MDS 2.2 User's Guide

Contents

Chapter 1. Introduction

This chapter provides an overview of the MDS 2.2 User's Guide.

Purpose of This Document

This MDS 2.2 User's Guide provides an overview of and introduction to the Monitoring and Discovery Service (MDS, Version 2.2) of the Globus Toolkit™ (Version 2.2).

This User's Guide describes the basics of MDS, including the Grid Resource Information Service (GRIS) information provider framework, the Grid Index Information Services (GIIS) aggregate directory, the procedures involved in querying MDS for resource information, the Directory Information Tree (DIT), and MDS security.

Support and Usage Help

For usage information about any Globus command, type the command with the `-help` or `-usage` option:

```
% <globus-command-name> -help
% <globus-command-name> -usage
```

Refer to http://www.globus.org/mds for more information on the features and functions of MDS 2.2

Audience

This guide is intended for systems administrators or programmers who are new to the Globus Toolkit and MDS. Some grid background and scripting abilities are assumed, as is a working knowledge of Unix.

Assumptions

This document assumes the following:

- The Globus Toolkit Version 2.2 and MDS Version 2.2 have been designed (indexing structure created), installed, and configured for your environment.

- The $GLOBUS_LOCATION environment variable and relative directories are properly set up.

- Default security measures are set up for your environment.

The starting point of this document, therefore, is that MDS 2.2 is available and ready for your use.

Organization

The remainder of this document is organized as follows:

Chapter 2, Overview of MDS, presents a general discussion of the features, functions, and operation of MDS and related services.

Chapter 3, Using MDS, describes how to query MDS for available resources and data.

Chapter 4, Directory Information Tree (DIT), describes the Lightweight Directory Access Protocol (LDAP) directory organization and the MDS information model.

Chapter 5, MDS Security Configuration, discusses MDS security dependencies, setup procedures, and operations.

Appendix A, Globus Toolkit Security Overview, provides background information on Toolkit security concepts, features, and functions.

Typographic Conventions

The conventions used in this guide are as follows:

%   Unix prompt; do not enter

<italics>   information to be substituted. For example, *<src-dir>* means enter in your source directory pathname. Also used for emphasis.

[]   parameters that sometimes may be omitted

\   indicates that a command line continues on the next line

How to Use This Manual Online

If you are reading the PDF version online, use the button marked with a T or ABC to make text selectable; then you can copy the examples and paste them into your Unix

command line. Be careful not to copy the backslashes (\) into the middle of your command lines. Also watch for disappearing hyphens (-).

Feedback on This Document

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

Related Documents

- *Grid Information Services for Distributed Resource Sharing*
- *Anatomy of the Grid*
- *Globus Toolkit 2.2 Release Notes*
- *Globus Toolkit 2.2 Installation Instructions*
- *MDS 2.2 New Features*
- *MDS 2.2 System Requirements*
- *MDS 2.2 Installation and Configuration*
- *MDS 2.2 Schemas*
- *MDS 2.2 Configuration Files*
- *MDS 2.2 Core GRIS Providers*
- *MDS 2.2 GRIS Specification Document: Creating New Information Providers*
- *MDS 2.2: Creating a Hierarchical GIIS*
- *MDS 2.2 Test Suite*
- *Frequently Asked Questions about MDS 2.2*
- *MDS 2.2 GRIS: Adding New Information Providers (aka sensors)*
- *MDS 2.2 Index of VOs – Public GIIS*
- *To Join the Index of VOs – Public GIIS*
- *Consuming Data From MDS 2.2*
- *OpenLDAP 2.0 Administrator's Guide*

Chapter 2. Overview of MDS

This chapter presents a general discussion of the features, functions, and operation of MDS and related services, in the following sections:

- MDS and the Globus Toolkit
- Uses/Benefits of MDS
- Types of Information Available from MDS
- GRIS and GIIS Basics
- Interfacing with MDS
- Basics of MDS Security

MDS and the Globus Toolkit

Let's begin by putting MDS in the perspective of computational grids, the Globus Project™, and the Globus Toolkit.

