Jeremy Millar, University of Tennessee
Feb. 2001

Information Services
Request for Comments: GWD-GIS-008
Obsoletes: None
Category: Technical Specification - Draft

Grid Software Object Specification

1 Abstract

This document provides a formal specification for software objects in the Grid environment. The software deployed on a compute resource is at least as important as the available memory or processing power. Consequently, resource allocators for the Grid need to take this information into account. As an example, consider an application that requires the LAPACK libraries. Suppose two machines are available for the job, one with LAPACK installed, and one without. Given the application requirements, the resource allocator should choose the machine with LAPACK installed. This document defines baseline object classes to support that operation.

Contents

1 Abstract **1**

2 Introduction **1**

3 BIDM **2**

4 Grid Software Objects **2**
4.1 GridSoftware 2
4.1.1 GridSoftware GOS Binding 3
4.2 GridSoftwareDeployment 3
4.2.1 GridSoftwareDeployment GOS Binding 4

5 Examples **5**

6 Summary **5**

2 Introduction

In an important sense, the Grid is only as useful as the software deployed upon its nodes. Application software often requires specific libraries to function. Resource selection processes should take these requirements into account when selecting compute nodes on the Grid.

A simple approach to maintaining software deployment information for the Grid is to require all Grid nodes to maintain a standard suite of software. This approach requires Grid application developers to write applications that make use of only the approved software. Furthermore, software upgrades need to be applied globally and across multiple administrative domains. This approach works reasonably well for small Grid computing projects (e.g. GrADS), but is neither scalable nor desirable in the large.

A better approach is to store software deployment information for each Grid node in the Grid Information Services. Once this information is available, resource selection processes that make use of it can be designed.

This document provides a formal specification for software objects in the Grid Information Services environment. Object specifications are provided in the GOS [**?**] format.

The object specifications outlined in this document are derived from the Basic Interoperability Data Model (BIDM), IEEE Standard 1420.1 [**?**]. Building upon the BIDM allows the Grid Information Services to leverage the standard

and any tools that support it. Section 3 describes the BIDM. Section 4 presents specifications for *GridSoftware* and attendant classes in GOS format.

3 BIDM

The IEEE developed the BIDM in conjunction with the Reuse Library Interoperability Group (RIG) as a standard for encoding software metadata in an interoperable manner. The IEEE has extended the core BIDM to cover asset certification and intellectual property rights; these may also be of use to the Grid community. Additionally, the NHSE has extended the core BIDM to cover software deployment information.

BIDM metadata is either an attribute, or a relationship. Attributes describe objects, while relationships connect objects to one another. The BIDM defines five object classes: Asset, Element, Library, Organization, and RIG Object. Of these, Asset and Element are of most interest.

An Asset is defined as any reusable entity. Most often, this entity is software, although it might be documentation, music, or books. For instance, the metadata about LAPACK forms an Asset. Assets may also contain other Assets to indicate structural hierarchies. For instance, a RedHat Asset might contain Assets describing the Linux kernel, gcc, and the other components of a Linux system.

An Element is a discrete piece of an Asset. An Element might be source code, a makefile, man pages, or a library file. Each Asset may contain zero or more Elements.

Additionally, the NHSE has defined Machine and Deployment classes. The Machine class will likely be subsumed by [**?**]. The Deployment class primarily acts as a mapping between Assets and Machines. The Deployment, Asset, and Element classes should be extended to better support the Grid environment. These classes are discussed in detail in section 4.

A primary advantage of extending the BIDM to suit the needs of the Grid community is the leveraging of pre-existing tools and data. The NHSE has developed Repository In a Box (RIB) [**?**], a toolkit for maintaining interoperable online software catalogs. Combined with the NHSE extensions to the BIDM, RIB provides organizations with a simple way to catalog and display software deployment information. A large amount of data of value to the Grid is available in RIB repositories. Extending the BIDM provides a simple way to incorporate this data into the Grid.

4 Grid Software Objects

The object classes defined by the BIDM are not adequate for use in the Grid as they stand. Therefore, they have been extended to increase their utility. Additionally, they have been renamed to conform to extant Grid naming standards. Bindings for GOS are provided.

4.1 GridSoftware

The *GridSoftware* class is loosely equivalent to the BIDM Asset class. It represents a software package or library.

