the objects.

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 (see http://www.ietf.org/rfc/rfc2119.txt).

1.1 Attribute Classification

Attributes are classified according to what kind of use case(s) they are expected to satisfy. These are only the potential uses, and there is no general guarantee that attributes will in fact be used in such a way. Knowledge of the likely uses may have implications, for example, in the requirements for accuracy, latency, update rate and the extent to which values may or should be cached,

- Service Discovery (SD): information which describes what services exist and what their properties are. This is expected to be information which changes rather slowly and hence may usefully be cached. Such information will typically be used to filter the results of a query, e.g. in the WMS JDL Requirements expression, or be returned as service metadata.
- Service Selection (SS): information, often highly dynamic, which facilitates the choice of a particular service at a given time among several which satisfy the SD requirements. This may for example reflect the current load on the service. This will typically be used to rank the results of an SD query, e.g. in the JDL Rank expression.
- Monitoring (M): dynamic or static information which may be useful to monitor and display the properties and state of services across the Grid. This may be presented either individually for each service or aggregated across a set of sites. Specific monitoring tools are likely to display only a subset of this information.
- Oversight (O): information, generally static or slowly changing, which can be aggregated to provide a high-level picture of Grid resources, for example the total installed CPU and disk capacity. This category also includes information which is useful for the overall management of the Grid, for example the distribution of versions of particular services.
- Diagnostic (D): information which facilitates the tracing of problems with Grid services or within the information system itself.

The schema marks attributes as mandatory (multiplicity of at least 1) or optional (minimum multiplicity 0). This profile extends this categorisation:

- Mandatory (M): Information which MUST be published. This includes all attributes which are mandatory in the schema, but adds some which are required by EGI even though optional in the schema. Information consumers may treat missing attributes in this category as an error. Note that this does not imply that publication of the containing object itself is mandatory, only that such attributes MUST be present *if* the object is published.

- Recommended (R): Information which SHOULD be published, and which may be absent only if there is a strong reason, for example that the attribute is meaningless in a particular case or is in practice impossible to determine. Information consumers may treat missing attributes in this category as an error or a null value as appropriate (null meaning that the value is an empty string).

- Desirable (D): Information whose publication is OPTIONAL, but which is likely to be useful in the EGI context, and developers are therefore encouraged to publish where possible. Information consumers should regard missing attributes as null or unknown as appropriate.

- Optional (O): Information whose publication is OPTIONAL. Developers are free to publish such information where it seems to be appropriate, or where it is known to be useful for a particular service in the context of a specific client. Information consumers should not rely on the presence of such information unless documented for a specific service, but may assume that it is accurate if published.

- Undesirable (U): Information whose publication is potentially damaging in the EGI context, for example if it would expose sensitive information or would imply an excessive data volume. Information in this category SHOULD NOT be published. Note that this only applies to the EGI information system as a whole – such information may be present at a service-specific level, but if so it should be filtered out of the Grid-wide aggregation.

1.2 Validation

One of the target purposes of this profile is to facilitate validation of the content of the information system. To that end, where possible this document defines values which should be regarded as exceptional, in one of four categories:

- INFO: values which are potentially valid but which are unknown or seem anomalous to the validation tool. This should for example apply to any open enumeration with an unknown value. There are likely to be many false-positives from these tests and hence INFO-level results should only be displayed on request, and should be maskable both as specific instances and in groups so that known false-positives can be excluded.

- WARNING: values which are likely, but not certain, to be erroneous, for example a very large number of queued jobs. Specific instances should be maskable if they are determined to be correct. This should also be used for values which are not compliant with this profile but are valid within the schema.

- ERROR: values which are definitely invalid, for example unknown values for closed enumerations and negative values for unsigned integers.
- FATAL: values which invalidate the entire object, for example badly-formed IDs which are not unique. Objects with such values may be removed from the information system. This category also includes faults which will result in objects being rejected by an LDAP server, for example missing mandatory attributes or strings containing invalid characters.

These rules are intended to give guidance rather than to be prescriptive, and the authors of validation tools may modify or enhance them if appropriate.

As described below, information providers which intend to be compliant with this profile must publish that fact using OtherInfo attributes. If possible, validation tools should provide a switch to apply the validation tests only to such objects.

It is assumed that all enumerated types should be validated, so this is not stated explicitly for each attribute. Open enumerations can be freely extended, so it must be easy to update the list against which attributes are validated. The lists of allowed values for all enumerated types are maintained by the GLUE Working Group as an extension to the schema definition.

GLUE 2 is defined to be case-sensitive, so all strings must be in the correct case. Where practical validation tools should check for values which differ only in case from a known value, and raise a WARNING in such a case (or an INFO for free-text strings where differences in case may be reasonable).

The working language of EGI is English, hence strings SHOULD in general also be in English, apart from names. In addition, the LDAP schema used in the BDII currently has a bug that restricts the allowed character set to 7-bit ASCII. This will be fixed to allow any UTF-8 string, but the fix will take some time to deploy. Strings should consist of a single line, and hence SHOULD NOT contain carriage return or line feed characters, or any other control characters including tabs. Strings SHOULD be limited to a maximum length of 255 characters. Leading and trailing spaces SHOULD NOT be present.

The schema document defines various placeholder values in Appendix A, and if published these should be treated as INFO unless the information for a particular attribute specifies otherwise, in which case it may be ignored or treated as a WARNING as appropriate

INFO-level checks are intended as a way of catching typing errors and other simple mistakes, and will in many cases be heuristics rather than fixed rules since the object is to detect values which are technically valid but are outliers or exceptions in some way. One way to “seed” such checks is to query the information system for common published values to use in an internal list, with exceptions being flagged as INFO. In such a case there should be a simple way to extend the list of known values.

All relations between objects should be published as implied by the schema and the LDAP rendering (i.e. as ForeignKey attributes). Validation tools should generate WARNINGs if referenced objects are missing (except where omission is explicitly allowed, e.g. **UserDomain**s). They may generate ERRORs if references appear to be inconsistent or incorrect, for example if a ForeignKey refers to an object of the wrong type or if an object lacks a reference which is expected to point to it, although full validation of referential integrity is likely to be a difficult task. Use of other technologies, e.g. EMIR, will mean that relations are represented in a different way, but the same basic principles apply.

The validation described here is purely textual and does not include any checks on the functionality of any service, validity of any URLs etc. Such tests should be documented in the context of the test plans for each service, and for specific clients like the WMS and GFAL which make use of the information system.

The LDAP servers used in the BDII will not allow objects to be published if they are not conformant with the schema. Any errors appear in the BDII logs, but in a somewhat cryptic format which makes it difficult to identify the source of the problem. It is therefore desirable to have explicit validation of the output of information providers to identify any non-compliance with the schema directly and report it to the system administrator in a useful way, e.g. via a Nagios probe. It may also be useful to edit the LDIF before insertion into the BDII to avoid the situation where individual non-compliant attributes, e.g. strings containing invalid characters, result in entire objects being absent.

There is no expectation that any single validation tool will implement every check indicated in this document, and in some cases it may be too complex to perform a full validation, but it is desirable to perform as much validation as possible. In particular, middleware developers should consider adding such validation inside their information providers to catch problems at the earliest possible stage. There is a range of possible uses for validation tools:

- Internal validation by the resource BDII on a service, reporting to the system manager e.g. via Nagios. The glue-validator [R4] tool is being extended to perform this.
- Grid-wide validation of the entire information system content, for example as performed for GLUE 1 by gstat.
- Service-specific tests, for example to ensure that all storage systems publish information in a consistent way.
- Acceptance tests for new middleware.
- Internal checks by client software which makes use of the information system, for example GFAL and the WMS.

1.3 Installed Capacity

For GLUE 1.3, WLCG published a document [R5] which specifies ways in which the schema can be used to publish information on the total installed computing and storage resources in the Grid. This document updates this information for GLUE 2.0, with specific information being given in the sections on the computing and storage classes.

1.4 Document Structure

The rest of this document consists of detailed specifications for each schema class and each attribute within the class, in the same order as the schema specification. Detailed descriptions of attributes are only given necessary; information in the schema document is not repeated here except as needed for clarification. Class names are given in **bold**. Inherited attributes are only mentioned again if they have a specialised interpretation in the derived class. Standardised information is presented in tables in the following format:

Attribute	Req.	SD	SS	M	O	D	Validation
Detail	M			✓	✓		INFO if the format is invalid or the information is inconsistent with the GOC DB.

The first column contains the attribute names, and the second the definition of the extent to which EGI requires the attribute to be published using the categorisation defined above. The following five columns have a tick if the attribute is likely to be useful for a particular kind of use case, again as defined above. The final column contains suggested validation rules.

2 MAIN ENTITIES

2.1 Entity

The **Entity** class is not instantiable in its own right, but it forms part of every object and is therefore central to the use of the schema.

2.1.1 CreationTime

This attribute specifies when the object was created. For dynamic objects this is the time at which the information provider runs, and for static objects created by a configuration tool it is the time at which that tool runs. If different attributes are defined at different times the CreationTime should be the latest time. It is RECOMMENDED to publish the CreationTime as it provides a useful diagnostic tool to track the propagation of objects through the information system and allow stale information to be identified.

2.1.2 Validity

This specifies the expected maximum lifetime of the object, relative to the CreationTime. Publication is Desirable, but it SHOULD NOT be published if the CreationTime is not. For dynamic information providers a typical value may be 1 hour, and for static objects 1 year, but other values may be used if appropriate.

2.1.3 ID

Each object must have a globally unique ID, different from every other ID in the information system. It is defined to be the case that the ID has no semantic content and no software may interpret or parse it in any way. The schema specifies that the ID is a URI, and the intention was to define a naming format which would rigorously ensure uniqueness. However, to date no such format has been defined – if it is defined in the future then such a format SHOULD be used.

In the interim, the ID MUST contain a string which is interpretable as a DNS name in a domain associated with the site where the ID is generated, and MUST NOT contain a string interpretable as a DNS name in any other domain. There is no requirement that the name is in fact registered in the DNS, although it will typically be the FQDN of a host on which a service runs, or a DNS alias for such a host. Software which generates IDs should use whatever other methods seem appropriate to disambiguate IDs associated with the same DNS name, bearing in mind that in general a host may run multiple services of the same or different types, and no particular deployment scenario should be assumed unless mandatory for the service concerned. As an example, a MyProxy Service object could have an ID of <hostname>_MyProxy_<object-code>, where <hostname> is the FQDN of the machine hosting the service and <object-code> distinguishes the Service ID from the ID of any other objects related to the MyProxy service on the same host. If a formal URI scheme is defined it is likely to be based on similar principles.