Computational grids are environments that enable software applications to integrate instruments, displays, computational resources, and information resources that are managed by diverse organizations and widespread locations. Grid applications often involve large amounts of data and/or computing and are not easily handled by today's Internet and web infrastructures.

Grid technologies enable large-scale sharing of resources within groups of individuals and/or institutions. In these settings, the discovery, characterization, and monitoring of resources, services, and computations are challenging problems due to the considerable diversity, large numbers, dynamic behavior, and geographical distribution on the entities in which a user might be interested. Consequently, information services are a vital part of any grid software or infrastructure, providing fundamental mechanisms for discovery and monitoring, and hence for planning and adapting application behavior.

The Globus Project is developing the fundamental technologies needed to build these computational grids. Globus research focuses not only on the issues associated with building computational grid infrastructures, but also on the problems that arise in designing and developing applications that use grid services.

The Globus Project provides software tools that make it easier to build computational grids and grid-based applications. These tools are collectively called the Globus Toolkit. The Toolkit is used by many organizations to build computational grids that can support their applications.

The composition of the Globus Toolkit can be pictured as the following three pillars. Security is the foundation common to all three pillars.
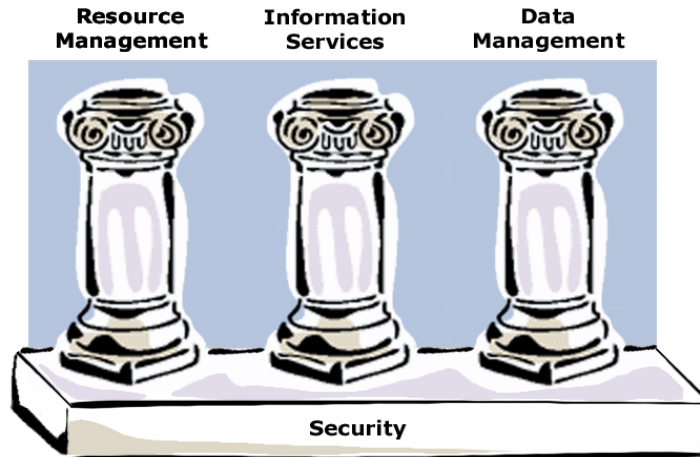
The first pillar of the Globus Toolkit provides Resource Management, which involves the allocation of Grid resources. It includes such packages as the Globus Resource Allocation Manager (GRAM) and Globus Access to Secondary Storage (GASS).

The second pillar of the Globus Toolkit is for Information Services, which provide information about Grid resources and are the focus of this document. Such utilities include the Monitoring and Discovery Service (MDS), which provides the Grid Resource Information Service (GRIS) and the Grid Index Information Service (GIIS).

The third pillar of the Globus Toolkit is for Data Management, which involves the ability to access and manage data in a Grid environment. This involves such utilities as GridFTP and globus-url-copy, which are used to move files between grid enabled devices

In the context of the Globus Toolkit, Information Services have the following requirements:

- Access to static and dynamic information regarding system components
- A basis for configuration and adaptation in heterogeneous, dynamic environments
- Uniform, flexible access to information
- Scalable, efficient access to dynamic data
- Access to multiple information sources
- Decentralized maintenance

As part of this information infrastructure, MDS provides directory services for grids using the Globus Toolkit. MDS uses an extensible framework for managing static and dynamic information about the status of a computational grid and all its components: networks, compute nodes, storage systems, and instruments.

MDS can answer questions about system information such as:

- What resources are available?

- What is the state of the computational grid?

- How can we optimize an application based on configuration of the underlying system?

The MDS uses the Lightweight Directory Access Protocol (LDAP) as a uniform interface to such information. A variety of software components (ranging from commercial servers to open source code) can be used to implement the basic features of MDS: generating, publishing, storing, searching, querying, and displaying the data.

MDS 2.2 provides a configurable information provider component called a Grid Resource Information Service (GRIS) and a configurable aggregate directory component called a Grid Index Information Service (GIIS).

The installation and setup of MDS 2.2 by default registers one GRIS to one GIIS on the same machine. These components are described in more detail later in this chapter.

For more information on grids and grid information services, refer to the white papers *The Anatomy of the Grid* (http://www.globus.org/research/papers.html#anatomy) and *Grid Information Services for Distributed Resource Sharing* (http://www.globus.org/research/papers.html#MDS-HPDC).

