MDS 2.2 GRIS Specification Document:
Creating New Information Providers

Introduction

MDS 2.2 includes a set of core GRIS information providers used to generate information such as platform type, host OS, system load, memory, file systems, etc. Core providers exist for use on Linux, Solaris, Irix, AIX, and Tru64. There are also generic providers available for platforms on which platform-specific core providers do not yet exist, so that basic MDS operability can be tested on those platforms.

A user can also create their own information providers to publish data into MDS. These providers can generate information such as host statistics, network status, storage or I/O details, application information, or literally anything that should be published in the Grid directory service.

The purpose of this document is to describe how a user can create their own information providers.

Document Overview

This document contains the following sections:

- GRIS Provider Overview
- Model of Operation
- Provider OIDs, Namespaces, and Schemas
- Related Documentation
- Appendix A. Example Core Provider Output
- Appendix B. Example grid-info-resource-ldif.conf File

Terminology

An information provider can also be referred to as a "sensor" or a "probe." For consistency, this document uses the term "provider" throughout. An "information provider," "GRIS provider," or "provider" all have the same meaning.

GRIS Provider Overview

This section presents definitions and basic functionality of core and user-created GRIS providers.

Definition

A GRIS provider is a program that gathers and generates data, and publishes it into MDS.

Core GRIS Providers

A set of core GRIS providers is included with MDS 2.2. These core providers and the data that they publish into MDS are described in the document *MDS 2.2 Core GRIS Providers*.

User-created Providers

It is relatively easy to create your own information providers. The essence of doing this involves three basic steps, as follows:

1. Decide what kind of information you want published into MDS. Then decide how that information should be represented in the Directory Information Tree (DIT). This requires the definition of a schema, an OID assignment, and naming conventions for that information.

2. Create a program that adheres to the input and output interface requirements described in this document. The program must be callable by the fork() and exec() facilities of the GRIS back end, and must return LDAP Data Interchange Format (LDIF) data objects according to the defined schema. An information provider can be written using whatever programming tools you prefer.

3. Enable the program by adding an entry for it in the grid-info-resource-ldif.conf file. The grid-info-resource-ldif.conf file defines all active GRIS providers, so when you create your own provider, a reference to it needs to be added to this file.

These steps are described in more detail in the *Model of Operation* section below.

Note that you cannot add information into MDS by using LDAP adds. You have to do it via an information provider, created by following the above steps.

Provider Examples

An example procedure for creating a provider is available at *Writing a Custom Information Provider in MDS 2.x From Start to Finish*.

A sample provider program is available from the *MDS Information Provider Example* page.

Model of Operation

This section first presents a brief overview of information provider operation, and then expands on the three basic steps:

1. Define provider schema, OID, and namespace.
2. Create a provider program.
3. Enable the provider program.

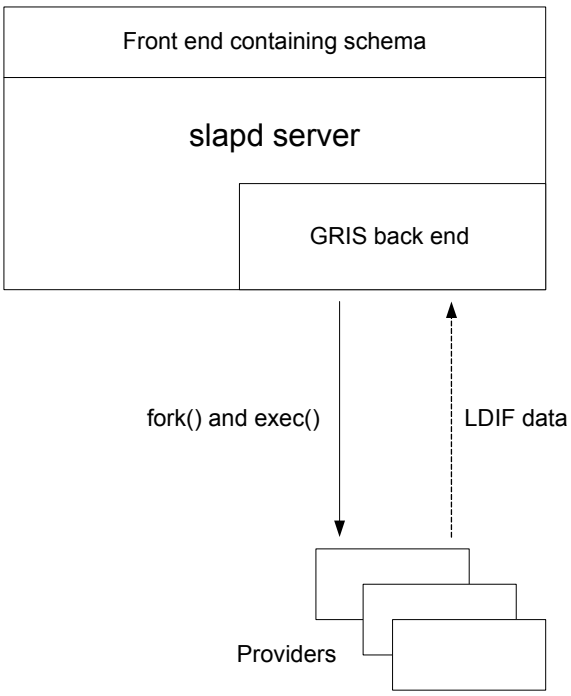
Overview of Information Provider Operation

A provider is any computer program written such that it adheres to two specific interfaces: the input and output interfaces of the GRIS back end. The GRIS back end is a module that extends the slapd server using the OpenLDAP *Generic modules API*. These interfaces need to be understood in order to write your own provider.

