The aim of this topic is to describe which BDII information is taken into account and how that information is composed to create the classad representation insertend in ISM, namely used while the WMS performs the matchmaking.

The current gLite 3.1 version of the bdii purchaser acquires information about computing resources by querying the BDII for the following objectclasses:

- gluecesebind
- gluece
- gluevoview
- gluecluster
- gluesubcluster

A prototype version of the gLite 3.1 bdii purchaser has been patched in order to support gluelocation information (task 6650)

gluecesebind

Describes the association class between a computing element (CE) and storage elements (SE). The bind information is inserted in the classad representing the computing element as a list:

```
GlueCESEBindGroupSEUniqueID =
{
  "se01.somewhere.in_the.world",
  "se02.somewhere.in_the.world",
   ...
  "se_n.somewhere.in_the.world"
};
```

In order to require that the computing element eligible for executing the job at is bound to a given storage element, the user should use the classad `member` function while defining the requirement expression within the relevant JDL:

```
requirements = member(other.GlueCESEBindGroupSEUniqueID ,
"se01.somewhere.in_the.world");
```

gluecluster

A cluster is an aggregation entity for representing a complex computing resource in terms of the SubCluster and Computing Element entities. A cluster is a heterogeneous set of resources.

All the attributes beloging to this objectclass are flattened, mantaining the same name (GlueCluster*) into the classad representation of the relevant CE.

gluesubcluster

The SubCluster entity provides details of a homogeneous set of hosts as regards the selected attributes. Host entities within the SubCluster provide the set which describes the charactheristics of the resources belonging to the subcluster:

- ApplicationSoftware
- Architecture
- Benchmark
- MainMemory

- NetworkAdapter
- OperatingSystem
- Processor

All the attributes beloging to these objectclasses are flattened, mantaining the same name (GlueSubcluster*, GlueHost*) into the classad representation of the relevant CE. In order to match such attributes, at matchmaking time, the user should refers to them by symply adding the classad attribute reference `other.` followed by the name of the attribute as published in the objectclass it belongs to:

```
requirements = other.GlueHostMainMemoryRAMSize > 2048;
```

It should be reminded that an attribute defaults to multivalued unless it is explicitly made single-valued, that is, by adding the SINGLE-VALUE keyword after the SYNTAX section in the schema specification. Multivalued attributes such as GlueHostApplicationSoftwareRuntimeEnvironment are represented in classad as lists, and thus, in order to match a given value, the user should use the classad `member` function:

```
requirements =
member(other.GlueHostApplicationSoftwareRuntimeEnvironment,
"GLITE-3_1_0");
```

gluece

Provides a set of attributes which describe the static characteristics (GlueCEInfo*), a status changing frequently (GlueCEState*), a general use policy (GlueCEPolicy*), a set of authorized users or groups (GlueCEAccessControlBase*). The entities Info, State, Policy and AccessControlBase were present in the first GLUE Schema as individual objectclasses, then they have been merged into the gluece. For backward compatibility, at LDIF representation level, the entity name is included in the attribute name in order to enable the maintenance of a consistent mapping onto concrete data models.

All the attributes beloging to this objectclass are flattened, mantaining the same name (GlueCE*) into the classad representation of the relevant CE.

Again multivalued attributes such as GlueCEAccessControlBaseRule are represented in classad as lists, and thus, in order to match a given value, the user should use the classad `member` function:

```
requirements = member(other.GlueCEAccessControlBaseRule , "VO:cms");
```

gluevoview

The gluevoview entity was meant for reporting the state information specific to a group or a virtual organization. Basically the attributes published within this objectclass are the same within gluece but assigned with different values.

For a given computing element all the VOView-related information is processed. In order to maintain the same space of authorization rules, the splitting of the classad representation of the CE into its "views" is performed by computing the set intersection of the GlueCEAccessControlBaseRules information:

```
SET_INTERSECTION(<CE>.GlueCEACBR, <View i>.GlueCEACBR)
```

Any ACBR which has not been mapped is inserted within an additional computing element classad in ISM.

All the attributes beloging to this objectclass mantaining the same name (GlueCE*) into the classad representation of the relevant CE.

gluelocation

This entity has been added in order to provide a mechanism to describe what software packages are available in the worker nodes part of the SubCluster. Since the relation between SubCluster and its Locations is one-to-many the locations defined for a SubCluster are described in the classad representation of the ComputingElement bound to that Cluster/SubCluster as a list of classads:

```
SubClusterLocations =
{

 [
   GlueLocationName = "VO-atlas-production-13.0.30.3";
   GlueSchemaVersionMajor = 1;
   GlueLocationPath = "$VO_ATLAS_SW_DIR";
   GlueLocationLocalID = "VO-atlas-production-13.0.30.3";
   GlueSchemaVersionMinor = 2;
   GlueLocationVersion = "Prod";
   GlueChunkKey =
   {
    "GlueSubClusterUniqueID=agh2.atlas.unimelb.edu.au"
   };
  ]  ,
  [
   GlueLocationName = "VO-atlas-production-13.0.30.2";
   GlueSchemaVersionMajor = 1;
   GlueLocationPath = "$VO_ATLAS_SW_DIR";
   GlueLocationLocalID = "VO-atlas-production-13.0.30.2";
   GlueSchemaVersionMinor = 2;
   GlueLocationVersion = "Prod";
   GlueChunkKey =
   {
    "GlueSubClusterUniqueID=agh2.atlas.unimelb.edu.au"
   };
  ]
}
```

It should be reminded that the classad library does not provide any function to bind a list of ClassAds and test if either any or all ClassAds within such list match the specified constraints. Nevertheless, to such an aim it is possible to use the function provided by the WMS gang-matching extention: `anyMatch`, `allMatch` and `whichMatch`. The gang-matching functions test for a match by evaluating the constraints-expression in an evaluation environment that maps the pseudo-attribute `target` to the match candidate, thus, to require a ComputingElement having perl version 5.8 installed by using the Location entity defined for the subcluster one should type:

```
requiremts = anyMatch(
  other.SubClusterLocations,
  target.GlueLocationName == "perl"  &&  target.GlueLocationVersion == "5.8"
);
```

-- SalvatoreMonforte - 20 May 2008

This topic: EgeeJra1It > BDIIInformationAndClassadRepresentationOfACE
Topic revision: r2 - 2008-05-20 - SalvatoreMonforte