For more information on the Globus Project, refer to www.globus.org. For details on the Globus Toolkit, refer to www.globus.org/toolkit.

Uses/Benefits of MDS

You query MDS to discover the properties of the machines, computers and networks that you want to use: How many processors are available at this moment? What bandwidth is provided? Is the storage on tape or disk? Using an LDAP server, MDS provides middleware information in a common interface to put a unifying picture on top of disparate resources.

Information is a critical resource in computational grids. MDS supports the creation of "virtual organizations", where groups of people engaged in collaborative activities share resources with each other. A virtual organization collates and presents information about these resources in a uniform view. An infrastructure that provides coherent system information spanning virtual organizations is necessary in order for applications to configure themselves and to adapt to changing conditions.

MDS provides the necessary tools to build an LDAP-based information infrastructure for computational grids. MDS uses the LDAP protocol as a uniform means of querying system information from a rich variety of system components, and for optionally constructing a uniform namespace for resource information across a system that may involve many organizations. MDS therefore defines an approach to presenting data to users (via the LDAP protocols and some particular schema), and some particular types of servers such as GRIS and GIIS that are part of the Globus Toolkit.

A GRIS provides a uniform means of querying resources on a computational grid for their current configuration, capabilities, and status. A GRIS is a distributed information service that can answer queries about a particular resource by directing the query to an information provider deployed as part of the Globus services on a grid resource. Examples of information provided by this service include host identity (e.g., operating systems and versions), as well as more dynamic information such as CPU and memory availability.

A GIIS provides a means of combining arbitrary GRIS services to provide a coherent system image that can be explored or searched by grid applications. A GIIS thus provides a mechanism for identifying resources of particular interest. For example, a GIIS could list all of the computational resources available within a confederation of laboratories, or all of the distributed data storage systems owned by a particular agency. A GIIS could pool information about all of the grid resources (computation, data, networks, and instruments) in a particular research consortium, thus providing a coherent system image of that consortium's computational grid.

Types of Information Available from MDS

In MDS, interactions between higher-level services (or users) and providers are defined in terms of two basic protocols: a soft-state *registration protocol* for identifying entities participating in the information service, and an *enquiry protocol* for retrieval of information about those entities, whether via query or subscription. A provider uses the registration protocol to notify higher-level services of its existence; a higher-level service uses the enquiry protocol to obtain information about the entities known to a provider, which it merges into its aggregate view.

The separation of concerns between information retrieval, on the one hand, and discovery and monitoring, on the other, means that a wide variety of discovery and monitoring strategies can be supported—implementing different tradeoffs between query language expressiveness, information timeliness, and cost—without modifying the various resources and services that comprise the grid.

MDS 2.2 implements information sources for static host information (operating system version, CPU type, number of processors, etc.), dynamic host information (load average, queue entries, etc.), storage system information (available disk space, total disk space,

etc.), and network information via the Network Weather Service (network bandwidth and latency, both measured and predicted).

A set of information provider programs is included with MDS 2.2. These programs can be used to publish various types of grid information into MDS. A user can also create their own custom information providers.

Core Information Providers

MDS 2.2 features core GRIS information providers for resource information and operational status. These information provider programs provide data to the LDAP server, which can be queried by MDS for the following types of data:

* Platform type and instruction set architecture
* Operating system (host OS) name and version
* CPU information – type, number of CPUs, version, speed, cache, etc.
* Memory – physical and virtual – size, free space, etc.
* Network interface information – machine names and addresses
* File system summary – size, free space, etc.

The GRIS being queried calls the information providers, based on the type of data in its cache. Results returned by an information provider are filtered by the GRIS to eliminate any objects that do not match the client's search space and/or filter constraints, prior to sending them to their destination.

These core information providers are described in detail in *MDS 2.2 Core GRIS Providers*, http://www.globus.org/mds/DefaultGRISProviders.html.

Core providers exist for use on Linux, Solaris, Irix, AIX, and Tru64. There are also generic providers currently available for platforms on which platform-specific core providers do not yet exist, so that basic MDS operability can be tested on those platforms.