There are two exceptions to these rules. For site names (**AdminDomain** IDs) the GOC DB serves as a central registry which ensures uniqueness, and the names may therefore be freely chosen subject to not already being registered. For VO names (**UserDomain** IDs) a DNS style is strongly recommended but not absolutely mandatory. In this case the EGI Operations Portal acts as a registry for existing names.

The persistency of an ID should be as long as reasonably possible, and sufficient for the likely uses. For example, a **StorageService** ID should persist at least as long as any files stored by the service.

2.1.4 Name

This is a human-readable name/description for the object. Publication is OPTIONAL and any reasonable text is acceptable.

2.1.5 OtherInfo

This attribute can be used to publish any additional information which is not otherwise available in defined attributes. The format will often be of the form “key=value”. This is not mandatory, any reasonable format is acceptable, but that format SHOULD be used for information which naturally follows a key/value pattern. Service documentation should include a specification for any such information. OtherInfo attributes are effectively local to their object class (including any derived classes); if there is a risk of name clashes within a class the attribute SHOULD be prefixed with the name of the implementation which generates it. OtherInfo itself is a multivalued attribute, but it should be assumed that any specific key may only occur once unless explicitly stated as multivalued.

This profile defines some OtherInfo attributes usable in any object as follows:

- InfoProviderHost=<host> gives the real hostname (FQDN) on which the information provider was executed.
- InfoProviderName=<name> gives the name of the information provider which generated the object.
- InfoProviderVersion=<version> gives the version of the information provider which generated the object.

- ProfileName=EGI indicates that the object is intended to be compliant with this document. This is REQUIRED for any information provider which intends to be compliant. Other profile names could potentially be defined but are out of scope for this document.
- ProfileVersion=<version> indicates that the object is intended to be compliant with the specified version of this document. This is REQUIRED for any information provider which intends to be compliant. Only one version can be published for a given object.

Further OtherInfo attributes specific to particular object types are defined with those objects.

Attribute	Req.	SD	SS	M	O	D	Validation
CreationTime	R					✓	ERROR if the value is more than one minute in the future; WARNING if more than two years in the past.
Validity	D					✓	ERROR if the validity period has expired, and FATAL if it is exceeded by a factor ten; WARNING if Validity is published and CreationTime is not.
ID	M	✓	✓	✓	✓	✓	FATAL if any two (distinct) objects have the same ID; ERROR if the ID format does not comply with the above rules.
Name	O			✓		✓	None.
OtherInfo	O	✓	✓	✓	✓	✓	Validation rules may be defined by the documentation for particular objects.
OtherInfo: InfoProviderHost	O					✓	None.
OtherInfo: InfoProviderName	D					✓	None.

OtherInfo: InfoProviderVersion	D					✓	None.
OtherInfo: ProfileName	M			✓	✓	✓	WARNING if the value is missing or anything other than “EGI”.
OtherInfo: ProfileVersion	M			✓	✓	✓	WARNING if the value is missing or unknown.

2.2 Extension

Extension objects serve a similar purpose to OtherInfo attributes in allowing the publication of information not otherwise available in the schema. In LDAP they are less convenient, since they incur the overhead of an additional object for each piece of information and make queries somewhat more complex, so in general OtherInfo is preferred, but either form is allowed. **Extension**s have the advantage of enforcing the Key=Value syntax, and should be considered for cases where new attributes are likely to be proposed as future additions to the schema. In such cases Keys should be recorded with the schema in a similar way to new values for enumerated types.

Attribute	Req.	SD	SS	M	O	D	Validation
LocalID	M	✓	✓	✓	✓	✓	FATAL if the value is not unique among **Extension**s of a single object.
Key	M	✓	✓	✓	✓	✓	INFO if the value is unknown. Documentation for the parent object may define further validation rules.
Value	M	✓	✓	✓	✓	✓	Known Keys may imply a syntax for the Value which can be validated.

2.3 Location

Location objects are used to describe the geographical location of a **Domain** or **Service**. Some **Domain**s are distributed over a number of physical sites so a **Location** may not represent a very well-defined place. **AdminDomain** objects representing Grid sites MUST have an associated **Location** object, and validation tools should raise a WARNING if not. **UserDomain** and **Service** objects MAY have a **Location** if desired. In any case, every **Location** object MUST have a relation to either a **Domain** or a **Service**. Note that the schema only allows one **Location** object per **Domain/Service**.

Due to limitations in the LDAP rendering of the schema, as well as the potential for problems in other applications, strings MUST NOT use non-ASCII characters.

2.3.1 Address

This specifies a postal address, in a free-text format. It is OPTIONAL, but if published it should be accurate.

2.3.2 Place

This gives the name of a town or city associated with the **Location**. If published it should be the name of somewhere reasonably associated with the **Domain** or **Service**, and likely to be known to human readers. It MAY include a comma-separated name of a state, region or county, e.g. “Atlanta, Georgia”.

2.3.3 Country

This gives the full name of the country or larger geographical region in which the **Domain** or **Service** is located. It is beyond the scope of this document to define the names and borders of countries, but national projects (NGIs) should agree on a uniform name, as sites may be grouped based on this name. Examples are “France”, “New Zealand”, “UK”.

2.3.4 PostCode

This specifies a postal code, in a free-text format. It is OPTIONAL, but if published it SHOULD be accurate.

2.3.5 Latitude, Longitude

These specify a position at which mapping tools should place a marker. For distributed **Domain**s or **Service**s the location may only be indicative.

Attribute	Req.	SD	SS	M	O	D	Validation
Address	O			✓			None.
Place	D			✓			INFO if the name is not identifiable, e.g. in a geographical database.
Country	R			✓	✓		INFO if the value is missing or unknown.
PostCode	O			✓			None.

Latitude	R			✓	✓		ERROR if a value is out of range or if only one of Latitude and Longitude are present; WARNING if the values are unlikely to be correct, e.g. in an ocean, at a pole or at (0,0); INFO if the location is inconsistent with the Place or Country or if the values are missing.
Longitude	R			✓	✓		

2.4 Contact

Contact objects define contact information for a **Domain** or **Service**. In EGI the primary source of contact information for sites is the GOC DB and for VOs is the Operations Portal, and it is therefore OPTIONAL to publish it in the information system, but if published it must be accurate. Every **Contact** object MUST have a relation to either a **Domain** or a **Service**.

2.4.1 Detail

This specifies the contact information. This usually consists of one or more email addresses, in which case the value SHOULD be a mailto: URL. Other forms of contact SHOULD use an appropriate URL format if defined.

Attribute	Req.	SD	SS	M	O	D	Validation
Detail	M			✓	✓		INFO if the format is not a valid mailto: URL or the information is inconsistent with the GOC DB.
Type	M			✓	✓		None.

2.5 Domain

The **Domain** class is abstract and cannot be instantiated itself, only in the form of the derived **AdminDomain** and **UserDomain** classes.

2.6 AdminDomain

In EGI the **AdminDomain** principally corresponds to the GOC DB and operational concept of a Grid site, in which case the DomainID MUST be the same as the site name in the GOC DB (and in the same case). Publication is REQUIRED and validation tools should raise an ERROR if the object is missing unless it can be determined that there is a temporary reason, for example that a corresponding site BDII is down. In general the DN structure in the site BDII will make it impossible to publish any service information for a site if the **AdminDomain** is missing. Publication via other technologies may have different restrictions, but a suitable **AdminDomain** object must still be available in the information system as a whole.

AdminDomains MAY also be published for groupings below or above the site level, e.g. for the components of a distributed site or for an NGI as a whole. The structure of the relations in the schema is such that lower-level **AdminDomain**s have a many-to-one relation to a higher-level **AdminDomain**, with the highest level by definition having no upward relation.

2.6.1 Description

A free-text description of the site, which may for example be displayed in a pop-up in a mapping tool. Any reasonable string is acceptable but it should be meaningful for a general reader. The string SHOULD be in English and MUST only use ASCII characters.

2.6.2 Distributed

This is a Boolean flag to indicate a geographically distributed site. If the attribute is omitted it implies a value of false. Distributed sites SHOULD set this flag; it may for example imply a different symbol for the site in a mapping tool.

2.6.3 Owner

This is intended to allow publication of the names of funding bodies or other organisations which request acknowledgement of their contribution. EGI does not mandate the format of this information, but may display it if published.

2.6.4 OtherInfo

A number of standard OtherInfo attributes are defined by EGI. Some are also defined by WLCG and documented here for convenience, but any validation is the responsibility of WLCG. See [R6] for further information; this relates to the GLUE 1.x GlueSite object but the principles are the same.

- BLOG=<URL> gives the URL of a blog related to the site. This key may appear multiple times.
- CONFIG=<config-tool> gives the name of the configuration tool(s) used at the site. This key may appear multiple times.
- EGI_NGI=<NGI-name> gives the name of the NGI to which the site belongs. This may for example be used to group or filter sites in display tools. This is REQUIRED for EGI sites and MUST correspond to the GOC DB entry.
- GRID=<Grid-name> gives the names of any Grids or projects to which the site belongs (one attribute per name). This may for example be used to group or filter sites in display tools. EGI sites MUST publish GRID=EGI. There is no registry for Grid names but projects should endeavour to ensure that names do not clash.

- <Grid-name>_<key>=<value>: Grid projects wishing to define their own site-based attributes SHOULD use their Grid name as a prefix.
- ICON=<URL> gives the URL of an image file which may be used to represent the site, typically 80*80 pixels. The file format SHOULD be one of PNG, GIF, JPG, ICO.
- OLDNAME=<site-name> gives the former name of a site which has changed its name. This key may appear multiple times.
- WLCG_NAME=<site-name> gives the site name as used by WLCG.
- WLCG_NAMEICON=<URL> gives the URL of an image file which may be used to represent the site for LCG purposes, typically 80*80 pixels.
- WLCG_PARENT=<site-name> gives the name of the parent site – for Tier-2 sites this is the associated Tier-1, for Tier-1 sites this is CERN.
- WLCG_TIER=<tier-number> gives the WLCG Tier of the site, as an integer from 0 to 4.