Refer to the *OpenLDAP Administrator's Guide* (http://www.openldap.org/devel/admin/guide.html) for more details.

The server front end contains schema appropriate to the information providers. The GRIS back end is cache hosted within the slapd server. The GRIS back end forks processes and execs provider programs, which then return LDIF data to the back end.

This process can be illustrated as shown below. The downward arrow from the GRIS back end to the Providers represent fork() and exec() operations, which invoke the Providers. Input to the Providers is a single LDIF record describing the incoming operation – add, delete, etc. Output from the Providers is a stream of LDIF records to the GRIS back end. For MDS, a line has been added to the interface to report the authenticated subject name of the client initiating the operation.



Define Provider Schema, OID, and Namespace

The LDIF output objects from the provider must match the schema in the slapd server front end, to ensure proper GRIS operation. An OID for each provider must exist in the schema for each class and attribute type.

The namespace used by ISI for MDS is an OID subspace registered with the Internet Assigned Numbers Authority (IANA). ISI uses this OID subspace to track and control OIDs for user-created information providers. To avoid OID and name collisions with other organizations, you should name your own providers and then acquire the names as described in the *Provider OIDs, Namespaces, and Schemas* section later in this document. You should choose a single prefix for all names created within your organization.

Create a Provider Program

The purpose of a user-created information provider is to get data from both MDS and the user, and put the data into an LDIF object. A provider can be created to publish specific data that the user requires into MDS.

Information Provider Input

The GRIS back end forks processes and executes a provider program. A user-created information provider must be callable by the fork() and exec() facilities of the GRIS back end to return LDIF data objects according to the defined schema. The input to the provider can be passed through a configuration file or an actual query at run time. There can be arbitrary command line options to the provider – they may not be standard. The provider can be anything that can speak to the interface.

Note that OpenLDAP has a shell back end that is not used by MDS. However, MDS follows the same input/output interface conventions as this shell back end when calling the provider programs specified in the GRIS back end configuration file (grid-info-resource-ldif.conf).

All of the core providers can be called from the GRIS back end with a command line argument.

The following example shows how to run the grid-info-cpufast-uptime core provider. You would cd to the proper path and then enter the grid-info-cpufast-uptime command as follows:

```
$GLOBUS_LOCATION/INSTALL/libexec

grid-info-cpufast-uptime -devclassobj -devobjs –dn
 Mds-Host-hn=dc-user.isi.edu,Mds-Vo-name=local,o=grid -ttl 60
```

See the next section for information on the output from this core provider.

Information Provider Output

A user-created information provider must generate LDIF objects as its output.
LDIF is a file format suitable for describing directory information or modifications made to directory information. LDIF is typically used to import and export directory information between LDAP-based directory servers, or to describe a set of changes that are to be applied to a directory.

LDIF format and syntax are described in detail in *The LDAP Data Interchange Format – Technical Specification* (RFC 2849: ftp://ftp.isi.edu/in-notes/rfc2849.txt).

The LDIF objects generated in MDS must be defined correctly with respect to the OID and namespace, as described in the *Provider OIDs, Namespaces, and Schemas* section later in this document.

The GRIS back end of the slapd server ensures that the LDIF objects returned conform to the search request. The data objects must also conform to LDIF syntax rules.

The grid-info-cpufast-uptime core provider in the example in the previous section rapidly generates the same output of grid-info-cpu-* by reading an inventory output from a cache file and updating only the "CPU free" attributes using system load-average information. This provides high-frequency load information at reduced cost. Example output from this provider appears in *Appendix A*.

Security Context for Information Providers

The security permissions a provider program has while it is executing are the same permissions that the slapd server has, since the GRIS back end is hosted in slapd, and the GRIS back end forks and execs the information provider. Thus if a custom information provider requires specific permissions, those permissions must also be granted to the slapd server.

Enable the Provider

The grid-info-resource-ldif.conf file lists active objects, which show how to invoke an information provider. The user adds their own objects to grid-info-resource-ldif.conf. The GRIS back end reads grid-info-resource-ldif.conf to get the path name (*path:* and *base:* parameters) and the arguments (*args:* parameter) of the information provider. Then the GRIS back end forks and execs the information provider.