The use of generic providers for platforms on which platform-specific core providers do not yet exist is described in *MDS 2.2 Test Suite*, http://www.globus.org/mds/TestSuite.html.

Custom Information Providers

You can also create your own information provider programs to publish specific kinds of data you need into MDS. Refer to the *MDS 2.2 GRIS Specification Document: Creating New Information Providers*, http://www.globus.org/mds/creating_new_providers.pdf for more details.

Querying MDS for information is described in more detail in Chapter 3, *Using MDS*.

GRIS and GIIS Basics

This section describes the GRIS and GIIS components of MDS. This architecture can be summarized and illustrated as follows:

- Resources run a standard information service (GRIS), which speaks LDAP and provides information about the resource (no searching).

- A GIIS provides a "caching" service much like a web search engine. Resources register with a GIIS and the GIIS pulls information from them when requested by a client and the cache has expired.

- A GIIS provides the collective-level indexing/searching function.

Client 1		Resource A
		GRIS

Clients 1 and 2 request info directly from resources.

Client 2		Resource B
		GRIS

GIIS requests information from GRIS services as needed.

Client 3 uses GIIS for searching collective information.

Client 3		GIIS
		Cache contains info from A and B
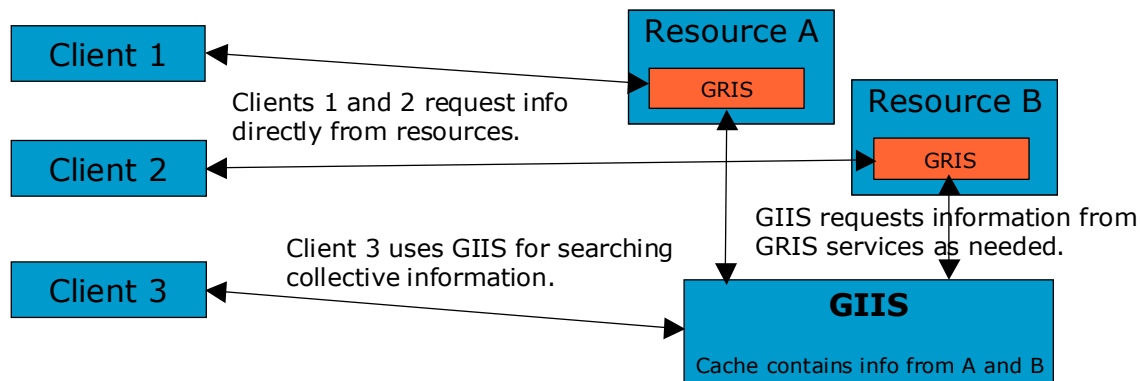
Grid Resource Information Service

MDS 2.2 includes a standard, configurable information provider framework called a Grid Resource Information Service (GRIS). This framework is implemented as an OpenLDAP server back end that can be customized by plugging in specific information sources. Each MDS resource can run a local GRIS. By default, a GRIS service is automatically configured and is assigned to port 2135.

A GRIS can respond to queries from other systems on the grid asking for information about a local machine or other specific resource. A GRIS can be configured to register itself with aggregate directory services (such as a GIIS) so that those services can pass on information about the machine to others.

A GRIS authenticates and parses each incoming information request and then dispatches those requests to one or more "local" information providers, depending on the type of information named in the request. Results are then merged back to the client. To efficiently prune search processing, a specific provider's results are only considered if the provider's namespace intersects the query scope.

A GRIS communicates with an information provider via a well-defined API. A GRIS is configured by specifying the type of information to be produced by a provider and the provider-defined set of routines that implement the GRIS API.

To control the intrusiveness of GRIS operation, improve response time, and maximize deployment flexibility, each provider's results may be cached for a configurable period of time to reduce the number of provider invocations; this cache time-to-live (TTL) is specified per-provider as part of the local configuration.

Results returned by an information provider are filtered by the GRIS to eliminate any objects that do not match the client's search space and/or filter constraints, prior to sending them to the destination. This ensures that the protocol's search semantics are implemented correctly. Filtering is implemented by the GRIS and not the information providers so that (1) simple providers need not duplicate this implementation effort, and (2) cached providers can maximize their performance by returning a superset of results that are then processed out of the cache for each client request.

