GFD-R-P.xxx
Category: Recommendation
GLUE Working Group
http://forge.ogf.org/sf/projects/glue-wg

Authors:
Sergio Andreozzi*°, INFN
Stephen Burke, RAL
Felix Ehm, CERN
Laurence Field*, CERN
Gerson Galang, ARCS
Balazs Konya*, Lund University
Maarten Litmaath, CERN
Shiraz Memon, FZJ
Paul Millar, DESY
JP Navarro, ANL
Adrian Taga, Oslo University
*co-chairs
°editor

April 4, 2011

GLUE v. 2.0 – Reference Realization to XML Schema

Status of This Document

This document provides information to the Grid community regarding the realization of the GLUE information model (v.2.0) as XML Schema. Distribution is unlimited. This realization is derived from the proposed recommendation of the specification document [glue-2].

Copyright Notice

Trademark

Abstract

The GLUE specification is an information model for Grid entities described in natural language enriched with a graphical representation using UML Class Diagrams. This document presents a realization of this information model as XML Schema.

Contents

1. Introduction

The GLUE 2.0 Information model defined in [glue-2] is a conceptual model of Grid entities. In order to be adopted by technology providers, a realization in terms of a concrete data model is needed.

This document provides the normative realization of the GLUE 2.0 conceptual model in terms of an XML Schema (XSD). The document also elaborates on the design choices adopted to map the entities and relationships of the conceptual model into the concrete data model.

2. Notational Conventions

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).

3. XML Schema Realization

3.1 Approach

There are many approaches to map the GLUE conceptual model into an XML Schema. Depending on the aspects to be privileged, different design choices can be considered. When defining the proposed solution, we considered a number of best practices; we have maximized the component reuse for entities that are cohesive and coupled. We also exploited the hierarchical nature of an XML document to reflect "part-of" relationships to avoid the creation of extra-linking elements and improve the writing of XPath/XQuery statements.

3.1.1 Elements vs. Attributes

When defining data types in an XML Schema, two main options are available. A data can be described in terms of an XML Element or an XML Attribute. For the GLUE conceptual model, we have adopted the following strategy:

- Each UML class (or Entity) of the conceptual model maps into an XML Element definition.
- Each attribute of a UML class in the conceptual model maps into an XML Element definition (this is a general rule and applies also to both `ID` and `LocalID` attributes); an exception is made for the attributes `CreationTime` and `Validity` of the `Entity` class. Since they can be considered as metadata about GLUE-based description of entities, they are modeled as XML attributes.
- If a class or an attribute can be instantiated multiple times, then a separate XML Element for each instance MUST be created.

As additional information, it should be noted that:

- Attributes which type is a timestamp are typed using `glue:DateTime` which is a restriction of the `xsd:dateTime` simple type to match the UTC Timezone: `yyyy'-'mm'-'dd'T'hh':'mm':'ssZ`
- If an information producer cannot define a value for a mandatory attribute, then you SHOULD use the placeholder values defined (see Annex A in [glue-2])

3.1.2 Namespace

The Open Grid Forum published a document with guidelines for identifying names uniquely and uniform in the GGF/OGF domain [ogf-ns]. Based on this document, we have adopted the following namespace for the XML Schema realization of GLUE 2.0:

```
GLUE-XSD-NS ::= 'http://schemas.ogf.org/glue/' YYYY '/' MM '/spec_' M.N '_r' R
```

- `YYYY`: year of the normative document of the GLUE specification
- `MM`: month of the normative document of the GLUE specification
- `M.N`: M is the major version and N is the minor version of the GLUE conceptual model
- `R`: component to be used to specify the revision number of the XSD realization; this number SHOULD be incremented each time a new non-backwards compatible version is published

As a non-normative example, the namespace for the first release of the XSD document for the final GLUE 2.0 specification [glue-2] is:

```
http://schemas.ogf.org/glue/2009/03/spec_2.0_r1
```

3.1.3 Enumerations

The GLUE specification defines a set of attributes as enumeration. These enumerations belong to two main categories:

- Closed enumeration: a list of values is defined; the value of the attribute MUST belong to the set of defined values.
- Open enumeration: a list of values is defined; the value of the attribute MAY belong to the set of defined values. An open enumeration offers a partial list of values among which to choose. It also provides hints on how new values MAY be defined.