Attribute	Req.	SD	SS	M	O	D	Validation
Description	D			✓			None.
WWW	D			✓			INFO if the URL format is invalid.
Distributed	D			✓			None.
Owner	O			✓	✓		None.
OtherInfo: BLOG	O			✓			INFO if the URL format is invalid.
OtherInfo: CONFIG	D			✓	✓	✓	INFO if the value is unknown.
OtherInfo: EGI_NGI	M			✓	✓		WARNING if the value is invalid, or incorrect for the site.
OtherInfo: GRID	M			✓	✓		WARNING if EGI sites do not publish EGI as a GRID; INFO if the

							value is unknown.
OtherInfo: ICON	D			✓			INFO if more than one ICON is defined or the URL format is invalid.
OtherInfo: OLDNAME	O			✓	✓		None.

2.7 UserDomain

In the EGI context a **UserDomain** object would usually correspond to a VO. Currently EGI has no requirement for such objects, but if VOs want to do so they MAY negotiate with a site, typically one hosting their VOMS server, to publish such an object. The DomainID MUST be the VO name, or if sub-groups within the VO are published it MUST be prefixed with the VO name. As for **AdminDomain**s it is possible to publish a hierarchy of **UserDomain** objects.

2.7.1 Description

A free-text description of the VO. Any reasonable string is acceptable but it should be meaningful for a general reader. The string SHOULD be in English and MUST only use ASCII characters.

2.7.2 Level

This is an integer representing the level in a hierarchy, with 0 being the VO itself and higher numbers representing lower levels. EGI has no expectation that this will be used, but VOs MAY publish it if desired.

2.7.3 UserManager

This attribute contains EndpointIDs for services managing the user information for the VO. In the EGI context these would usually be voms(-admin) endpoints.

2.7.4 Member

This attribute is defined to allow publication of individual VO members. In EGI this is unlikely to be desirable; VOs considering publication should consider the security implications given that the information system is world-readable, and also the impact on the object size.

Attribute	Req.	SD	SS	M	O	D	Validation

Description	D			✓	✓		None.
WWW	D			✓			INFO if the URL format is invalid.
Level	O			✓	✓		None.
UserManager	D	✓		✓	✓		INFO if the value does not correspond to a published **Endpoint** object of a suitable type.
Member	U						INFO if this attribute is published.

2.8 Service

This is the top-level object representing a Grid service, so every service which should be published at all MUST have a **Service** object. Validation tools should raise a WARNING if services are published which do not correspond to GOC DB entries, or vice versa for those service types which are expected to be published, unless run outside the context of the EGI production Grid, e.g. on a testbed Each **Service** MUST be related to exactly one **AdminDomain**. **Service**s MAY be related to other **Service**s; these need not relate to the same **AdminDomain**.

2.8.1 Capability

This is an open enumeration representing the capabilities of the service in a high-level way. All suitable values MUST be published; if no such value is available in the existing list, one or more new values should be defined and published.

2.8.2 Type

This specifies the type of service according to a defined list. Uses for this attribute are currently unclear, and at the time of writing the GLUE working group is in the process of defining a list of Types. For services with a single type of **Endpoint** it is recommended that the ServiceType should be the same as the EndpointInterfaceName unless there is a good reason for a different value.

2.8.3 QualityLevel

This allows the possibility to publish various levels of non-production services in the production system. The published value SHOULD default to “production” but it is desirable for middleware configuration tools to provide the option to set the other values. Standard service discovery tools should select only “production” services by default, but should allow other values to be specified.

2.8.4 Complexity

This is a string giving a compact summary of the other objects associated with the **Service**. Publication is useful to provide a cross-check on the correctness of publication.

2.8.5 OtherInfo

In addition to the standard ProfileName and ProfileVersion attributes used to assert compliance with this document, an additional attribute ProfileCompliance=ALL may be published to indicate that all component objects for this Service should be regarded as compliant, to avoid publishing the attributes individually for a large number of objects.

Attribute	Req.	SD	SS	M	O	D	Validation
Capability	M	✓		✓			WARNING if particular service types publish Capabilities which don't match the expected values.
Type	M	✓		✓	✓		INFO if the value seems incorrect, e.g. failing to match the **Endpoint** type(s).
QualityLevel	M	✓	✓	✓	✓	✓	INFO if the value is other than "production".
StatusInfo	O			✓			INFO if the URL format is invalid.
Complexity	D					✓	ERROR if the format is invalid, WARNING if the value is inconsistent with the other published objects.
OtherInfo: ProfileCompliance	O			✓	✓	✓	INFO if the value is not ALL.

2.9 Endpoint

This is the primary class for service discovery, and all network endpoints intended to be discoverable must therefore be published. There may however be rare cases where a service is not intended to be accessed via the network at all. Validation tools should therefore raise a WARNING if a **Service** has no related **Endpoint** unless the ServiceType is known to be an exception.

2.9.1 URL

This is the contact URL for the service, and it is therefore essential for it to be correct.

2.9.2 Capability

This is an open enumeration representing the capabilities of the endpoint in a high-level way. All suitable values MUST be published; if no such value is available in the existing list, one or more new values should be defined and published.

2.9.3 Technology

This identifies the technology used to implement the service. The only likely value for EGI is "webservice". Publication is OPTIONAL but if published it must be correct.

2.9.4 InterfaceName

This attribute defines the primary interface protocol supported by the **Endpoint**. It is generally equivalent to the GLUE 1 ServiceType attribute, and in most cases should be the same. It is vital for this to be correct for service discovery to function.

2.9.5 InterfaceVersion

This specifies which protocol version(s) the **Endpoint** supports. For most services currently in use this is not vital because in practice we usually only have one version of a protocol – a partial exception is SRM [R7], but version 1 of that protocol is now deprecated. However, it SHOULD be published correctly as far as possible. The schema defines the attribute as optional but it is RECOMMENDED to publish "1.0" as a minimal default if no version is defined.

2.9.6 InterfaceExtension

This allows additional protocol type/version information to be published as a formatted URI. Publication of such information is OPTIONAL, but if published the meaning must be defined in the documentation for the service or implementation which publishes it.

2.9.7 WSDL

This specifies the URLs of one or more documents which define the web service provided by the **Endpoint**. This is only relevant to web services. Publication is OPTIONAL unless the use of a particular service requires it.

2.9.8 SupportedProfile

This defines one or more profiles for the protocol(s) supported by the **Endpoint**. Publication of such information is OPTIONAL, but if published the meaning must be defined in the documentation for the service or implementation which publishes it.

2.9.9 Semantics

This specifies one or more URLs of human-readable documents, e.g. manuals.

2.9.10 Implementor

This specifies the project or other organisation responsible for the implementation. Projects should agree suitable names amongst themselves.

2.9.11 ImplementationName

This specifies the name of the implementation. Projects should agree suitable names amongst themselves.

2.9.12 ImplementationVersion

This specifies the internal version of the implementation, for example the version of the main rpm (or equivalent) providing the service. Implementors can define the version as they choose, but it should be adequate to allow tracking at least of major releases and preferably of all releases. The Version string MAY be a concatenation of the internal versions of several components.

2.9.13 QualityLevel

This allows the possibility to publish various levels of non-production services in the production system. The published value SHOULD default to "production" but it is desirable for middleware configuration tools to provide the option to set the other values. Service discovery tools should select only "production" endpoints by default, but should allow other values to be specified.

2.9.14 HealthState

This is a mandatory attribute giving a dynamic indication of whether the service is running correctly. Information providers should attempt to set this in a useful way, e.g. by checking that appropriate daemons are running, but are not expected to do an extensive test, and SHOULD default to "ok". The test should not take a significant amount of time, involve contacting external services or be likely to fail if the service is in fact running normally. Service discovery tools should select only "ok" endpoints by default, but should allow other values to be specified. It is desirable for configuration/management tools to enable a service administrator to set the state to "warning" to allow it to be temporarily excluded from being selected by default while still being available with an explicit selection.

2.9.15 HealthStateInfo

This is an optional human-readable explanation of the health state. Information providers should use best endeavours to publish something meaningful, especially in the case of a HealthState other than "ok". Care should be taken to avoid exposing any sensitive information. It is useful for configuration/management tools to enable a service administrator to set the text explicitly.

2.9.16 ServingState

This is a closed enumeration indicating the current behaviour of the service. Information providers should reflect the state according to the service concerned - the values were originally defined for batch queues, and for services without the concept of a queue "draining" and "queueing" may not be meaningful states. The published value SHOULD default to "production", and service discovery tools should select only "production" endpoints by default, but should allow other values to be specified.

The GLUE 1 analogue of this attribute is sometimes used to prevent discovery of services which are in fact in a "production" state, but in GLUE 2 the QualityLevel explicitly serves such a purpose.

2.9.17 StartTime

This is the time at which the underlying service instance was started, e.g. taken from the creation time of a pid or lock file. Information providers should use best efforts to publish a reasonable value. The attribute SHOULD be omitted if the service is not running, and the Unix epoch (1970-01-01T00:00:00Z) SHOULD be used as an error indicator if the information is unavailable.

2.9.18 IssuerCA

This is the DN (in the usual openssl format) of the CA which issued the host certificate, if any. Information providers SHOULD publish this if the endpoint presents a certificate for authentication. Service discovery tools may offer an option to filter services according to the CA, but this should not be the default. The DN can for example be extracted with the command: `openssl x509 -subject -noout -in /etc/grid-security/hostcert.pem | sed 's/^[^/]*//'`

2.9.19 TrustedCA

This is defined to be the list of CA DNs trusted by the service. Most EGI sites will trust the standard set of IGTF-approved CAs, and to avoid publishing a long list of DNs the reserved word "IGTF" SHOULD be used in this case. Other reserved words may be defined if necessary to cover specific configurations. Service discovery tools may offer an option to filter services according to the issuer of the client certificate, but this should not be the default.

2.9.20 DowntimeAnnounce, DowntimeStart, DowntimeEnd, DowntimeInfo

These attributes allow publication of the next scheduled downtime. In EGI downtimes are managed via the GOC DB, and at present there is no requirement to publish them in the information system, but this may change in the future. Service discovery tools may provide options to filter out endpoints with downtimes in the near future, but this should not be the default. If a Start time is published without an End time the implication is that the downtime is indefinite.