With the grid-info-resource-ldif.conf file set up with the proper objects and names, the desired information is returned by the provider when you run the grid-info-search command.

Refer to *Appendix B* for an example of a grid-info-resource-ldif.conf file containing the core providers.

Provider OIDs, Namespaces, and Schemas

This section describes the concepts and issues behind OIDs, namespaces, and schemas required for GRIS providers, and presents recommendations for OIDs and naming.

Concepts/Issues

For proper GRIS operation, LDIF output objects must match the schema in the slapd server front end. Nonmatching data is suppressed by slapd.

Where the provider fits into the GRIS namespace depends on schema for the provider.

Each provider is configured/defined with an LDIF data block. This is the block in the configuration file that enables a GRIS provider. The provider is described in the grid-info-resource-ldif.conf file, as shown in the example in *Appendix B* of this document.

Where the provider resides in the DIT depends on the schema and namespace used for the provider.

If the provider generates information that doesn't fit into the subtree, the information is removed.

An OID for each provider must exist in the schema for each class and attribute type. The attribute name and class name have aliases and can use their own prefix in a string.

Recommendations for OIDs and Naming

As the development of user-created information providers expands, it becomes increasingly important that the providers have unique, nonconflicting OIDs and names. Without this, providers may collide with each other. ISI therefore recommends that a unique prefix (typically identifying your organization) must exist for each provider; the latter part of the name may be anything that you choose. You should choose a single prefix for all names created within your organization. Another reason for this type of unique prefix is that the LDAP community has nonprefixed programs; thus in creating a nonprefixed information provider, you risk a name collision with names of standard LDAP object classes or attribute types.

To guarantee avoidance of collisions, OID and naming assignments must be coordinated and controlled.

One way to coordinate these assignments is via a Private Enterprise Number (PEN) from the Internet Assigned Numbers Authority (IANA).

The Globus Project™ has registered a PEN with IANA. You can contact ISI to acquire a subtree of this OID space, or you can obtain your own PEN by applying directly to IANA at www.iana.org/cgi-bin/enterprise.pl.

ISI uses an OID subspace from IANA for MDS. ISI uses a segment of this subspace to track and control OIDs for user-created information providers. ISI will therefore not pick names for these providers, but will track the prefixes various organizations are using to ensure that no two organizations use the same prefix.

The OIDs of interest are:

1.3.6.1.4.1	IANA PEN space
1.3.6.1.4.1.3536.*	Globus OID subspace
1.3.6.1.4.1.3536.2.*	Globus Information Services OID subspace
1.3.6.1.4.1.3536.2.6.*	MDS OID subspace

Although we recommend that organizations obtain their own PENs from IANA, the Globus Project™ will, on request, provide subtrees of the MDS OID subspace to facilitate the development of providers and avoidance of collisions. For these assignments, contact the ISI OID naming authority at mds-oid-registrar@globus.org.

The MDS software includes a core schema using the reserved MDS OID subspace and the name prefix "MDS." For example, ISI uses mds-vo-name, mds-xxx, mds-yyy, etc. All names created at ISI are prefixed with "mds-". No one else should use this prefix or the 1.3.6.1.4.1.3536.2.6.* OID subspace, except MDS toolkit developers doing schema work.

The MDS schema for core information providers should not be extended to new providers. However, you can create an alternate provider for a new platform, which publishes data matching the MDS schema. In this case, you are porting an existing core provider to a new platform rather than creating a "new" provider as described in this document. Therefore, you can use the same MDS prefix, OIDs, classes, and attribute names as for the core provider.

To help you understand more about creating and modifying LDAP schemas in general, refer to www.openldap.org/doc/admin/schema.html. (Note, however, that this page does not directly discuss MDS schema or describe how to implement schema changes and/or additions with MDS.)

Ultimately, all OID and naming standards should be brought to an appropriate community-based standards organization such as a Global Grid Forum (GGF) working group. (Refer to www.gridforum.org for more information.)

Feedback on This Document

Please send any questions or comments on this document to:
mds-documentation@globus.org

Related Documentation

MDS 2.2 Object ID's (http://www.globus.org/mds/oid.html) summarizes the OID and naming recommendations described above.