As regards the XSD realization, closed enumerations are modeled as restrictions on a base type. By using the element `<enumeration>`, each allowed value can be defined. An element which type is a restricted string type in terms of a set of values is valid if and only if the value matches one of those defined. The following XSD fragment presents a definition of the enumeration for the `Endpoint.HealthState` attribute:

```
<simpleType name="EndpointHealthState_t">
        <restriction base="string">
                <enumeration value="ok"/>
                <enumeration value="warning"/>
                <enumeration value="critical"/>
                <enumeration value="unknown"/>
                <enumeration value="other"/>
        </restriction>
</simpleType>
```

Concerning the open enumeration, the natural approach would be to use the `union` capability of XSD [xsd-oe, xsd-ap]. Unfortunately, this is not well supported in current implementations of XML software libraries; therefore we decide to model them by using the `annotation` element. Each enumeration value is represented by an `appinfo` sub-element. A software validating an XML document according to the defined XSD for GLUE 2 SHOULD be instrumented in order to consider these values. The following XSD fragment presents the definition of the open enumeration for the `DataStore.Type` attribute:

```
<simpleType name="DataStoreType_t">
        <restriction base="string">
            <annotation>
                <appinfo>
                    <enumeration value="disk"/>
                    <enumeration value="optical"/>
                    <enumeration value="tape"/>
                </appinfo>
            </annotation>
        </restriction>
    </simpleType>
```

3.1.4 Associations

In the conceptual model, several associations are represented. They can be classified in terms of the multiplicity (one-to-one, one-to-many, many-to-many), in terms of the navigability (directed, undirected) or in terms of the association type (binary, aggregation, composition, association class).

When mapping the associations from the conceptual model to the XSD, we adopt the following rules:

- One-to-one: modeled by using the parent-child relationship between XML elements
  - E.g.: an `AdminDomain` class has a directed association to a `Location` class; this is represented as an `AdminDomain` element having a child `Location` element
- One-to-many: modeled by parent-child relationships; the “one” is the parent, while the “many” are the children
  - E.g.: a `Service` class has a one-to-many association to an `Endpoint` class; this is represented as a `Service` element having zero or more child `Endpoint` elements
  - If the class participating in the “many” side of the relationship participates also in other associations, then only one association can be mapped into a parent-child relationship
    - E.g., the `ComputingActivity` class participates in three associations on the “many” side; the design choice was to use the parent-child option to map the `ComputingEndpoint`–`ComputingActivity` while relying on the many-to-many approach to describe the other associations; the association modeled as parent-child is not represented in the `Associations` element (see below)
  - Many-to-many: XML schema does not provide a natural support to represent this kind of relationship; therefore we need to add dedicated elements to carry this information. These elements are called by the name of the referenced class with the suffix “ID” and are grouped together under the `Associations` element. For instance, the `ComputingShare` has a many-to-many association to the `ComputingEndpoint` and a many-to-many association to the `ExecutioEnvironment`. In this XSD rendering, this is represented as follows:

```
<ComputingShare>
        <ID>urn:share_id1</ID>
        <Associations>
            <ComputingEndpointID>urn:myendpoint1</ComputingEndpointID>
              <ComputingEndpointID>urn:myendpoint2</ComputingEndpointID>
              <ExecutionEnvironment>urn:execenv1</ExecutionEnvironment>
        </Associations>
</ComputingShare>
```