Grid Index Information Service

MDS 2.2 provides a framework for constructing aggregate directories called a Grid Index Information Service (GIIS), plus an instantiation of this framework that implements a simple aggregate directory that provides a hierarchical structure. The simple directory accepts registration messages from "child" GRIS or GIIS instances and merges these information sources into a unified information space. Client searches may obtain information from any or all of the GIIS' children.

A GIIS can therefore be used to obtain information from multiple GRIS resources via a single command. A GIIS can be set up to act as an organization-wide information server, for a single site or for a multi-site collaboration.

A GIIS can supply a collection of information gathered from multiple GRIS resources, as well as support efficient queries against information spread across multiple GRIS resources.

The GIIS framework comprises three major components: generic registration handling, pluggable index construction, and pluggable search handling. As with a GRIS, GIIS functionality is implemented as a special purpose back end for an OpenLDAP server.

The OpenLDAP front end decodes registration messages and delivers them to the back end to perform any actions necessary to construct and maintain the GIIS' indices.

The MDS 2.2 GRIS and GIIS implementations have much in common. Both rely on an LDAP front end for protocol processing, authentication, and result filtering. Both use a common API for customization, and can in fact co-exist within the same server. Using a

common protocol for provider and directory interactions not only promotes interoperability, it also simplifies implementation.

While a typical configuration is to have a GIIS act as a server to one or more GRIS clients, a GIIS can be either a client or a server or both, depending on its hierarchical relationship (if any) to other host machines. See the *Creating a Hierarchical GIIS* document (http://www.globus.org/mds/hierarchical_GIIS.pdf) for more information.

Interfacing with MDS

You can interface with MDS in either of two ways: interactively or programmatically.

To use MDS interactively, you enter commands at the Unix command line. This type of usage is appropriate for a search of system resources or status.

To use MDS programmatically, you insert program calls (search commands) in your application.

For interactive use, you enter MDS queries (typically with the `grid-info-search` command) to return values for GRIS or GIIS data objects based on the current schema. You specify these objects in the command.

The output from MDS commands is typically a listing of data values, displayed at your workstation or sent to wherever *stdout* is defined on your system.

The process of querying MDS is described in more detail, with example command input and output, in the *Querying MDS* section of Chapter 3, *Using MDS*.

Basics of MDS Security

This section presents an overview of MDS security. Security is described in more detail in Chapter 5, *MDS Security Configuration*.

This section assumes that your security certificates and authorizations have been set up by your Globus administrator. If you need a new certificate or need to modify a certificate, contact your administrator or refer the *Certificates* section below and to the documentation mentioned therein.

The essence of MDS security is twofold: are you who you say you are (authentication) and are you allowed to access the resources you are requesting for the tasks you want to perform (authorization). Authorization grants access based on authenticated identity.

In MDS 2.2, client and server can mutually authenticate using public key technology. Access can be restricted to trees of data or categories of information such as object

classes and attribute types. A particular name or everything below it can be accessed to return information on a set of results such as CPU load.

MDS 2.2 adheres to the security requirements of the Globus Toolkit 2.2 and has some specific security requirements of its own. Appendix A, Globus Toolkit Security, describes basic Toolkit security concepts.

The security mechanism used by the Globus Toolkit is the Grid Security Infrastructure (GSI), which enables the use of certificates and various files to provide authentication and authorization services. GSI is described in more detail at www.globus.org/security.

GSI is a library for providing generic security services for applications that will be run on the grid. GSI provides programs to facilitate login to a variety of sites, while each site has its own flavor of security measures.

GSI provides a single sign-on authentication service, to identify a user for access to multiple Grid resources via one sign-on procedure. GSI also provides local control over access rights and mapping from global to local user identities.

Certificates

A central concept in GSI authentication is the certificate. Every user and service on the Grid is identified via a certificate. A GSI certificate contains four primary pieces of information, as follows:

- A subject name, which identifies the person or object represented by the certificate.
- The public key belonging to the subject.
- The identity of a Certificate Authority (CA) that has signed the certificate to verify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

For authenticated access to MDS, each Globus Toolkit user requires a user certificate and each server requires an LDAP service certificate. These certificates allow the user and the server, respectively, to participate in the authentication process. (Each host requires a host certificate if it is running a gatekeeper or other service besides MDS.) All of these certificates are X.509 compatible.