2.9.21 OtherInfo

Some standard attributes are defined:

- MiddlewareName=<project-name> identifies the middleware distribution from which the service is taken, e.g. EMI. This SHOULD be published as agreed for a given project.
- MiddlewareVersion=<version> is a distribution-specific version identifier for the middleware. This SHOULD NOT be published unless MiddlewareName is also published.
- OSName=<name> is the name of the Operating System used on the host on which the endpoint runs. The values SHOULD be the same as used for the OSName attribute in the ExecutionEnvironment.

- OSVersion=<version> is the version of the Operating System used on the host on which the endpoint runs. The values SHOULD be the same as used for the OSVersion attribute in the ExecutionEnvironment.

- HostDN=<DN> gives the DN of the certificate presented by the endpoint.

URL	Req.	SD	SS	M	O	D	Validation
URL	M	✓					The format should be validated as far as possible, based on the InterfaceName, with invalid values classed as ERROR, WARNING or INFO as appropriate.
Capability	R	✓		✓			WARNING if particular endpoint types publish Capabilities which seem incorrect.
Technology	O	✓		✓			WARNING if the value appears to be incorrect for a given InterfaceName.
InterfaceName	M	✓		✓	✓		The value should be verified as far as possible, e.g. against the information in the GOC DB. Violations should usually be classed as WARNING.
InterfaceVersion	R	✓		✓			INFO if the attribute is missing.
InterfaceExtension	O	✓		✓			The format may be validated against a documented specification for a particular service or implementation.

WSDL	O	✓					WARNING if the Technology is not “webservice” or if the format seems to be invalid.
SupportedProfile	O	✓					The format may be validated against a documented specification for a particular service or implementation.
Semantics	D			✓			INFO if the format seems to be invalid.
Implementor	D			✓	✓		INFO if the name is unknown.
ImplementationName	M			✓	✓		WARNING if the value is missing; INFO if the name is unknown.
ImplementationVersion	M			✓	✓	✓	WARNING if the value is missing; INFO if the format seems invalid for a given implementation.
QualityLevel	M	✓	✓	✓	✓	✓	INFO if the value is other than “production”.
HealthState	M	✓		✓		✓	INFO if the value is other than “ok”.
HealthStateInfo	D			✓		✓	None.
ServingState	M	✓		✓			INFO if the value is other than “production”.
StartTime	D			✓			ERROR if the value is more than one minute in the future; WARNING if it is more than two years in the past.
IssuerCA	R	✓				✓	ERROR if the value is badly-formatted; INFO if it is unknown.

TrustedCA	O	✓				✓	ERROR if the value is badly-formatted; INFO if it is unknown.
DowntimeAnnounce	O			✓			ERROR if the time is more than one minute in the future; WARNING if it is more than one year in the past or if there is an Announce time with no Start time.
DowntimeStart	O	✓		✓		✓	ERROR if the time is later than DowntimeEnd; WARNING if it is more than one year in the future or more than one year in the past.
DowntimeEnd	O	✓		✓		✓	ERROR if there is an End time with no Start time; WARNING if the time is more than one week in the past or more than one year in the future.
DowntimeInfo	O			✓		✓	None.
OtherInfo: MiddlewareName	R			✓	✓		INFO if the attribute is missing or the name is unknown.
OtherInfo: MiddlewareVersion	R			✓	✓	✓	INFO if the attribute is missing or the version is unknown.
OtherInfo: OSName	R			✓	✓	✓	INFO if the attribute is missing or the name is unknown.
OtherInfo: OSVersion	R			✓	✓	✓	INFO if the attribute is missing or the version is unknown.
OtherInfo: HostDN	D	✓				✓	ERROR if the format is invalid.

2.10 Share, Manager, Resource, Activity

The schema defines these as abstract classes, with the intention that concrete classes will be derived from them as needed. However, given the difficulty of updating the LDAP schema used in EGI these are defined there as real object classes which can be instantiated. The intention is that this will be used for prototyping of new derived classes which will be proposed to the GLUE working group for inclusion in a future revision of the schema. The putative new attributes should be defined using **Extension** objects with Keys equal to the intended attribute names, and should be clearly documented in a similar format to the schema definitions of the existing derived classes for computing and storage systems. Validation tools should be extended in a corresponding way.

2.11 Policy, AccessPolicy, MappingPolicy

The **Policy** class is abstract and cannot be instantiated itself, only the derived **AccessPolicy** and **MappingPolicy** classes. These have the same attributes, but different relations and semantics. **AccessPolicy** objects relate to **Endpoint**s, and define which client credentials are likely to be authorised to access the endpoint – "likely" because this is not a security decision point, the local policies on the service will always take precedence, but for service discovery to be useful the published policy should match the local decision in normal circumstances. Similarly, **MappingPolicy** objects relate to **Share**s, and indicate which parts of the underlying resource are likely to be usable for a client with given credentials – the detailed semantics will be service-specific. The union of all mapping policies will often be the same as the union of all access policies for a given service, but this is not required.

Policy objects have two attributes, a Scheme and a Rule. The Scheme is an enumerated type which defines the syntax and semantics of the Rules. The intention is that for a given **Endpoint** or **Share** there will be at most one **Policy** object per Scheme, but multiple Schemes may be published. If no Policy is published for a given Scheme then access/mapping is denied in the context of that Scheme, but may still be allowed by Rules in other Schemes. In general there is no requirement that the Rules for different Schemes should express exactly the same underlying policy, or even that the syntax should be capable of expressing identical semantics.

The objects also have a relation to one or more **UserDomain** objects. Note that the schema document marks this relation as mandatory but there is an erratum which makes it optional since e.g. Rules referring to individual DNs may not relate to any particular **UserDomain**. In EGI these SHOULD be a list of VO names to which the Rules relate, since these are used as the IDs of such objects. However, some rules may be unrelated to any specific **UserDomain**, e.g. if they specify an individual DN. In EGI there is no requirement that **UserDomain** objects actually exist. Service discovery tools may have an option to filter the results according to the VO of the client, and this may be the default, but if so it should be possible to remove the filter or specify a different VO or list of VOs.

One aspect of the semantics is forced by the structure of the schema, that the Rules are an unordered list. This is a non-trivial point since the security rules on the service may be evaluated as an ordered list, so the real decision algorithm may not be representable in GLUE (and also the real rule sets may be too long to publish in a practical way).

The general principle for clients to use this information is to apply a matching algorithm between each Rule and the credentials of the user. If any of the Rules match then the overall result is success (access/mapping is allowed), otherwise failure.

The schema document defines a "basic" Scheme sufficient for simple cases, and this is currently published by some EMI services. However, gLite services have for many years been using a more complex policy syntax for publication of the GLUE 1.x AccessControlBaseRule attributes, and this is carried over into GLUE 2 publication using a Scheme name of "org.glite.standard". Since this is not explicitly documented elsewhere the syntax for this Scheme is specified here. A Rule has one of the following formats:

- A DN (in the usual openssl string representation, e.g. /C=UK/O=eScience/OU=CLRC/L=RAL/CN=stephen burke) which matches against the client DN.
- The reserved word ALL which always matches - this implies no authorisation, but not necessarily no authentication.
- The reserved word NONE which never matches (note that **Policy** objects must have at least one Rule so this is effectively a placeholder value).
- Or has the form <type>:<string>, where the interpretation of <string> depends on <type>:
- VO:<vo-name> matches if the user is a member of VO <vo-name>.
- VOMS:<fqan> matches if the primary FQAN of the user (if any) is <fqan>. In EGEE there was a proposal (https://edms.cern.ch/document/887174/1) to extend this syntax to allow trailing wildcards, but so far it has not been implemented. EGI may wish to consider resurrecting it, but it would imply changes in the services as well as the information system.
- MYPROXY:<myproxy-rule> matches according to the rules defined for MyProxy configuration.

Any format unknown to the client must fail to match but not be faulted, hence new <type>s can be added without breaking backward compatibility. This implies that there can be no DENY (negative) Rules. All matching is case-sensitive as implied by the schema.

EGI does not mandate the use of any specific Scheme, but clearly the middleware must be suitably interoperable, and the choices must be clearly documented. There are broadly three approaches which could be followed:

- The middleware may converge on a single Scheme, with all services publishing only that scheme and all clients interpreting it.
- Services may each publish in a Scheme of their choice, in which case clients may need to interpret multiple Schemes.
- Services may publish in multiple Schemes as required, with clients using whichever Scheme is most convenient.

3 COMPUTING SERVICES

Computing services have a large and complex schema. Many of the attributes are carried over from GLUE 1 and in general are expected to have the same values and semantics in GLUE 2, but the structure of the classes is somewhat different. Developers of services and clients must co-operate to ensure a consistent interpretation.

3.1 Installed capacity

The total installed computing capacity of a site is an important metric. For computing services it can be measured using the **ExecutionEnvironment** objects and associated **Benchmark**s. In particular the LogicalCPUs attribute counts the total number of execution units (=cores, with or without hyperthreading) – this should be a static value, i.e. it should include capacity on worker nodes which are temporarily down, but not capacity which is permanently unavailable. This can be multiplied by the benchmark value to yield a measure of total computing power. If the maximum number of jobs allowed is less than the number of execution units that number should be used instead – the basic principle is that the product of LogicalCPUs and the benchmark value should give the maximum deliverable power for the system, assuming that the benchmark is representative of the jobs being run.

It is also desirable to have a measure of the resource share allocated to each VO. Sites are therefore requested to publish nominal VO shares in an OtherInfo attribute of the **ComputingManager** object (WLCG requires this to be published). These will typically be fairshare targets in the LRMS and there is no requirement for them to be absolutely enforced. The actual balance of jobs among VOs will of course also depend on how many jobs each VO submits and with what time profile.

3.2 ComputingService

This is an extension of the **Service** class, with some additional attributes to summarise the overall state.

3.2.1 StagingJobs

This is the total number of Grid jobs currently waiting pre- or post-execution for files to be staged in or out respectively. Publication is OPTIONAL, otherwise such jobs should be treated as Running.

3.2.2 SuspendedJobs

This is the total number of Grid jobs which have started execution but are currently suspended. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting.

3.2.3 PreLRMSWaitingJobs

This is the total number of jobs currently managed by the Grid service which have not yet been submitted to the underlying LRMS. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting or omitted from the count.