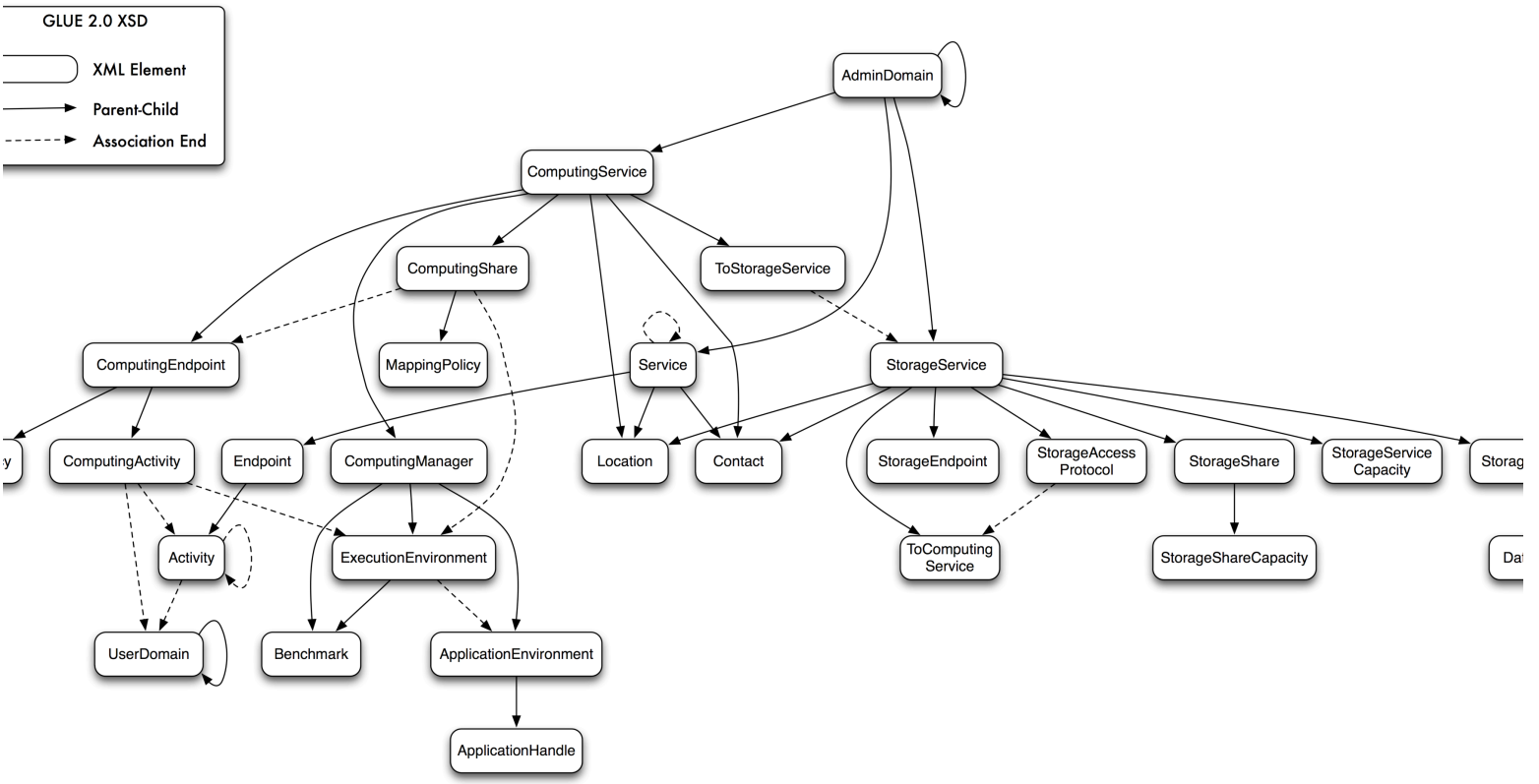
For each entity participating in a many-to-many relationship, we could add a reference element. Nevertheless, we represent the reference only in one side of the relationship to reduce inconsistencies and data to carry.

An exception to the above rules is made for the association `UserDomain-Policy`. This association is not represented in the XSD realization because the `Policy` entity implicitly encodes it through the `Rule` attribute. No exception is made for the associations `AccessPolicy-Endpoint` and `MappingPolicy-Share;` they follow the one-to-many relationship approach.

Table 1 lists which elements own the association reference for the many-to-many relationships and for the one-to-many relationships that are not represented as parent-child.

