GLUE2 XML Renderings

David Meredith

GLUE2: XSD Style, Flat or Nested

- Flat:
  - Entities are equal siblings listed in a global element bag.
  - <Associations> modelled using 1 method:
    - Element ID references (all associations defined as children of <Associations> elements).
- Nested:
  - Defines multiple Document Root elements.
  - <Associations> modelled using 2 methods:
    - Nesting (defines strong parent-child relationships)
    - Element ID references (a single entity can have many parents which cannot be modelled by nesting alone – GLUE2 is not a pure tree structure).

- Flat voted as preferred style at OGF 35 but...
- Not a complete consensus, some prefer nested.
- Hope to derive a style consensus/solution soon...

GLUE2: GOCDB Requirements

- Render results from Projection queries
  - Projection queries simply specify the entities you need to render when building a SELECT query (for SQL, you would normally specify fields/cols).

- GOCDB provides 18 projection style methods:
  - get_service_endpoint
  - get_ngi
  - get_site
  - get_contact
  - get_downtime
  - get_site_contacts ...

GLUE2: GOCDB Requirements

1.


```
results>
    <ROC ROC_NAME="NGI_UK">
        <ROCNAME>NGI_UK</ROCNAME>
        <MAIL_CONTACT>UKNGI-OPERATIONS@jiscma
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT USER_ID="5955046" PRIMARY_KE
            <FORENAME>John</FORENAME>
            <SURNAME>Kewley</SURNAME>
            <TITLE>Mr</TITLE>
            <DESCRIPTION/>
            <EMAIL>john.kewley@stfc.ac.uk</EM
            <TEL>+44 1925 603513</TEL>
            <WORKING_HOURS_START/>
            <WORKING_HOURS_END/>
            <CERTDN>/C=UK/O=eScience/OU=CLRC/
            <ROLE_NAME>NGI Operations Deputy
        </CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
        <CONTACT>
```


2.


```
results>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME ID="25994" PRIMARY_KEY="44479G0" CI
        <PRIMARY_KEY>44479G0</PRIMARY_KEY>
        <HOSTNAME>goc.egi.eu</HOSTNAME>
        <SERVICE_TYPE>egi.GOCDB</SERVICE_TYPE>
        <ENDPOINT>goc.egi.euegi.GOCDB</ENDPOINT>
        <HOSTED_BY>GRIDOPS-GOCDB</HOSTED_BY>
        <GOCDB_PORTAL_URL>https://goc.egi.eu/port
        <SEVERITY>WARNING</SEVERITY>
        <DESCRIPTION>Network disruptions to allow
        <INSERT_DATE>1299751740</INSERT_DATE>
        <START_DATE>1300176000</START_DATE>
        <END_DATE>1300190400</END_DATE>
        <FORMATED_START_DATE>2011-03-15 08:00</FO
        <FORMATED_END_DATE>2011-03-15 12:00</FORM
    </DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
    <DOWNTIME>
```


~ Consider 1000's of records = can produce large XML documents.

1) https://goc.egi.eu/gocdbpi/private/?method=get_roc_contacts&roc=NGI_UK
2) https://goc.egi.eu/gocdbpi/public/?method=get_downtime&topentity=GOCDB

Nested

```
<Domains>
    <AdminDomain BaseType="Domain">
        <ID>99876</ID>
        <WWW>http://ngs.ac.uk</WWW>
        ...
        <ComputingService BaseType="Service
            <ID>2341</ID>
            <Type>org.some.compute</Type>
            <QualityLevel>production</Quali
            <TotalJobs>434</TotalJobs>
            ...
            <ComputingEndpoint>
            <ComputingEndpoint>
            <ComputingEndpoint>
        </ComputingService>

        <StorageService BaseType="Service">
            <ID>2342</ID>
            <Type>org.some.compute</Type>
            <QualityLevel>production</Quali
            <StorageEndpoint>
            <StorageShare>
            <StorageManager>
        </StorageService>

        <Service BaseType="Service">
            <ID>2342</ID>
            <Type>org.srb.SRB3</Type>
            <QualityLevel>production</Quali
            <Endpoint>
        </Service>
    </AdminDomain>
</Domains>
```

(... detail elided)

1. Associations: uses both nesting + ID references (nesting can't do many parents). ✗
2. XSD enforces nested relationships. ✓
3. Easy doc traversal for many associations (i.e. XPath to select nested children rather than cross referencing) ✓
4. Can't project/select only the required entities without using multiple Doc roots. ✗
5. Redundant parent + sibling elements = bloated docs (consider 1000s of records). ✗
   - Could exclude optional siblings and optional parents which are redundant, but this is misleading (entities MUST always be rendered in full).