Writing a Custom Information Provider in MDS 2.x From Start to Finish (http://www.thecodefactory.org/mds) presents an example procedure for creating a provider program.

MDS Information Provider Example (http://www-unix.mcs.anl.gov/~slang/mds_iprovider_example/index.html) presents a sample provider program that you can download and install.

MDS 2.2 New Features (http://www.globus.org/mds/NewFeatures.html) provides an overview of new and enhanced features in MDS 2.2.

MDS 2.2 Core GRIS Providers (http://www.globus.org/mds/DefaultGRISProviders.html) describes the core providers in detail.

MDS 2.2 GRIS: Adding New Information Providers (aka sensors) (http://www.globus.org/mds/GRIS_createnewcore.html) describes the creation of platform-specific core GRIS providers.

MDS 2.2 Test Suite (http://www.globus.org/mds/TestSuite.html) describes several types of testing for MDS 2.2, including the use of non-platform-specific providers.

MDS 2.2 Schemas (http://www.globus.org/mds/Schema.html) provides a complete listing of current MDS schema object classes, attribute types, and their definitions.

Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions (ftp://ftp.isi.edu/in-notes/rfc2252.txt) discusses LDAP attribute syntax in detail, and describes how users can create their own schemas.

The LDAP Data Interchange Format (LDIF) – Technical Specification (ftp://ftp.isi.edu/in-notes/rfc2849.txt) describes the syntax of the LDIF objects output by information providers.

OpenLDAP Administrator's Guide (http://www.openldap.org/devel/admin/guide.html) provides additional information on the slapd server and the API for the server back end.

Appendix A. Example Core Provider Output

This appendix presents example output from the grid-info-cpufast-uptime core information provider.

Entering the following at the command line:

```
grid-info-cpufast-uptime -devclassobj -devobjs –dn
 Mds-Host-hn=dc-user.isi.edu,Mds-Vo-name=local,o=grid -ttl 60
```

produces the following output:

```
dn: Mds-Device-Group-name=processors, Mds-Host-hn=dc-user.isi.edu,Mds-
Vo-name=local,o=grid
objectclass: MdsCpu
objectclass: MdsCpuSmp
objectclass: MdsCpuTotal
objectclass: MdsCpuCache
objectclass: MdsCpuFree
objectclass: MdsCpuTotalFree
objectclass: MdsDeviceGroup
Mds-Device-Group-name: processors
Mds-validfrom: 200111211918.25Z
Mds-validto: 200111211919.25Z
Mds-keepto: 200111211919.25Z
Mds-Cpu-Cache-l2kB: 256
Mds-Cpu-features: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 mmx fxsr sse
Mds-Cpu-Free-15min: 200
Mds-Cpu-Free-1min: 200
Mds-Cpu-Free-5min: 200
Mds-Cpu-model:  Pentium III (Coppermine)
Mds-Cpu-Smp-size: 2
Mds-Cpu-speedMHz: 997
Mds-Cpu-Total-count: 2
Mds-Cpu-Total-Free-15min: 200
Mds-Cpu-Total-Free-1min: 200
Mds-Cpu-Total-Free-5min: 200
Mds-Cpu-vendor:  GenuineIntel
Mds-Cpu-version: 6.8.6

dn: Mds-device-name=cpu 0, Mds-Device-Group-name=processors, Mds-Host-
hn=dc-user.isi.edu,Mds-Vo-name=local,o=grid
objectclass: MdsDevice
objectclass: MdsCpu
objectclass: MdsCpuCache
Mds-Device-name: cpu 0
Mds-Cpu-vendor:  GenuineIntel
Mds-Cpu-model:  Pentium III (Coppermine)
Mds-Cpu-version: 6.8.6
Mds-Cpu-features: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 mmx fxsr sse
Mds-Cpu-speedMHz: 997
Mds-Cpu-Cache-l2kB: 256
Mds-validfrom: 200111211918.25Z
Mds-validto: 200111211919.25Z
Mds-keepto: 200111211919.25Z
```

```
dn: Mds-device-name=cpu 1, Mds-Device-Group-name=processors, Mds-Host-
hn=dc-user.isi.edu,Mds-Vo-name=local,o=grid
objectclass: MdsDevice
objectclass: MdsCpu
objectclass: MdsCpuCache
Mds-Device-name: cpu 1
Mds-Cpu-vendor:  GenuineIntel
Mds-Cpu-model:  Pentium III (Coppermine)
Mds-Cpu-version: 6.8.6
Mds-Cpu-features: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 mmx fxsr sse
Mds-Cpu-speedMHz: 997
Mds-Cpu-Cache-l2kB: 256
Mds-validfrom: 200111211918.25Z
Mds-validto: 200111211919.25Z
Mds-keepto: 200111211919.25Z
```