Attribute	Req.	SD	SS	M	O	D	Validation
TotalJobs	R			✓	✓		WARNING if the value is not the sum of RunningJobs, WaitingJobs, StagingJobs, SuspendedJobs and PreLRMSWaitingJobs; INFO if the attribute is missing.

RunningJobs	R			✓	✓		INFO if the value is missing or greater than 1 million.
WaitingJobs	R			✓	✓		INFO if the value is missing or greater than 1 million.
StagingJobs	O			✓			INFO if the value is greater than 1 million.
SuspendedJobs	O			✓			INFO if the value is greater than 1 million.
PreLRMSWaitingJobs	O			✓			INFO if the value is greater than 1 million.

3.3 *ComputingEndpoint*

This is an extension of the **Endpoint** class. Again it has some additional attributes to summarise the jobs submitted via the specific **Endpoint**. Depending on the technology it may or may not be straightforward to collect this information, so publication is OPTIONAL. There are also two extra attributes to define possible properties of the interface. Note that the way the schema is defined means that all **Endpoint**s related to a **ComputingService** have the **ComputingEndpoint** type even if they offer capabilities other than job submission and management. For such **Endpoint**s the additional attributes are not relevant.

3.3.1 Staging

This is a closed enumeration describing the staging capabilities of the **Endpoint**. Publication is OPTIONAL; if published the middleware documentation should define the precise semantics. Service discovery tools may offer the option to filter **Endpoint**s based on this attribute, but it should not be the default.

3.3.2 JobDescription

This is an open enumeration describing the job description language(s) understood by the **Endpoint**. Service discovery tools may offer the option to filter **Endpoint**s based on this attribute, but it should not be the default.

3.3.3 StagingJobs

This is the total number of jobs currently waiting pre- or post-execution for files to be staged which were submitted via this **Endpoint**. Publication is OPTIONAL, otherwise such jobs should be treated as Running.

3.3.4 SuspendedJobs

This is the total number of jobs which have started execution but are currently suspended which were submitted via this **Endpoint**. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting.

3.3.5 PreLRMSWaitingJobs

This is the total number of jobs which have not yet been submitted to the underlying LRMS which were submitted via this **Endpoint**. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting or omitted from the count.

Attribute	Req.	SD	SS	M	O	D	Validation
Staging	O	✓		✓			None.
JobDescription	O	✓		✓			INFO if the value seems inconsistent with the InterfaceName.
TotalJobs	O			✓			WARNING if the value is not the sum of RunningJobs, WaitingJobs, StagingJobs, SuspendedJobs and PreLRMSWaitingJobs.
RunningJobs	O			✓			INFO if the value is greater than 1 million.
WaitingJobs	O			✓			INFO if the value is greater than 1 million.
StagingJobs	O			✓			INFO if the value is greater than 1 million.
SuspendedJobs	O			✓			INFO if the value is greater than 1 million.
PreLRMSWaitingJobs	O			✓			INFO if the value is greater than 1 million.

3.4 *ComputingShare*

This class represents the properties of a scheduling class, i.e. a class of jobs with a common **MappingPolicy** which are scheduled in a uniform way by an LRMS. In a Grid context this will usually be a set of jobs submitted by users in a particular VO or a subgroup within the VO (information at the level of individual users SHOULD NOT be published). Depending on the LRMS this may or may not also represent grouping by batch queue. This broadly corresponds to the VOView object in GLUE 1. The associated **MappingPolicy** specifies which user groups are mapped to the **Share**. In general a given user may map to multiple **Share**s, but this is only useful to the extent that the submission interface is able to select a particular **Share**, e.g. using a queue name. It is also possible to publish **Share**s without an associated **MappingPolicy** to provide monitoring information, e.g. of all jobs in a queue regardless of VO. Implementations must document the way in which **Share**s are mapped to the LRMS.

Publishing **Share**s for subgroups within a VO may cause practical difficulties if the number of groups is large. EGEE had a proposal to use wildcards to simplify the publishing but so far this has not been implemented – see https://edms.cern.ch/document/887174/1 for a full discussion of the issues. This was in the context of GLUE 1 but the issues are the same for GLUE 2.

Note that in GLUE 1 the queue time limits are in minutes, but in GLUE 2 they are in seconds.

WLCG is currently discussing a mechanism to publish support for the scheduling of multicore jobs, i.e. jobs which will be allocated to multiple cores on a single worker node. This is likely to be via one or more OtherInfo attributes in the **ComputingShare**, but at the time of writing this is not finalised.

3.4.1 MappingQueue

This is the name of the associated queue in the LRMS, if any. It MUST be published if there is such a queue.

3.4.2 MaxWallTime

This is the wallclock time limit for single-slot jobs in this **Share**, read from the LRMS. This SHOULD be published; if there is no limit use the placeholder value (all nines).

3.4.3 MaxMultiSlotWallTime

This is the wallclock time limit for multi-slot jobs in this **Share**, read from the LRMS. This SHOULD be published unless multi-slot jobs are not allowed; if there is no limit use the placeholder value (all nines).

3.4.4 MinWallTime

This is the minimum wallclock time for jobs in this **Share**, if any, read from the LRMS. This SHOULD be published if there is such a limit, or 0 if not.

3.4.5 DefaultWallTime

This is the wallclock time limit which will apply if no explicit request is made for a different limit. Some Grid interfaces may not allow such a request, in which case this is the limit which will apply to all jobs. This MUST be published; if there is no limit use the placeholder value (all nines).

3.4.6 MaxCPUTime

This is the CPU time limit for single-slot jobs in this **Share**, read from the LRMS. This SHOULD be published; if there is no limit use the placeholder value (all nines).

3.4.7 MaxTotalCPUTime

This is the aggregate CPU time limit for multi-slot jobs in this **Share**, read from the LRMS. This SHOULD be published unless multi-slot jobs are not allowed; if there is no limit use the placeholder value (all nines).

3.4.8 MinCPUTime

This is the minimum CPU time for jobs in this **Share**, if any, read from the LRMS. This SHOULD be published if there is such a limit, or 0 if not.

3.4.9 DefaultCPUTime

This is the CPU time limit which will apply if no explicit request is made for a different limit. Some Grid interfaces may not allow such a request, in which case this is the limit which will apply to all jobs. This MUST be published; if there is no limit use the placeholder value (all nines).

3.4.10 MaxTotalJobs, MaxRunningJobs, MaxWaitingJobs

These are policy limits applied by the LRMS. Note that RunningJobs will often be unable to reach the limit due to other constraints on available job slots. These MUST be published if such a limit is set; if not the placeholder value (all nines) SHOULD be used.

3.4.11 MaxPreLRMSWaitingJobs

This SHOULD be published if the Grid interface imposes such a limit.

3.4.12 MaxUserRunningJobs

This is the limit on jobs from a single user. This SHOULD be published if such a limit is set; if not the value MUST be the same as MaxRunningJobs. If the limit is different for different users the lowest limit SHOULD be published.

3.4.13 MaxSlotsPerJob

This is the maximum number of slots a single job can occupy. Publication is REQUIRED. Any further restrictions, for example whether the slots correspond to cores on a single node, should be published in a way documented by the implementation.

3.4.14 SchedulingPolicy

Publication is OPTIONAL but should be accurate if published.

3.4.15 MaxMainMemory, GuaranteedMainMemory, MaxVirtualMemory, GuaranteedVirtualMemory

Memory limits are a complex area, and Grid sites may be configured in significantly different ways. The main purpose of publishing these attributes is to allow service discovery to locate services able to run jobs with specific memory requirements, and the published limit should therefore be conservative but realistic. Publication is REQUIRED; if there is no Guaranteed memory available or the existence of a guarantee is unknown the corresponding values MUST be zero. Note that the units are MB.

3.4.16 Preemption

Publication is OPTIONAL. If not published it is undefined as to whether jobs may be pre-empted.

3.4.17 ServingState

This is a closed enumeration indicating the current behaviour of the batch queue. The published value should default to "production", and service discovery tools should select only "production" shares by default, but should allow other values to be specified.

3.4.18 SuspendedJobs

This is the total number of jobs which have started execution but are currently suspended. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting.

3.4.19 LocalSuspendedJobs

This is the total number of jobs which have started execution but are currently suspended which were not submitted via the Grid interface. Publication is OPTIONAL, otherwise such jobs should be treated as LocalWaiting.

3.4.20 StagingJobs

This is the total number of jobs currently waiting pre- or post-execution for files to be staged. Publication is OPTIONAL, otherwise such jobs should be treated as Running.

3.4.21 PreLRMSWaitingJobs

This is the total number of jobs currently managed by the Grid service which have not yet been submitted to the underlying LRMS but belong to this **Share**. Publication is OPTIONAL, otherwise such jobs should be treated as Waiting or omitted from the count.

3.4.22 EstimatedAverageWaitingTime, EstimatedWorstWaitingTime

There is a standard algorithm (lcg-info-dynamic-scheduler-generic [R8]) independent of the LRMS which SHOULD be used to calculate these values.

3.4.23 ReservationPolicy

Reservations are not currently supported in EGI. Any use of this attribute must be documented by the implementation.

3.4.24 Tag

There is currently no specific use for this attribute, but it would be useful to have some mechanism to configure it.

3.4.25 OtherInfo

By analogy with the MaxUserRunningJobs attribute, the Computational Chemistry VO has requested publication of the following additional attributes if the LRMS has a corresponding limit:

- MaxUserWaitingJobs=<limit> is the maximum number of queued jobs for a single user. If the limit is different for different users the lowest limit SHOULD be published.
- MaxUserTotalJobs=<limit> is the maximum number of jobs for a single user. If the limit is different for different users the lowest limit SHOULD be published.
- MaxSlotsPerUser=<limit> is the maximum number of job slots which can be occupied by jobs from a single user. If the limit is different for different users the lowest limit SHOULD be published.
- MaxNodesPerUser=<limit> is the maximum number of whole Worker Nodes which can be allocated to a single user. If the limit is different for different users the lowest limit SHOULD be published.

Attribute	Req.	SD	SS	M	O	D	Validation
Description	O			✓		✓	None.
MappingQueue	R	✓		✓		✓	INFO if the value is missing.
MaxWallTime	R	✓	✓	✓			INFO if the value is missing.
MaxMultiSlotWallTime	R	✓	✓	✓			INFO if the value is missing.
MinWallTime	R	✓	✓	✓			WARNING if the value is bigger than MaxWallTime; INFO if the value is missing.
DefaultWallTime	M	✓	✓	✓			WARNING if the value is missing or does not lie between MinWallTime and MaxWallTime.