Flat XSD

```
<!-- Entities is the DOCUMENT ROOT ELEMENT -->
<element name="Entities" type="glue:ExtensibleEntit
 <annotation>
</element>
<complexType name="ExtensibleEntities_t">
    <sequence>
        <element ref="glue:Location" minOccurs="0" 
        <element ref="glue:Contact" minOccurs="0" m
        <!-- Abstract element references -->
        <element ref="glue:Domain" minOccurs="0" ma
        <element ref="glue:AbstractService" minOccu
        <element ref="glue:AbstractEndpoint" minOcc
        <element ref="glue:Share" minOccurs="0" max
        <element ref="glue:Manager" minOccurs="0" m
        <element ref="glue:Resource" minOccurs="0" 
        <element ref="glue:AbstractActivity" minOcc
        <element ref="glue:Policy" minOccurs="0" ma
        <!-- Concrete element references -->
        <element ref="glue:Benchmark" minOccurs="0"
        <element ref="glue:ApplicationEnvironment" 
        <element ref="glue:ToComputingService" minO
        <element ref="glue:ToStorageService" minOcc
        <element ref="glue:StorageAccessProtocol" m
        <element ref="glue:StorageServiceCapacity" 
        <element ref="glue:StorageShareCapacity" mi
        <element ref="glue:ApplicationHandle" minOc
    </sequence>
</complexType>

<element name="Location" type="glue:Location_t" />
<element name="Contact" type="glue:Contact_t" />
<element name="Domain" type="glue:Domain_t" abstrac
<element name="AdminDomain" type="glue:AdminDomain_
<element name="UserDomain" type="glue:UserDomain_t"
```

<Entities> is Document Root: (element bag that lists GLUE entities as siblings in a defined order).

Elements declared globally and referenced from within <Entities>.

GLUE elements can be imported into standalone in 3rd party XSD.

Flat

```
<glue:Entities
    xmlns:xsi='http://www.w3.or
    xmlns:glue='http://schemas.
    xsi:schemaLocation='http://
    <glue:Location>
    <glue:Contact>
    <glue:Contact>
    <glue:UserDomain>
    <glue:AdminDomain>
    <glue:AdminDomain>
    <glue:Service>
    <glue:ComputingService>
    <glue:StorageService>
    <glue:Endpoint>
    <glue:ComputingEndpoint>
    <glue:StorageEndpoint>
    <glue:ComputingShare>
    <glue:StorageShare>
    <glue:ComputingManager>
    <glue:StorageManager>
    <glue:DataStore>
    <glue:ExecutionEnvironment>
    <glue:Activity>
    <glue:ComputingActivity>
    <glue:AccessPolicy>
    <glue:MappingPolicy>
</glue:Entities>
```

(elements are collapsed)

1. Single Doc Root element (<Entities>).
2. Relationships modelled using one method; (ID references which caters for many-to-many parents – GLUE2 is not a pure tree structure!).
3. Weaker association (relationship is not enforced by XSD = extra coding effort to validate that a reference points to correct element).
4. Traversing associations requires sub-queries (cross referencing element IDs)
5. Supports Bi and Uni directional associations.
6. Efficient = project just the required entities (e.g. select all contacts, select all endpoints etc).

```
<glue:ComputingService BaseType="Service">
    <glue:ID>computingService1</glue:ID>
    <glue:Type></glue:Type>
    <glue:QualityLevel>production</glue:QualityLevel>
    <glue:Associations>
        <glue:ComputingEndpointID>computingEndpoint1</
    </glue:Associations>
</glue:ComputingService>

<glue:ComputingEndpoint BaseType="Endpoint">
    <glue:ID>computingEndpoint1</glue:ID>
    <glue:URL>uri://some.url.ac.uk/service</glue:URL>
    <glue:InterfaceName></glue:InterfaceName>
    <glue:QualityLevel>development</glue:QualityLevel>
    <glue:HealthState>ok</glue:HealthState>
    <glue:ServingState>production</glue:ServingState>
    <glue:Associations>
        <glue:ComputingServiceID>computingService1</gl
    </glue:Associations>
</glue:ComputingEndpoint>
```