Table 1 Relationships and representation as Association Element (AE) or Parent-Child (PC)

Association End 1	Multiplicity	Association End 2	How	Owner/Parent
UserDomain	(1)←→(*)	UserDomain	PC	UserDomain
AdminDomain	(1)←→(*)	Service	PC	AdminDomain
AdminDomain	(1)←→(*)	AdminDomain	PC	AdminDomain
AdminDomain	(1)←→(*)	Location	PC	AdminDomain
AdminDomain	(1)←→(*)	Contact	PC	AdminDomain
Service	(1)←→(*)	Service	AE	Service
Service	(1)←→(*)	Endpoint	PC	Service
Endpoint	(1)←→(*)	Activity	PC	Endpoint
Endpoint	(1)←→(*)	AccessPolicy	PC	Endpoint
Activity	(1)←→(1)	UserDomain	AE	Activity
Activity	(1)←→(*)	Activity	AE	Activity
ComputingEndpoint	(*)←→(*)	ComputingShare	AE	ComputingShare
ComputingEndpoint	(1)←→(*)	AccessPolicy	PC	ComputingEndpoint
ComputingShare	(1)←→(*)	MappingPolicy	PC	ComputingShare
ExecutionEnvironment	(*)←→(*)	ComputingShare	AE	ComputingShare
ComputingActivity	(*)←→(1)	ComputingShare	AE	ComputingActivity
ComputingActivity	(*)←→(1)	ComputingEndpoint	PC	ComputingEndpoint
ComputingActivity	(*)←→(1)	ExecutionEnvironment	AE	ComputingActivity
ComputingActivity	(1)←→(1)	UserDomain	AE	ComputingActivity
ComputingActivity	(1)←→(*)	Activity	AE	ComputingActivity
ComputingService	(1)←→(*)	ComputingEndpoint	PC	ComputingService
ComputingService	(1)←→(*)	ComputingShare	PC	ComputingService
ComputingService	(1)←→(*)	ComputingManager	PC	ComputingService
ComputingService	(1)←→(*)	Service	AE	ComputingService
ComputingService	(1)←→(1)	ToStorageService	PC	ComputingService
ToStorageService	(1)←→(*)	StorageService	AE	ToStorageService
ExecutionEnvironment	(*)←→(*)	ApplicationEnvironment	AE	ApplicationEnvironment
ApplicationEnvironment	(1)←→(*)	ApplicationHandle	PC	ApplicationEnvironment
ComputingManager	(1)←→(*)	ExecutionEnvironment	PC	ComputingManager
ComputingManager	(1)←→(*)	ApplicationEnvironment	PC	ComputingManager
ComputingManager	(1)←→(*)	Benchmark	PC	ComputingManager
ExecutionEnvironment	(1)←→(*)	Benchmark	PC	ExecutionEnvironment
StorageService	(1)←→(*)	StorageEndpoint	PC	StorageService
StorageService	(1)←→(*)	StorageShare	PC	StorageService
StorageService	(1)←→(*)	StorageManager	PC	StorageService
StorageService	(1)←→(*)	StorageAccessProtocol	PC	StorageService
StorageService	(1)←→(*)	StorageServiceCapacity	PC	StorageService
StorageService	(1)←→(*)	ToComputingService	PC	StorageService
StorageAccessProtocol	(1)←→(*)	ToComputingService	AE	StorageAccessProtocol
StorageManager	(1)←→(*)	DataStore	PC	StorageManager
StorageEndpoint	(*)←→(*)	StorageShare	AE	StorageShare
StorageShare	(*)←→(*)	DataStore	AE	StorageShare
StorageShare	(1)←→(*)	StorageShareCapacity	AE	StorageShare
StorageService	(1)←→(*)	Service	AE	StorageService