If they have not already been provided by your Globus administrator, you can obtain the Globus Toolkit user and host (if required) certificates by submitting a request for each one to the CA. Doing this creates a private key and allows you to acquire a certificate, in each case. This process is described in more detail in Chapter 5, *MDS Security Configuration*.

When a job is submitted to the host by the user, before any allocation of resources occurs, a process of mutual authentication ensures that the user has permission to execute jobs on the computer and that the host is the correct resource.

The LDAP service certificate is needed by the LDAP server in order to run in authenticated mode. If it has not already been provided by your Globus administrator, you can request a service certificate by using the `grid-cert-request` command. In fact, this single command is used to request user, host, and service certificates, depending on the options used on the command. The use of `grid-cert-request` is described in more detail in Chapter 5, *MDS Security Configuration*.

Without the LDAP service certificate, only anonymous bind will work in MDS commands and files. You can install the certificate later, restart the service, and then test with authentication.

The LDAP service certificate is located by MDS 2.2 based on the GSI installation and configuration.

Proxies

If a Grid computation requires that several Grid resources be used (each requiring mutual authentication), or if there is a need to have agents (local or remote) requesting services on behalf of a user, the need to re-enter the user's password can be avoided by creating a proxy.

Proxies are certificates signed by the user (rather than the CA), or by another proxy, that do not require a password to submit a job. They are intended for short-term use, when the user is submitting many jobs and cannot be troubled to repeat their password for every job.

Proxies provide a convenient alternative to constantly entering passwords and enable delegation of access rights, but are also less secure than the user's normal security credential. Therefore, they should always be user-readable only, and should be deleted after they are no longer needed (or after they expire).

Refer to Chapter 5, *MDS Security Configuration*, for more specifics on creating proxies.

Chapter 3. Using MDS

You can use MDS to both locate and determine characteristics of resources.

For locating resources, you can ask: Where are resources with required architecture, installed software, available capacity, network bandwidth, etc.?

For determining resource characteristics, you can ask: What are the physical characteristics, connectivity, and capabilities of a resource?

More specifically, you can perform queries with MDS such as the following:

- Determine the characteristics of a compute resource, such as IP address, software available, system administrator, networks to which connected, OS version, load, process information, storage information, memory, and queue availability for large jobs.

- Determine the characteristics of a network, such as bandwidth and latency, protocols, and logical topology.

- Determine the characteristics of the Globus infrastructure, such as hosts and resource managers.

This chapter describes how to do the following:

- Query MDS using the `grid-info-search` command
- Discover available resources and their characteristics
- Query resources directly
- Understand performance and timing dependencies affecting queries
- Use the diagnostics and logging capabilities of MDS

Before you begin using MDS for queries, make sure you have set your environment.

You need to set your environment before using MDS (or any other) commands from the Globus Toolkit. You need to do this on both the client and server machines.

First, make sure you have set GLOBUS_LOCATION to the location of your Toolkit installation. This directory should be owned by a user named "globus."

There are two environment scripts, called GLOBUS_LOCATION/etc/globus-user-env.sh and GLOBUS_LOCATION/etc/globus-user-env.csh. Enter the one that corresponds to the type of shell you are using.

For example, in csh or tcsh, you enter:

```
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

In sh, bash, ksh, or zsh, you enter:

```
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

Additionally, make sure of the following:

1. You have obtained the necessary certificates.
2. You have created a proxy on your client for authenticated access.
3. You have started MDS on your server.

These steps are described in detail under *MDS Security Setup Procedures* in Chapter 5, MDS Security Configuration. Contact your Globus Project administrator if you need additional assistance.

Querying MDS

You can query MDS for information using the `grid-info-search` command. This command defaults to the site service/port (GIIS).

You can also use the core information provider commands, as described in *MDS 2.2 Core GRIS Providers* (http://www.globus.org/mds/DefaultGRISProviders.html). Although `grid-info-search` calls `ldapsearch`, which calls the GRIS back end, which calls core information providers, you can also enter the core information provider commands directly as required.