Sample Flat Rendering
(projecting services and endpoints)


```xml
<?xml version="1.0" encoding="UTF-8"?>
 <glue:Entities
    xmlns:xsi='http://www.w3.org/2001/XM
    xmlns:glue='http://schemas.ogf.org/g
    xsi:schemaLocation='http://schemas.o
    <glue:ComputingService>
    <glue:ComputingService>
    <glue:ComputingService>
    <glue:ComputingService>
    <glue:ComputingService>
    <glue:ComputingService>
    <glue:StorageService>
    <glue:Endpoint BaseType="Endpoint">
        <ID>endpoint1</ID>
        <Extensions>
        <URL>uri://some.url.ac.uk/service</URL>
        <InterfaceName></InterfaceName>
        <QualityLevel>development</QualityLevel>
        <HealthState>ok</HealthState>
        <ServingState>production</ServingState>
        <DowntimeAnnounce>2012-03-29T12:30:00Z</Do
        <DowntimeStart>2012-04-29T12:30:00Z</Downt
        <DowntimeEnd>2012-05-29T12:30:00Z</Downtin
        <DowntimeInfo>We had a power outage!</Dowr
        <Associations>
    </glue:Endpoint>
    <glue:ComputingEndpoint>
    <glue:StorageEndpoint>
    <glue:StorageEndpoint>
    <glue:StorageEndpoint>
</glue:Entities>
```


- Can select/render (project) <u>just the required entities</u> under the same Doc root.
- Efficient: No redundant data (consider 1000s of records).
- When selecting multiple entities (e.g. ‘select * services, endpoints, Contacts for NGI_X’) its harder to traverse the associations in the results (lots of ID lookups).

Flat XSD

```
// Option: Element ID Referencing
<complexType name="ComputingService_t">
<complexContent>
    <extension base="glue:ServiceBase_t">
        <sequence>
            <element name="TotalJobs" type="unsignedInt" minOccurs="0" maxOccu
            <element name="RunningJobs" type="unsignedInt" minOccurs="0" maxOc
            <element name="WaitingJobs" type="unsignedInt" minOccurs="0" maxOc
            <element name="StagingJobs" type="unsignedInt" minOccurs="0" maxOc
            <element name="SuspendedJobs" type="unsignedInt" minOccurs="0" max
            <element name="PreLRMSWaitingJobs" type="unsignedInt" minOccurs="0
            <element name="Associations" minOccurs="1" maxOccurs="1">
                <complexType>
                    <sequence>
                        <element name="ComputingEndpointID" type="glue:ID_t" m
                        <element name="ComputingShareID" type="glue:LocalID_t"
                        <element name="ComputingManagerID" type="glue:ID_t" mi
                        <element name="StorageServiceID" type="glue:ID_t" minO
                        <element name="ContactID" type="glue:ID_t" minOccurs="
                        <element name="LocationID" type="glue:ID_t" minOccurs=
                        <element name="ServiceID" type="glue:ID_t" minOccurs="
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </extension>
</complexContent>
</complexType>
```

All associations are element ID references

ID References to endpoints

```
<glue:ComputingService BaseType="Service">
    <ID>computingServiceReferencedEndpoints</ID>
    <Type></Type>
    <QualityLevel>production</QualityLevel>
    <Associations>
        <ComputingEndpointID>computingEndpoint1</Com
        <ComputingEndpointID>computingEndpoint3</Com
    </Associations>
</glue:ComputingService>
```

Sample XML

Nested XSD

```
// Option: Nested
<complexType name="ComputingService_t">
    <complexContent>
        <extension base="glue:ServiceBase_t">
            <sequence>
                <element name="TotalJobs" type="unsignedInt" minOccurs="0" max(
                <element name="RunningJobs" type="unsignedInt" minOccurs="0" ma
                <element name="WaitingJobs" type="unsignedInt" minOccurs="0" ma
                <element name="StagingJobs" type="unsignedInt" minOccurs="0" ma
                <element name="SuspendedJobs" type="unsignedInt" minOccurs="0"
                <element name="PreLRMSWaitingJobs" type="unsignedInt" minOccurs
                <element name="Associations" minOccurs="1" maxOccurs="1">
                    <complexType>
                        <sequence>
                            <element ref="glue:ComputingEndpoint" minOccurs="0'
                            <element ref="glue:ComputingShare" minOccurs="0" ma
                            <element ref="glue:ComputingManager" minOccurs="0"
                            <element ref="glue:StorageService" minOccurs="0" ma
                            <element ref="glue:ToStorageService" minOccurs="0"
                            <element ref="glue:Contact" minOccurs="0" maxOccurs
                            <element ref="glue:Location" minOccurs="0" maxOccu
                            <element name="ServiceID" type="glue:ID_t" minOccu
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

All associations are directly nested

Sample XML