"Parent-Child" and "Association End" relationships modeled in the GLUE XSD

3.1.5 Document Structure

XML documents are hierarchical with a single root element; therefore there MUST be a decision on what is the root element. In the GLUE conceptual model, `AdminDomain` and `UserDomain` cannot be grouped under a common entity. All other elements are publishable as descendent of `AdminDomain`. Therefore, we define a container XML Element called `Domains` to be the primary root element. The `Domains` root element MAY contain only instances of `AdminDomain` and `UserDomain`. The following XML fragment provides an example:

```
<Domains>
     <AdminDomain>
              <Service> …</Service>
             <ComputingService> …</ComputingService>
     </AdminDomain>
     <AdminDomain>
        <StorageService> …</StorageService>
     </AdminDomain>
     <UserDomain> … </UserDomain>
     <UserDomain> … </UserDomain>
</Domains>
```

In addition to the `Domains` root element, we also consider the introduction of other root elements useful for software components that need to generate valid XML document about only a part of the full model. For instance, an information producer MAY want to generate a description only about a `ComputingActivity` instance. Such an information producer SHOULD be able to validate the fragment document without necessarily instantiating the containing elements. Furthermore, adopters of the GLUE XSD rendering MAY need to reuse XML element definitions to embed GLUE information in other proprietary documents.

For both use cases, we allow the following elements to be valid root in a GLUE-compliant XML document:

- `Domains` (already defined above)
- `AdminDomain`
- `UserDomain`
- `Location`
- `Contact`
- `Service`
- `ComputingService`
- `StorageService`
- `ToComputingService`
- `StorageManager`
- `ComputingActivity`
- `ComputingActivities`
- `ExecutionEnvironment`
- `ApplicationEnvironment`

The choice of the root element should be agreed between the information provider and the information consumer (an example of this is the documentation in the WSDL). If you are unable to define an agreement between all the information consumers and the information provider, then you SHOULD use `Domains` as root element.

3.1.6 Grouping

Elements having siblings of the same type in the order of O(10) are grouped via grouping elements. For instance, a `ComputingEndpoint` MAY contain thousands `ComputingActivity` elements; these will be grouped via an intermediate `ComputingActivities` element. The following grouping elements are defined:

- Domains
- Activities
- ComputingActivities
- ExecutionEnvironments
- ApplicationEnvironments
- Extensions
- Associations

3.1.7 Inheritance

The GLUE conceptual model makes use of inheritance to capture common properties among entities. For instance both ComputingService and StorageService inherit from the Service class. As a valid use case, we consider the possibility of querying "all services" regardless their specialization. In order to simplify this type of query, we introduce an XML attribute called BaseType which value is fixed and equals to the name of the super-class. Such an attribute is defined for the entities as presented in Table 2.

Table 2 XML Elements having a fixed BaseType attribute value

BaseType attribute value	GLUE Entity
Domain	AdminDomain, UserDomain
Service	Service, ComputingService, StorageService
Endpoint	Endpoint, ComputingEndpoint, StorageEndpoint
Share	Share, ComputingShare, StorageShare
Manager	Manager, ComputingManager, StorageManager
Resource	Resource, ExecutionEnvironment, DataStore
Activity	Activity, ComputingActivity
Policy	Policy, AccessPolicy, MappingPolicy

3.1.8 Extensibility

At the conceptual level, the GLUE model defines two main "hooks" for extensions: the Extension class and the OtherInfo attribute (see Section 5.1 [glue-2]). In the XML Schema mapping, the Extension class is mapped as an Extension XML element in a parent-child relationship with the related class. The OtherInfo attribute is mapped as an OtherInfo XML Element. They are both available in all XML Elements for extensions

The above extension hooks are defined in the conceptual model. The XML Schema enables to add hooks for extensibility based on a flexible mechanism. Content models can be extended by any elements and attributes belonging to specified namespaces (i.e., we refer to the lax value for the processContent attribute of an xsd:any element definition). This option is adopted only for the Extensions element.