The grid-info-search Command

To query the GIIS at your site, use `grid-info-search` in the following format:

```
grid-info-search [options]
```

where `options` are as follows:

```
-config file
```
    Specifies a different configuration file to obtain MDS defaults.
    Note that this option overrides the variables set in the user's environment and previously listed command line options.

```
-mdshost host (-h)
```
    Is the host name on which the MDS server is running. The default is $GRID_INFO_HOST.

`-mdsport port (-p)`
Is the port number on which the MDS server is running. The default is port $GRID_INFO_PORT.

`-anonymous (-x)`
Uses anonymous binding instead of GSSAPI.

`-mdsbasedn branch-point (-b)`
Is the base location in DIT from which to start the search. The default is "${GRID_INFO_BASEDN}".

`-s` *scope*
Specifies the scope of the search. *scope* can be `base` (base dn level), `one` (base dn level plus one level down) or `sub` (all from base dn level and down).

`giisregistrationstatus`
Specifies that the status of servers registered to a GIIS or from which a GIIS is getting data be returned from the search. This option requires the `-b` and `-s base` options on the command for querying from the default server (as defined in the grid-info.conf file). The `-h` and `-p` options are required for querying from other servers. See *Diagnostics and Logging* later in this chapter.

`"attribute"`
Specifies a single attribute to be returned from the search, such as MdsCpu.

`"filter"`
Sets an attribute relationship to a value for the search:
> `"(attribute=value)"`
> `"(attribute>=value)"`
> `"(attribute<=value)"`
> `"(attribute~=value)"`

> `"(attribute=*)"`   Searches for the presence of an item.
> `"(attribute=*a)"`   Searches for a substring item.

`-mdstimeout seconds (-T)`
Is the amount of time (in seconds) one should allow to wait on an MDS request. The default is $GRID_INFO_TIMEOUT.

Examples of using this command are presented below.

grid-info-search Examples

The following sections present typical examples of using the `grid-info-search` command:

- Example 1 shows how to query all objects on a GRIS.

- Example 2 shows how to query for the amount of free file system space on all machines on a GIIS.

- Example 3 shows how to query for CPU model and speed on a single machine on a GIIS.

Example 1 – Query all objects on a GRIS

This example shows how to display all of the data objects and resources on a single, local machine set up as a GRIS. The command is as follows:

```
grid-info-search -h giis-demo.globus.org -p 8463 -b 'Mds-vo-name=local,
o=Grid'
```

The search runs on the host and port indicated; `Mds-Vo-name=local` means that the search will start on a GRIS.

For illustration, segments of the output are as follows:

```
SASL/GSI-GSSAPI authentication started
SASL SSF: 56
SASL installing layers
version: 2

#
# filter: (objectclass=*)
# requesting: ALL
#

# dc-user2.isi.edu, local, grid
dn: Mds-Host-hn=dc-user2.isi.edu,Mds-Vo-name=local,o=Grid
objectClass: MdsComputer
objectClass: MdsComputerTotal
objectClass: MdsCpu
objectClass: MdsCpuCache
objectClass: MdsCpuFree
objectClass: MdsCpuSmp
objectClass: MdsCpuTotal
objectClass: MdsCpuTotalFree
objectClass: MdsFsTotal
objectClass: MdsHost
objectClass: MdsMemoryRamTotal
objectClass: MdsMemoryVmTotal
objectClass: MdsNet
objectClass: MdsNetTotal
objectClass: MdsOs
Mds-Computer-isa: IA32
Mds-Computer-platform: i686
Mds-Computer-Total-nodeCount: 1
Mds-Cpu-Cache-l2kB: 512
```

```
[…]

# processors, dc-user2.isi.edu, local, grid
dn: Mds-Device-Group-name=processors, Mds-Host-hn=dc-user2.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: 20020404024020Z
Mds-validto: 20020404024120Z
Mds-keepto: 20020404024120Z
Mds-Cpu-Cache-l2kB: 512
[…]

# local, Grid
dn: Mds-Vo-name=local,o=Grid
objectClass: GlobusStub

# search result
search: 5
result: 0 Success

# numResponses: 20
# numEntries: 19
```

Example 2 – Query file system space on a GIIS

This example shows how to query for the amount of free file system space on all
machines on a GIIS. The command is as follows:

```
grid-info-search -x -h giis-demo.globus.org -p 8422 -b 'Mds-Vo-
name=site,o=Grid' Mds-Fs-freeMB
```

This search uses anonymous binding (-x) and runs on the host and port indicated; Mds-
vo-name=site means that the search will start on a GIIS. Your GIIS may be named
something other than the default of site.