MaxCPUTime	R	✓	✓	✓			INFO if the value is missing.
MaxTotalCPUTime	R	✓	✓	✓			INFO if the value is missing.
MinCPUTime	R	✓	✓	✓			WARNING if the value is bigger than MaxCPUTime; INFO if the value is missing.
DefaultCPUTime	M	✓	✓	✓			WARNING if the value is missing or does not lie between MinCPUTime and MaxCPUTime.
MaxTotalJobs	R	✓	✓	✓			WARNING if the value is zero or smaller than MaxRunningJobs or MaxWaitingJobs; INFO if the value is missing.
MaxRunningJobs	R	✓	✓	✓			WARNING if the value is zero; INFO if it is missing.
MaxWaitingJobs	R	✓	✓	✓			INFO if the value is missing or zero.
MaxPreLRMSWaitingJobs	O	✓	✓	✓			None.
MaxUserRunningJobs	R	✓	✓	✓			WARNING if the value is zero; INFO if it is missing or greater than MaxRunningJobs.
MaxSlotsPerJob	M	✓		✓			WARNING if the value is missing or zero.
MaxStageInStreams	O	✓		✓			None.
MaxStageOutStreams	O	✓		✓			None.
SchedulingPolicy	O	✓		✓			None.

MaxMainMemory	M	✓		✓			WARNING if the value is missing or less than 100; INFO if it is greater than 100,000.
GuaranteedMainMemory	M	✓		✓			WARNING if the value is missing or greater than MaxMemory; INFO if it is greater than 100,000.
MaxVirtualMemory	M	✓		✓			WARNING if the value is missing or less than MaxMainMemory; INFO if it is greater than 100,000.
GuaranteedVirtualMemory	M	✓		✓			WARNING if the value is missing or greater than MaxVirtualMemory; INFO if it is greater than 100,000.
MaxDiskSpace	O	✓		✓			None.
DefaultStorageService	O	✓					WARNING if the value is not the ID of a storage service.
Preemption	O	✓		✓			None.
ServingState	M	✓		✓			INFO if the value is other than "production".
TotalJobs	R		✓	✓			WARNING if the value is not the sum of the other job counts; INFO if the value is missing.
RunningJobs	R		✓	✓			INFO if the value is missing or greater than 1 million.
LocalRunningJobs	O			✓			INFO if the value is greater than 1 million.

WaitingJobs	R		✓	✓			INFO if the value is missing or greater than 1 million.
LocalWaitingJobs	O			✓			INFO if the value is greater than 1 million.
SuspendedJobs	O		✓	✓			INFO if the value is greater than 1 million.
LocalSuspendedJobs	O			✓			INFO if the value is greater than 1 million.
StagingJobs	O		✓	✓			INFO if the value is greater than 1 million.
PreLRMSWaitingJobs	O		✓	✓			INFO if the value is greater than 1 million.
EstimatedAverageWaitingTime	M		✓	✓			WARNING if the value is missing; INFO if it is greater than 1 million.
EstimatedWorstWaitingTime	M		✓	✓			WARNING if the value is missing; INFO if it is greater than 1 million.
FreeSlots	M		✓	✓			WARNING if the value is missing; INFO if it is greater than 1 million.
FreeSlotsWithDuration	O		✓	✓			None.
UsedSlots	M		✓	✓			WARNING if the value is missing; INFO if it is greater than 1 million.
RequestedSlots	R		✓	✓			INFO if the value is missing or greater than 1 million.
ReservationPolicy	O	✓		✓			None.

Tag	O	✓					None.
OtherInfo: MaxUserWaitingJobs	D	✓	✓	✓			WARNING if the value is zero; INFO if it is greater than MaxWaitingJobs.
OtherInfo: MaxUserTotalJobs	D	✓	✓	✓			WARNING if the value is zero; INFO if it is greater than MaxTotalJobs.
OtherInfo: MaxSlotsPerUser	D	✓	✓	✓			WARNING if the value is zero.
OtherInfo: MaxNodesPerUser	D	✓	✓	✓			WARNING if the value is zero.

3.5 *ComputingManager*

This class represents the LRMS as a whole. In most cases a **ComputingService** will have only one associated **ComputingManager**.

3.5.1 ProductName, ProductVersion

This is the name and overall version of the LRMS as defined by the vendor. Names for a given LRMS should be harmonised between implementations.

3.5.2 WorkingAreaShared, WorkingAreaGuaranteed, WorkingAreaTotal, WorkingAreaFree, WorkingAreaLifeTime, WorkingAreaMultiSlotTotal, WorkingAreaMultiSlotFree, WorkingAreaMultiSlotLifeTime, CacheTotal, CacheFree

At present there is no system to manage the working area (the disk space available to running jobs), but it is desirable for middleware configuration tools to allow system managers to set these values by hand.

3.5.3 OtherInfo

Attributes are defined to publish the per-VO resource share, and tags may also be published to identify specific features:

- Share=<VO>:<share>, where <VO> is a VO name and <share> is an integer representing the percentage share for that VO. This key may appear multiple times, but the sum of all shares MUST NOT exceed 100 and a given VO name MUST NOT appear more than once.

- CPUScalingReference<benchmark-name>=<value> implies that the LRMS uses internal scaling of job times to normalise to CPUs of the specified power.
- glexec is a logical flag indicating that the glexec mechanism may be used to switch the user identity for running jobs using suitably authorised credentials.

Attribute	Req.	SD	SS	M	O	D	Validation
ProductName	M			✓	✓		INFO if the value is unknown.
ProductVersion	M			✓	✓		INFO if the value seems invalid for the given ProductName.
Reservation	O	✓		✓			None.
BulkSubmission	O	✓		✓			None.
TotalPhysicalCPUs	M			✓	✓		WARNING if the value is missing or if there are more Physical than Logical CPUs; INFO if the value is less than 10 or more than 1 million.
TotalLogicalCPUs	M			✓	✓		WARNING if the value is missing; INFO if the value is less than 10 or more than 1 million.
TotalSlots	R			✓			INFO if the value is missing, less than 10 or more than 1 million, or more than twice TotalLogicalCPUs.
SlotsUsedByLocalJobs	O			✓			INFO if the value is more than TotalSlots.

SlotsUsedByGridJobs	R			✓			INFO if the value is missing or more than TotalSlots.
Homogeneous	O	✓					WARNING if the value seems inconsistent with the published **ExecutionEnvironment**s.
NetworkInfo	O	✓					None.
LogicalCPUDistribution	O	✓				✓	WARNING if the value seems inconsistent with the published **ExecutionEnvironment**s.
WorkingAreaShared	D	✓	✓	✓			None.
WorkingAreaGuaranteed	D	✓	✓	✓			None.
WorkingAreaTotal	D	✓	✓	✓			WARNING if the value is more than 1 million.
WorkingAreaFree	D	✓	✓	✓			WARNING if the value is more than 1 million.
WorkingAreaLifeTime	D	✓	✓	✓			None.
WorkingAreaMultiSlotTotal	D	✓	✓	✓			WARNING if the value is more than 1 million.
WorkingAreaMultiSlotFree	D	✓	✓	✓			WARNING if the value is more than 1 million.
WorkingAreaMultiSlotLifeTime	D	✓	✓	✓			None.
CacheTotal	D	✓	✓	✓			WARNING if the value is more than 1 million.
CacheFree	D	✓	✓	✓			WARNING if the value is more than 1 million.

TmpDir	O	✓					INFO if the format seems invalid.
ScratchDir	O	✓					INFO if the format seems invalid.
ApplicationDir	O	✓					INFO if the format seems invalid.
OtherInfo: Share	R			✓	✓		ERROR if the format is invalid, any VO name appears more than once, or the sum of all published shares exceeds 100; INFO if a VO name is unknown.
OtherInfo: CPUScalingReference	O	✓		✓	✓		ERROR if the format is invalid or if more than one such attribute is published; INFO if the value seems unphysical or the benchmark name is unknown.
OtherInfo: glexec	O	✓		✓			None.

3.6 Benchmark

This is a small class which allows publication of arbitrary CPU benchmark values. Middleware configuration tools should allow system managers to configure publication of arbitrary benchmarks. It is also possible to publish benchmarks using a dynamic information provider, but in most cases this is likely to be impractical.

The schema allows **Benchmark**s to be related to either or both of a **ComputingManager** or **ExecutionEnvironment** (only one of each). **Benchmark**s MUST have at least one of those relations present and it is an ERROR if both are missing. In addition, in EGI **Benchmark**s will usually be used to report the power of a specific **ExecutionEnvironment**, and it is therefore an INFO if a **Benchmark** does not have a relation to an **ExecutionEnvironment**.

3.6.1 Type

EGI requires publication of at least one benchmark Type, currently hep-spec06.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M	✓	✓	✓	✓		WARNING if the required Type(s) is/are missing.
Value	M	✓	✓	✓	✓		INFO if the value for a given Type seems unphysical.

3.7 *ExecutionEnvironment*

This class summarises the properties of a set of sufficiently homogeneous worker nodes. These are largely the same as the SubCluster attributes in GLUE 1 and are normally configured by hand by the system manager. It may be possible to collect the information dynamically, but past attempts have not been successful. It is left to the discretion of the system manager as to what constitutes "sufficiently homogeneous". Where the underlying nodes are not identical the published values should where possible be pessimistic, i.e. they should represent the minimum specification.

3.7.1 VirtualMachine

Publication is RECOMMENDED if the value is true, otherwise omission is equivalent to false. Support for virtualised environments may be a subject for further work.

Attribute	Req.	SD	SS	M	O	D	Validation
Platform	M	✓		✓			WARNING if the value is missing.
VirtualMachine	D	✓		✓			None.
TotalInstances	R			✓			INFO if the value is missing, less than 10 or greater than 1 million.
UsedInstances	O			✓			WARNING if the value is greater than TotalInstances.
UnavailableInstances	O			✓			WARNING if the value is greater than TotalInstances.