4.1.1 GridSoftware GOS Binding

```
GridSoftware OBJECT CLASS
OID                          1.3.6.1.4.1.6757.?
DESCRIPTION                  "A software package or library"
STRUCTURAL
KEY                          software                  ::single, cis
+                            version                   ::single, cis
MUST CONTAIN {
                             description               ::single, cis;
                             domain                    ::single, cis;
                             keywords                  ::multiple, cis;
                             documentation             ::multiple, cis;
}
MAY CONTAIN {
                             targetEnvironment         ::multiple, cis;
                             bugs                      ::multiple, cis;
                             notes                     ::multiple, cis;
                             contact                   ::single, cis;
                             language                  ::multiple, cis;
                             URL                       ::multiple, cis;
}
```

software: The name of the software.

version: The version number of the software.

description: A brief description of the software.

domain: The application area of the software, e.g. parallel tools.

keywords: A list of keywords describing the software for index purposes.

documentation: A URL pointing to documentation (FAQs, user guides, etc.) for the software.

targetEnvironment: The computer system, OS, and/or compiler required by the software.

bugs: A description of any know bugs in the software.

notes: Miscellaneous notes concerning the software.

contact: A point of contact for the software. This should point to a *GridPerson* or *GridOrganization*.

language: The language the software is expressed in, e.g. Ada, German, IDEF.

URL: A Uniform Resource Locator for more information about the software (e.g., company homepage, project website).

4.2 GridSoftwareDeployment

The *GridSoftwareDeployment* class is an expanded version of the NHSE Deployment extension to the BIDM. It represents an actual deployment of a software package on a compute resource.

4.2.1 GridSoftwareDeployment GOS Binding

```
GridSoftwareDeployment OBJECT
CLASS
OID                          1.3.6.1.4.1.6757.?.?.?
DESCRIPTION                  "A deployment of software on a
                             computer"
STRUCTURAL
KEY                          software                  ::single, cis
+                            hostname                  ::single, cis
MUST CONTAIN {
}
MAY CONTAIN {
                             dateOfDeployment          ::single, date;
                             deployedBy                ::single, cis;
                             administeredBy            ::single, cis;
                             binPath                   ::single, cis;
                             includePath               ::single, cis;
                             libPath                   ::single, cis;
                             manPath                   ::single, cis;
                             configuration             ::single, cis;
                             usageNotes                ::single, cis;
                             accessPort                ::single, cis;
                             controlPort               ::single, cis;
                             URL                       ::multiple, cis;
}
```

software: The software that is deployed. This should point to a *GridSoftware*.

hostname: The host the software is deployed on. This should point to a *GridComputeResource*.

dateOfDeployment: The date the software was deployed.

deployedBy: The person(s) responsible for deploying the software. This should point to a *GridPerson* or *GridOrganization*.

administeredBy: The person(s) responsible for maintaining this software deployment. This should point to a *GridPerson* or *GridOrganization*.

binPath: The path to the software's bin directory.

includePath: The path to the software's include directory.

libPath: The path to the software's lib directory.

manPath: The path to the software's man pages.

configuration: A brief description of configuration parameters used during the build process.

usageNotes: Notes describing the usage of this particular software deployment.

accessPort: The port to contact to access the software, e.g. 80 for Apache.

controlPort: The port to contact to control the software, e.g. 23 for FTP.

URL: A Uniform Resource Locator for more information about this software deployment.

5 Examples

This section provides several concrete examples of the GridSoftware and GridSoftwareDeployment objects. Examples are provided in LDIF format, although the object definitions are general enough to permit other representations. In fact, the RIB toolkit implements these objects in an SQL database.

```
dn: software=LAPACK, version=3.0, o=Grid
objectclass: GridSoftware
software: LAPACK
version: 3.0
description: Linear Algebra PACKage - provides FORTRAN routines for common linear algebra c
domain: Mathematical Software
keywords: linear algebra, mathematical software
documentation: http://www.netlib.org/lapack/
contact: lapack@cs.utk.edu
language: FORTRAN

dn: software=ATLAS, version=3.2, o=Grid
objectclass: GridSoftware
software: ATLAS
version: 3.2
description: Automatically Tuned Linear Algebra Subkernels - provides optimized BLAS routin
domain: Mathematical Software
keywords: BLAS, linear algebra, mathematical software
documentation: http://www.netlib.org/atlas/
contact: atlas@cs.utk.edu
language: C

dn: software=LAPACK, hostname=dinah.cs.utk.edu, o=Grid
objectclass: GridSoftwareDeployment
software: LAPACK
hostname: dinah.cs.utk.edu
deployedBy: Jeremy Millar
includePath: /usr/local/include
libPath: /usr/local/lib
manPath: /usr/local/man

dn: software=ATLAS, hostname=dinah.cs.utk.edu, o=Grid
objectclass: GridSoftwareDeployment
software: ATLAS
hostname: dinah.cs.utk.edu
deployedBy: Jeremy Millar
includePath: /usr/local/include
libPath: /usr/local/lib
manPath: /usr/local/man
usageNotes: Libraries should be linked in the order -llapack -lcblas -lf77blas -latlas
```

6 Summary

In summary, the GIS needs to provide software and software deployment information to support resource allocation. The software available on a resource is at least as important as its memory or processor speed. The object classes defined in Section 4 provide baseline support for this information.