Appendix B. Example grid-info-resource-ldif.conf File

This appendix presents a grid-info-resource.ldif.conf file containing the core GRIS providers, as follows:

```
# This file contains the core GRIS providers and must be
# configured for a particular platform to specialize the
# template...

# generate top-level Mds-Host-hn=host object every minute
dn: Mds-Host-hn=giis-demo.globus.org, Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-platform-merged
args: -dn Mds-Host-hn=giis-demo.globus.org, Mds-Vo-name=local, o=grid
-validto-secs 60 -keepto-secs 60
cachetime: 60
timelimit: 20
sizelimit: 1

# generate CPU availablity information every minute
dn: Mds-Device-Group-name=processors,
Mds-Host-hn=giis-demo.globus.org, Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-cpufast-uptime
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local, o=grid -validto-secs 60 -keepto-secs 60
cachetime: 60
timelimit: 20
sizelimit: 100
```

```
# generate CPU inventory (hidden cache) every 12 hours
dn: Mds-Device-Group-name=processors, Mds-Host-hn=giis-demo.globus.org,
Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-cpu-linux
args: -noobjs
cachetime: 43200
timelimit: 20
sizelimit: 1

# generate memory info every minute
dn: Mds-Device-Group-name=memory, Mds-Host-hn=giis-demo.globus.org,
Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-mem-linux
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local, o=grid -validto-secs 60 -keepto-secs 60
cachetime: 60
timelimit: 10
sizelimit: 3

# generate disk info every 15 minutes
dn: Mds-Device-Group-name=filesystems,
Mds-Host-hn=giis-demo.globus.org, Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-fs-posix
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local, o=grid -validto-secs 900 -keepto-secs 900
cachetime: 900
timelimit: 20
sizelimit: 20

# generate network info every 15 minutes
dn: Mds-Device-Group-name=networks, Mds-Host-hn=giis-demo.globus.org,
Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-net-linux
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local, o=grid -validto-secs 900 -keepto-secs 900
cachetime: 900
```

```
timelimit: 20
sizelimit: 20

# generate OS info every 12 hours
dn: Mds-Software-deployment=operating system, Mds-Host-hn=giis-
demo.globus.org, Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-os-uname
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local, o=grid -validto-secs 900 -keepto-secs 900
cachetime: 43200
timelimit: 20
sizelimit: 1

# generate GRIS info every 12 hours
dn: Mds-Software-deployment=MDS GRIS, Mds-Host-hn=giis-demo.globus.org,
Mds-Vo-name=local, o=grid
objectclass: GlobusTop
objectclass: GlobusActiveObject
objectclass: GlobusActiveSearch
type: exec
path: /testing/beta2.0/globus-install/libexec
base: grid-info-mds-core
args: -devclassobj -devobjs -dn Mds-Host-hn=giis-demo.globus.org, Mds-
Vo-name=local,o=grid -validto-secs 43200 -keepto-secs 43200
cachetime: 43200
timelimit: 20
sizelimit: 1

# generate GridFTP performance information every 1 day
# dn: Mds-Device-Group-name=performance, Mds-Host-hn=giis-
demo.globus.org, Mds-Vo-name=local, o=grid
# objectclass: GlobusTop
# objectclass: GlobusActiveObject
# objectclass: GlobusActiveSearch
# type: exec
# path: /testing/beta2.0/globus-install/libexec
# base: gridftp-perf-info
# args: -devclassobj -devobjs -dn Mds-Host-hn=giis-
# demo.globus.org,Mds-Vo-name=local,o=grid -validto-secs 86400
# -keepto-secs 86400
# cachetime: 86400
# timelimit: 100
# sizelimit: 100
```