PhysicalCPUs	R	✓		✓	✓		INFO if the value is missing or greater than 10.
LogicalCPUs	R	✓		✓	✓		INFO if the value is missing or greater than 1000.
CPUMultiplicity	R	✓		✓	✓		ERROR if the value is inconsistent with PhysicalCPUs and LogicalCPUs.
CPUVendor	R	✓		✓		✓	INFO if the value is missing or appears not to be an expected name.
CPUModel	R	✓		✓		✓	INFO if the value is missing or appears not to be an expected name.
CPUVersion	R	✓		✓		✓	INFO if the value is missing or appears not to be an expected format.
CPUClockSpeed	R	✓		✓			INFO if the value is missing, less than 100 or greater than 10,000.
CPUTimeScalingFactor	M	✓		✓	✓		WARNING if the value is missing or greater than 1, and INFO if less than 0.1.
WallTimeScalingFactor	M	✓		✓	✓		WARNING if the value is missing or greater than 1, and INFO if less than 0.1.
MainMemorySize	M	✓		✓			WARNING if the value is missing; INFO if less than 100 or greater than 1 million.
VirtualMemorySize	M	✓		✓			WARNING if the value is missing or less than MainMemorySize; INFO if less than 100 or greater than 1 million.

OSFamily	M	✓		✓		✓	WARNING if the value is missing; INFO if it appears not to be a standard value.
OSName	M	✓		✓		✓	WARNING if the value is missing; INFO if it appears not to be a standard value.
OSVersion	M	✓		✓		✓	WARNING if the value is missing; INFO if it appears not to be a standard value.
ConnectivityIn	M	✓		✓		✓	None.
ConnectivityOut	M	✓		✓		✓	None.
NetworkInfo	O	✓					None.

3.8 ApplicationEnvironment

An **ApplicationEnvironment** represents a piece of installed software available to running jobs. It allows the publication of various properties and restrictions of the software, and also any setup commands which are required to make the software available. **ApplicationEnvironment**s have a mandatory relation to a **ComputingManager**. In addition they may be related to one or more **ExecutionEnvironment**s, implying that the software is only available on a subset of the Worker Nodes managed by the **ComputingManager**.

At present the only system which manages application environments in EGI uses a single tag string to identify software installed on worker nodes. This can either be defined directly by the system manager or published by authorised software managers in the VOs. In GLUE 1 this is published as the RunTimeEnvironment attribute in the SubCluster.

The most natural way for this to be mapped into GLUE 2 is to publish the tag string as the AppName attribute of an ApplicationEnvironment object. However, the total number of tags is currently very large (more than 7000 distinct values), so this may lead to an excessive data volume. If so it may be preferable to publish in a more compact way, for example with a single AppName per VO with the individual tags published as OtherInfo attributes.

The other attributes are currently unused and publication is OPTIONAL, but it would be useful to have a mechanism by which publication can be configured by a system manager on the request of a user community.

Attribute	Req.	SD	SS	M	O	D	Validation
AppName	M	✓		✓			None.
AppVersion	O	✓		✓			None.
Repository	O	✓					INFO if the URL format is invalid.
State	O	✓		✓			None.
RemovalDate	O	✓		✓			INFO if the date is in the past.
License	O	✓		✓			None.
Description	O			✓			None.
BestBenchmark	O	✓					INFO if the value does not match a published BenchMarkType.
ParallelSupport	O	✓		✓			None.
MaxSlots	O	✓	✓	✓			INFO if the value is zero.
MaxJobs	O	✓	✓	✓			INFO if the value is zero.
MaxUserSeats	O	✓	✓	✓			INFO if the value is zero.
FreeSlots	O		✓	✓			INFO if the value is greater than MaxSlots.
FreeJobs	O		✓	✓			INFO if the value is greater than MaxJobs.
FreeUserSeats	O		✓	✓			INFO if the value is greater than MaxUserSeats.

3.9 *ApplicationHandle*

Publication of this object is OPTIONAL, but it would be useful to have a mechanism by which publication can be configured by a system manager on the request of a user community.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M	✓					None.
Value	M	✓					INFO if the format is invalid for the given Type.

3.10 *ComputingActivity*

EGI currently has no requirements for publication of information about individual jobs, and the data volume is likely to be excessive if they were to be published, so these objects SHOULD NOT be inserted into the information system.

3.11 *ToStorageService*

The purpose of this class is to carry a relation to any **StorageService** where the file system is directly mounted on the worker nodes. This is one aspect of the "close SE" concept implemented with the CESEBind object in GLUE 1.

3.11.1 LocalPath, RemotePath

These define the mount point on the worker nodes and the corresponding export point on the storage system.

Attribute	Req.	SD	SS	M	O	D	Validation
LocalPath	M	✓					INFO if the format does not correspond to a filesystem path.
RemotePath	M	✓					INFO if the format does not correspond to a filesystem path.

4 STORAGE SERVICES

Storage services also have a complex schema. In GLUE 1 the structure of the original schema pre-dated the SRM protocol, although the 1.3 revision attempted to add extra information to aid its use.

For GLUE 2 the intention was to design for SRM from the start. Many of the attributes are carried over from GLUE 1 and in general are expected to have the same values and semantics in GLUE 2, but there are also many differences. In addition we have a number of widely-deployed implementations. Developers of services and clients must therefore co-operate to ensure a consistent interpretation.

There are now some use-cases for storage systems which do not involve the SRM protocol, and discussion is ongoing over how to publish and use such systems.

4.1 Installed capacity

For the measurement of installed storage capacity, the main object is to know the total size of installed disk and tape storage which is potentially usable by VOs. This should include storage which is temporarily unavailable due to servers being down, as well as storage which is not currently allocated to any VO, but not storage which is not directly usable, e.g. cache, RAID parity/mirror disks, hot spares, cold spares etc. It is also desirable to know how the space is allocated to the VOs and how much is currently in use.

The schema has the **StorageServiceCapacity** and **StorageShareCapacity** classes to enable publication of the storage capacity in a flexible way. In particular the Type attribute in each class is an open enumeration defining the meaning of the other attributes, with the "installedonline" and "installednearline" types being specifically intended for installed capacity publication. Objects with these Types MUST be published if the storage service supports the relevant type of storage medium – for EGI purposes "online" storage means disk and "nearline" means tape. For tape storage the size for installed capacity purposes is the actual size on the tape, after compression if applied. Within these objects the TotalSize is REQUIRED, configured by hand if there is no way to collect it dynamically. The other size attributes SHOULD be published unless the storage technology makes it impractical. If **StorageShare** objects represent overlapping portions of physical storage the installed capacity information MUST be published in a way which allows double-counting to be avoided.

In addition, the **DataStore** class allows publication of the total physical storage of a given type. This will usually be larger than the size reported in the Capacity objects as it includes storage which is not directly usable to store files, for example cache, parity disks and hot spares.

Capacity information is important, and developers should make their best efforts to publish as much information as possible. However, it is recognised that particular technologies may make it difficult to collect some kinds of information. It may be reasonable to publish estimated values if the accuracy is sufficient for the likely applications. In all cases implementations should document the meaning of the information they publish.

4.2 StorageService

This derived class has no additional attributes. The Service ID attribute has a strong persistency requirement since it may be used to locate stored files.

4.3 StorageServiceCapacity

This class represents the total storage managed by the system. The information is purely for monitoring and installed capacity measurement, so accuracy of a few % is sufficient. This SHOULD include all space which is potentially usable for the storage of user data, even if not currently writeable by any VO.

4.3.1 Type

Each **Capacity** object describes a particular kind of storage as defined by its Type. In EGI, publication of online and installedonline types is REQUIRED; nearline and installednearline are RECOMMENDED for systems which make use of nearline (tape) storage. Other types are OPTIONAL. The "installed" prefix indicates the inclusion of storage which is temporarily unavailable, e.g. due to disk servers being down. Hence validation tools should raise a WARNING if the "installed" values are smaller than the corresponding dynamic values.

4.3.2 TotalSize

This is defined to be the sum of FreeSize, UsedSize and ReservedSize, but there may be circumstances in which it is possible to publish the total but not to break it down.

4.3.3 FreeSize, UsedSize, ReservedSize

UsedSize is the size of all user files stored in the system, other than temporarily cached copies which can be automatically removed. For Types without the "installed" prefix this SHOULD exclude files stored on disk servers which are currently down. The remaining space, free for the storage of new files, is split according to whether the space is reserved for the use of a particular VO or subgroup. For SRM systems the ReservedSize SHOULD correspond to so-called "space tokens", with FreeSpace representing the remaining unreserved space.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M			✓	✓		None.
TotalSize	M			✓	✓		ERROR if the value is not the sum of FreeSize, UsedSize and ReservedSize if all values are published; WARNING if the value is missing; for "online" and "nearline" Types, INFO if the value is less than 1000 or greater than 1 billion.
FreeSize	R			✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing.
UsedSize	R			✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing.

ReservedSize	R			✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing.

4.4 *StorageAccessProtocol*

This publishes the file access protocols supported by the service. For SRM systems these should use the internal protocol type names as used e.g. in the getTURL method. The full list of protocols for the storage system SHOULD be published, whether accessible via the SRM interface or not.

4.4.1 Version

The protocol version is usually not very important, but should be accurate as far as possible.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M	✓	✓	✓			WARNING if there is no object with a Type of "gsiftp".
Version	M	✓		✓			INFO if the value appears wrong for the given Type.
MaxStreams	O	✓					None.

4.5 *StorageEndpoint*

This derived class has no additional attributes. All endpoints which may need to be discoverable MUST be published. For SRM systems this MUST include the SRM endpoint itself, and MAY also include file access endpoints if these are usable directly by clients, but not if they are only intended to be used via the SRM interface.

4.6 *StorageShare*

The **StorageShare** represents a portion of the storage managed by the service, and is a vital part of the published information. It may represent either a physical or logical division of the storage, and **StorageShare**s may overlap in terms of the storage they represent. **StorageShare**s MAY be published without an associated **MappingPolicy** to provide information for monitoring purposes. Implementations must document the mapping between the published information and the underlying organisation of the storage. However, as an overriding consideration the publication for all storage services used in EGI MUST allow interoperability – the primary client tools being GFAL/lcg-utils and FTS.

4.6.1 Description

This is a free text description of the purpose or nature of the **StorageShare**. If published it should be meaningful in the context of the storage system architecture.

4.6.2 ServingState

The published value SHOULD default to "production", but other values may be published according to need. Service discovery tools should select only "production" **StorageShare**s by default, but should allow other values to be specified.