In the following example, we present a fragment showing how the three extensibility options can be used:

```
<ExecutionEnvironment BaseType=”Resource”>
     <ID>urn:myexecenv1</ID>
                  …
     <OtherInfo>This is a powerful GPU system</OtherInfo>
     <Extensions>
          <Extension>
              <LocalID>GeForge</LocalID>
              <Key>GeForge</Key>
              <Value>GeForge 7</Extension>
          </Extension>
          <Extension>
              <LocalID>CoreLib</LocalID>
              <Key>CoreLib</Key>
              <Value>glibc:3.4.9</Extension>
          </Extension>
              <typ:TextInfo xmlns:typ="">http://unigrids.org/2006/04/types">
                  <typ:Name>StagingInPath</typ:Name>
                  <typ:Value>/user-home/in</typ:Value>
              </typ:TextInfo>
              <typ:TextInfo xmlns:typ="">http://unigrids.org/2006/04/types">
                  <typ:Name>StagingOutPath</typ:Name>
                  <typ:Value>/user-home/out</typ:Value>
              </typ:TextInfo>
      </Extensions>
</ExecutionEnvironment>
```

3.2 The Normative XML Schema Realization of GLUE 2.0

https://github.com/OGF-GLUE/XSD/blob/master/schema/GLUE2.xsd

4. Security Considerations

Please refer to RFC 3552 (http://www.ietf.org/rfc/rfc3552.txt) for guidance on writing a security considerations section. This section is required in all documents, and should not just say “there are no security considerations.” Quoting from the RFC:

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

5. Author Information

Sergio Andreozzi
EGI.eu
Science Park 105
1098 XG Amsterdam
the Netherlands
sergio.andreozzi@egi.eu

Stephen Burke
Science and Technology Facilities Council
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Chilton, Didcot, Oxfordshire, OX11 0QX (UK)
E-mail: s.burke@rl.ac.uk

Felix Nikolaus Ehm
CERN
Route de Meyrin 385
CH-1211 Geneva 23 (Switzerland)
E-mail: Felix.Ehm@cern.ch

Laurence Field
CERN
Route de Meyrin 385
CH-1211 Geneva 23 (Switzerland)
E-mail: Laurence.Field@cern.ch

Gerson Galang,
Australian Research Collaboration Service (ARCS)
Carlton South, Victoria (Australia)
E-mail: gerson.sapac@gmail.com

Balazs Konya,
Department of Physics, Lund University,
Professorsgatan 1, Box 118,
SE-221 00 Lund (Sweden)
E-mail: balazs.konya@hep.lu.se

Maarten Litmaath
CERN
Route de Meyrin 385
CH-1211 Geneva 23 (Switzerland)
E-mail: Maarten.Litmaath@cern.ch

Shiraz Memon
Jülich Supercomputing Centre (JSC)
Wilhelm-Johnen-Straße
52425 Jülich, Germany
Email: a.memon@fz-juelich.de

Paul Millar,
Deutsches Elektronen-Synchrotron (DESY),
Notkestraße 85,
22607 Hamburg (Germany)
E-mail: paul.millar@desy.de

John-Paul Navarro
University of Chicago/Argonne National Laboratory
Mathematics & Computer Science Division, Building 221
9700 S. Cass Avenue
Argonne, IL 60439 (USA)
E-mail: navarro@mcs.anl.gov

Adrian Taga
Add address

6. Contributors & Acknowledgements

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

7. Intellectual Property Statement

8. Disclaimer

9. Full Copyright Notice

10. References

[glue-wg] The GLUE Working Group of OGF, *https://forge.gridforum.org/sf/projects/glue-wg*
[glue-usecases] GLUE 2.0 Use Cases (early draft), *https://forge.gridforum.org/sf/go/doc14621*

[glue-2] GLUE Specitication v. 2.0, OGF GFD.147, 3 Mar 2009,
http://www.ogf.org/documents/GFD.147
[ogf-ns] Standardised Namespaces for XML infosets in OGF.
http://www.ogf.org/documents/GFD.84.pdf
[xsd-oe] XForms 1.0. Open Enumeration.
http://www.w3.org/TR/2002/WD-xforms-20020118/slice6.html#model-using-openenum
[xsd-ap] Advanced XML Schema Patterns for Databinding Version 1.0
http://www.w3.org/TR/xmlschema-patterns-advanced/#group-Unions