```
<glue:ComputingService BaseType="Service">
    <ID>computingServiceReferencedEndpoints</ID>
    <Type></Type>
    <QualityLevel>production</QualityLevel>
    <Associations>
        <glue:ComputingEndpoint BaseType="Endpoint">
            <ID>computingEndpoint1</ID>
            <URL>uri://some.url.ac.uk/service</URL>
            <InterfaceName></InterfaceName>
            <QualityLevel>development</QualityLevel>
            <HealthState>ok</HealthState>
            <ServingState>production</ServingState>
            <Associations>
        </glue:ComputingEndpoint>
    </Associations>
</glue:ComputingService>
```

Inner (nested) endpoint s

Q. Should we consider a combined approach that enables a choice of nesting and/or element ID referencing according to the rendering requirements?

Combined Approach (Nested +Refs)

```
<complexType name="ComputingService_t">
    <complexContent>
        <extension base="glue:ServiceBase_t">
            <sequence>
                <element name="TotalJobs" type="unsignedInt" minOccurs="0" maxOccurs="1",
                <element name="RunningJobs" type="unsignedInt" minOccurs="0" maxOccurs="1
                <element name="WaitingJobs" type="unsignedInt" minOccurs="0" maxOccurs="1
                <element name="StagingJobs" type="unsignedInt" minOccurs="0" maxOccurs="1
                <element name="SuspendedJobs" type="unsignedInt" minOccurs="0" maxOccurs=
                <element name="PreLRMSWaitingJobs" type="unsignedInt" minOccurs="0" maxO
                <element name="Associations" minOccurs="1" maxOccurs="1">
                    <complexType>
                        <sequence>
                            <element name="ComputingEndpointID" type="glue:ID_t" minOccu
                            <element ref="glue:ComputingEndpoint" minOccurs="0" maxOccurs
                                <element name="ComputingShareID" type="glue:LocalID_t" m
                                <element ref="glue:ComputingShare" minOccurs="0" maxOccu
                            <element name="ComputingManagerID" type="glue:ID_t" minOccurs
                            <element ref="glue:ComputingManager" minOccurs="0" maxOccurs=
                                <element name="StorageServiceID" type="glue:ID_t" minOcc
                                <element ref="glue:StorageService" minOccurs="0" maxOccu
                            <element name="ContactID" type="glue:ID_t" minOccurs="0" max
                            <element ref="glue:Contact" minOccurs="0" maxOccurs="unbound
                                <element name="LocationID" type="glue:ID_t" minOccurs="0
                                <element ref="glue:Location" minOccurs="0" maxOccurs="unl
                            <element name="ServiceID" type="glue:ID_t" minOccurs="0" max
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Associations can be directly nested and/or referenced

Note, a nested '<Service>' association is not suitable here (thus only provide <ServiceID> option)

```
<glue:ComputingService BaseType="Service">
    <ID>computingServiceReferencedEndpoints</ID>
    <Type></Type>
    <QualityLevel>production</QualityLevel>
    <Associations>
        <ComputingEndpointID>computingEndpoint1</Comput
        <ComputingEndpointID>computingEndpoint3</Comput
        <glue:ComputingEndpoint BaseType="Endpoint">
            <ID>computingEndpoint1</ID>
            <URL>uri://some.url.ac.uk/service</URL>
            <InterfaceName></InterfaceName>
            <QualityLevel>development</QualityLevel>
            <HealthState>ok</HealthState>
            <ServingState>production</ServingState>
            <Associations>
                <ComputingServiceID>service2</Computir
            </Associations>
        </glue:ComputingEndpoint>
    </Associations>
</glue:ComputingService>
```

2 Referenced
+
1 Nested endpoints

Sample XML

Combined Approach

- Single XSD for both styles
- <Entities> as single + consistent Doc Root element.
  - <Child_Entity_Elements> can then nest their associated elements, or reference other entity elements to suit use cases, i.e.
    - Use Element refs to render projection queries
    - Use Nesting for other (eg XPath friendly) renderings

- Combined approach is not new (its quite common); e.g. Spring framework caters for both Inner Beans + Bean references in 'spring-beans.xml' in exact same way.

- Explicit Validation Rule:
  - MUST fail if a duplicate <Element> exists with same ID in doc.

- But, is this too flexible/complex?
  - Would need to be selective where we offer a choice of nested/idref ; in some associations, a choice is not necessary (see previous slide). Note, it is possible to refine the combined approach e.g. using <xsd:choice> if necessary.
  - Implementations would need to be clear if they require/support a particular style (a profiling requirement ?).