4.6.3 Path

This is used as a prefix for file names, either physical or logical depending on the technology. Publication should be suitable for the semantics of the system. Clients should by default prefix user-supplied names with the Path if published, but should also allow the Path value to be overridden.

4.6.4 SharingID

This is an internal identifier to connect **StorageShare**s which represent the same underlying storage, to avoid double-counting. This MUST correctly identify overlapping **StorageShare**s.

4.6.5 AccessLatency, RetentionPolicy, ExpirationMode

These are defined to match the properties of spaces in the SRM protocol. For non-SRM systems they should be mapped as far as possible to the concepts of that system.

4.6.6 DefaultLifeTime, MaximumLifeTime

Currently in EGI all files are permanent so these attributes will usually be absent.

4.6.7 Tag

This is a user-defined name which in SRM usage corresponds to a space token description. It is RECOMMENDED to publish such names if defined.

Attribute	Req.	SD	SS	M	O	D	Validation
Description	D			✓			None.
ServingState	M	✓		✓			INFO for values other than "production".

Path	O	✓					Services may define service-specific validation rules.
AccessMode	O	✓		✓			None.
SharingID	M			✓	✓		WARNING if values other than “dedicated” exist in only one **StorageShare** in the **StorageService**.
AccessLatency	M	✓		✓			INFO if the value is “offline”.
RetentionPolicy	M	✓		✓			INFO if the value is other than “custodial” or “replica”.
ExpirationMode	M	✓		✓			INFO if the value is other than “neverexpire”.
DefaultLifeTime	O	✓					INFO if the value is less than 100,000.
MaximumLifeTime	O	✓					INFO if the value is less than 100,000.
Tag	D	✓		✓	✓		INFO if the value is not of a known format.

4.7 *StorageShareCapacity*

This has the same properties as **StorageServiceCapacity**, but reports values relevant only to the related **StorageShare**. The precise interpretation must be documented by the implementation as it may depend for example on whether a **Share** represents a physical or logical extent. The SharingID should be used to prevent double-counting of space. Validation tools should raise a WARNING if the sum of the sizes of (non-overlapping) **StorageShare**s exceeds the overall values published in the **StorageServiceCapacity**.

For installed capacity measurement, EGI requires sufficient information to determine the sharing of space between VOs, as represented by the **UserDomain** reference in the associated **MappingPolicy**. It SHOULD be possible to distinguish Reserved space, e.g. in space tokens, and space which is shared between multiple VOs. For the latter case it is Desirable for the UsedSpace to be split between VOs by publishing separate **StorageShare**s per VO, but it is recognised that this may not be practical, in which case monitoring tools may attribute all the used space to each VO.

4.7.1 Type

Each **Capacity** object describes a particular kind of storage as defined by its Type. In EGI, publication of online types is REQUIRED; nearline is RECOMMENDED for systems which make use of nearline (tape) storage. Other types are OPTIONAL. The "installed" prefix indicates the inclusion of storage which is temporarily unavailable, e.g. due to disk servers being down. Hence validation tools should raise a WARNING if the "installed" values are smaller than the corresponding dynamic values.

4.7.2 TotalSize

This is defined to be the sum of FreeSize, UsedSize and ReservedSize, but there may be circumstances in which it is possible to publish the total but not to break it down.

4.7.3 FreeSize, UsedSize, ReservedSize

UsedSize is the size of all user files stored in the **StorageShare**, other than temporarily cached copies which can be automatically removed. For Types without the "installed" prefix this SHOULD exclude files stored on disk servers which are currently down. The remaining space, free for the storage of new files, is split according to whether the space is reserved for the use of a particular VO or subgroup. For SRM systems the ReservedSize SHOULD correspond to so-called "space tokens", with FreeSpace representing the remaining unreserved space. At the **StorageShare** level it is therefore likely that either the FreeSpace or ReservedSpace will be zero.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M	✓		✓	✓		INFO if the published Types do not match the properties of the **StorageShare**.
TotalSize	M	✓	✓	✓	✓		ERROR if the value is not the sum of FreeSize, UsedSize and ReservedSize if all values are published; WARNING if the value is missing; for "online" and "nearline" Types, INFO if the value is less than 1000 or greater than 1 billion.

FreeSize	R	✓	✓	✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing or both FreeSize and ReservedSize are non-zero.
UsedSize	R	✓	✓	✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing.
ReservedSize	R	✓	✓	✓	✓		ERROR if the value is greater than TotalSize; INFO if the value is missing or both FreeSize and ReservedSize are non-zero.

4.8 StorageManager

For storage systems this simply publishes the name and version of the underlying management software. Publication is RECOMMENDED except where it simply duplicates information (ImplementationName and Version) available in one or more **Endpoint**s. The Version should be an overall version number for the product suitable for tracking at least major releases. For example, this may be the version of a key component, or a concatenation of the versions of several components.

Attribute	Req.	SD	SS	M	O	D	Validation
ProductName	M			✓	✓		INFO if the name is unknown.
ProductVersion	M			✓	✓		INFO if the version seems invalid for a given product.

4.9 DataStore

This class represents the physical hardware underling the service. The main purpose is to report the overall physical size of the system, which is useful for EGI management. Implementations should therefore make best efforts to publish this information. Validation tools should raise a WARNING if the published sizes are smaller than those published in the **StorageServiceCapacity**.

4.9.1 TotalSize

The total physical capacity. This MUST be published; if it is not possible to collect the information dynamically it may simply be configured by the system manager.

Attribute	Req.	SD	SS	M	O	D	Validation
Type	M			✓	✓		INFO if the value is not “disk” or “tape”.
Latency	M			✓	✓		INFO if the value is not “online” for “disk” or “nearline” for “tape”.
TotalSize	M			✓	✓		INFO if the size is less than 1000 or greater than 1 billion.
FreeSize	D			✓	✓		ERROR if the sum of FreeSize and UsedSize is greater than TotalSize.
UsedSize	D			✓	✓		ERROR if the sum of FreeSize and UsedSize is greater than TotalSize.

4.10 ToComputingService

This should be configurable by the system manager to advertise high-capacity network connections.

Attribute	Req.	SD	SS	M	O	D	Validation
NetworkInfo	M	✓		✓			None.
Bandwidth	M	✓		✓			INFO if the value is less than 100 or greater than 100,000.

5 APPENDIX: INFORMATION SERVICES IN EGI

5.1 BDII

The BDII [R4] is the primary information system in EGI, using the LDAP rendering of the schema. Clients should read information from a top-level BDII, which is normally configured to collect information from all certified sites (other configurations, e.g. including uncertified sites, are possible for special purposes). Grid-wide information is available under the root DN GLUE2GroupID=grid,o=glue. Top-level BDIIs may cache objects for a configurable amount of time. It is also possible that information collected from the lower-level sources may be removed or edited, for example to prevent malfunctioning services from being discovered.

Clients should not rely on the structure of the LDAP Directory Information Tree below the root DN, but should follow relationships between objects using their ID and ForeignKey attributes. However, information related to a particular Grid site is usually grouped under the **AdminDomain** object with GLUE2DomainID=<site-name>, and for special purposes may also be queried directly from the site BDII under GLUE2DomainID=<site-name>,o=glue.

Both top-level and site-level BDIIs are Grid services in their own right, and hence are published in the information system and are discoverable in the usual way. However, to bootstrap the system another method is needed. Site BDII URLs are stored in the GOC DB (see below), and this information is used to configure top-level BDIIs. In turn, clients generally use one or more top-level BDIIs defined in an environment variable (LCG_GFAL_INFOSYS). EGI, in co-ordination with WLCG, is currently discussing a more formal method to define which top-level BDIIs should be used.

Site BDIIs in turn collect information from "resource" BDIIs running on individual service nodes. However, these are regarded as site-internal services, hence they are not documented or published, and sites may choose to firewall them from external access.

5.2 GOC DB

The GOC DB [R9] is a central registry of sites and services, and historically served a substantially orthogonal set of use cases from the information system. In particular the information it contains is configured by hand by system managers using a web interface, and hence can only be slowly varying. However there is currently a plan for the GOC DB to use GLUE 2 concepts and structures where possible. To the extent that the BDII and GOC DB contain the same information, for example Latitude/Longitude and Contact details, it is clearly important to have cross-checks to ensure consistency.

5.3 EMIR

EMIR [R10] is a new service provided by EMI, which satisfies some of the same uses as both the BDII and GOC DB. The purpose is to provide a service registry, i.e. it targets the use cases marked as SD in this document. EGI is currently consulting with EMI on a deployment strategy to enable it to be tested in the production Grid, but at present it does not form part of the production infrastructure.

5.4 GLUE 2 Deployment

The LDAP rendering of the GLUE 2 schema has been integrated into the BDII since 2009, and information providers have been progressively implemented and deployed as part of the gLite and

EMI middleware releases. The EMI 3 release in 2013 is expected to have full GLUE 2 support. GLUE 2 is therefore ready to be used in 2013, although the GLUE 1 information continues in parallel and remains the default. The existing glue-validator tool [R4] is being extended to implement the validation described in this document, and the intention is to integrate it into the standard Nagios tests. Bugs will be submitted to developers and sites as necessary.

There have been a number of discussions about the future development of the information system technology. The GOC DB is migrating to GLUE 2, EMI have produced the EMIR service, and WLCG are also working to develop an information aggregation service. However, GLUE 2 is likely to remain as the information model whatever the technology, and the majority of the information in this document is technology-independent.

6 REFERENCES

R 1	http://www.ogf.org/documents/GFD.147.pdf
R 2	https://forge.ogf.org/sf/wiki/do/viewPage/projects.glue-wg/wiki/HomePage
R 3	https://github.com/OGF-GLUE/Enumerations
R 4	http://gridinfo.web.cern.ch/
R 5	https://twiki.cern.ch/twiki/pub/LCG/WLCGCommonComputingReadinessChallenges/WLCG_GlueSchemaUsage-1.8.pdf
R 6	https://wiki.egi.eu/wiki/MAN01
R 7	https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html
R 8	https://wiki.italiangrid.it/twiki/bin/view/CREAM/ERTAlgorithm
R 69	https://wiki.egi.eu/wiki/GOCDB_Documentation_Index
R 10	http://eu-emi.github.com/emiregistry/documentation/registry-1.1.1/emir-manual.pdf