For illustration, segments of the output are as follows:

```
#
# filter: (objectclass=*)
# requesting: Mds-Fs-freeMB
#

# dc-user2.isi.edu, site, Grid
dn: Mds-Host-hn=dc-user2.isi.edu,Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 174
Mds-Fs-freeMB: 26
Mds-Fs-freeMB: 42888
```

```
Mds-Fs-freeMB: 503
[…]

# filesystems, dc-user2.isi.edu, site, Grid
dn: Mds-Device-Group-name=filesystems, Mds-Host-hn=dc-
user2.isi.edu,Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 174
Mds-Fs-freeMB: 26
Mds-Fs-freeMB: 42888
Mds-Fs-freeMB: 503
[…]

# jupiter.isi.edu, site, Grid
dn: Mds-Host-hn=jupiter.isi.edu,Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 1075
Mds-Fs-freeMB: 141
Mds-Fs-freeMB: 1711
Mds-Fs-freeMB: 2430
Mds-Fs-freeMB: 3657
Mds-Fs-freeMB: 4756
[…]

# filesystems, jupiter.isi.edu, site, Grid
dn: Mds-Device-Group-name=filesystems, Mds-Host-hn=jupiter.isi.edu,
Mds-Vo-name=site,o=Grid
Mds-Fs-freeMB: 1075
Mds-Fs-freeMB: 141
Mds-Fs-freeMB: 1711
Mds-Fs-freeMB: 2430
Mds-Fs-freeMB: 3657
Mds-Fs-freeMB: 4756
[…]

# search result
search: 2
result: 0 Success

# numResponses: 131
# numEntries: 130
```

Example 3 – Query CPU data on a single machine on a GIIS

This example shows how to query for CPU model and speed on a single machine on a
GIIS. The command is as follows:

```
grid-info-search -x -h giis-demo.globus.org -p 8422 -b 'Mds-Vo-
name=site, o=Grid' '(&(objectclass=MdsCpu)(Mds-Host-
hn=lucky6.mcs.anl.gov))' Mds-Cpu-model Mds-Cpu-speedMHz
```

This search uses anonymous binding (-x) and runs on the host and port indicated; Mds-
Vo-name=site means that the search will start on a GIIS. Your GIIS may be named
something other than the default of site. The objectclass expression specifies CPU
data on a specific machine on the GIIS. The last two arguments specify the CPU model
and speed.

The output appears as follows:

```
version: 2

#
# filter: (&(objectclass=MdsCpu)(Mds-Host-hn=lucky6.mcs.anl.gov))
# requesting: Mds-Cpu-model Mds-Cpu-speedMHz
#
# lucky6.mcs.anl.gov, lucky6, site, Grid
dn: Mds-Host-hn=lucky6.mcs.anl.gov,Mds-Vo-name=lucky6,Mds-Vo-
name=site,o=Grid
Mds-Cpu-model: Intel(R) Pentium(R) III CPU family      1133MHz
Mds-Cpu-speedMHz: 1125

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

LDAP Browsers

Note that there are LDAP browsers available that provide a convenient GUI-based way to peruse and in some cases query the resources on a computational grid. This can be particularly helpful in a hierarchical GIIS with many resources registering with each other.

For users of Java JRE 1.3.1 and higher (e.g., Linux, Windows, Solaris, etc.) there is a cross-platform compatible browser written by Jarek Gawor of the Globus Project. This browser is available from: http://www-unix.mcs.anl.gov/~gawor/ldap/demo.html

For Windows users, the Softerra LDAP Browser provides an interface similar to Windows Explorer, as shown in the following example:
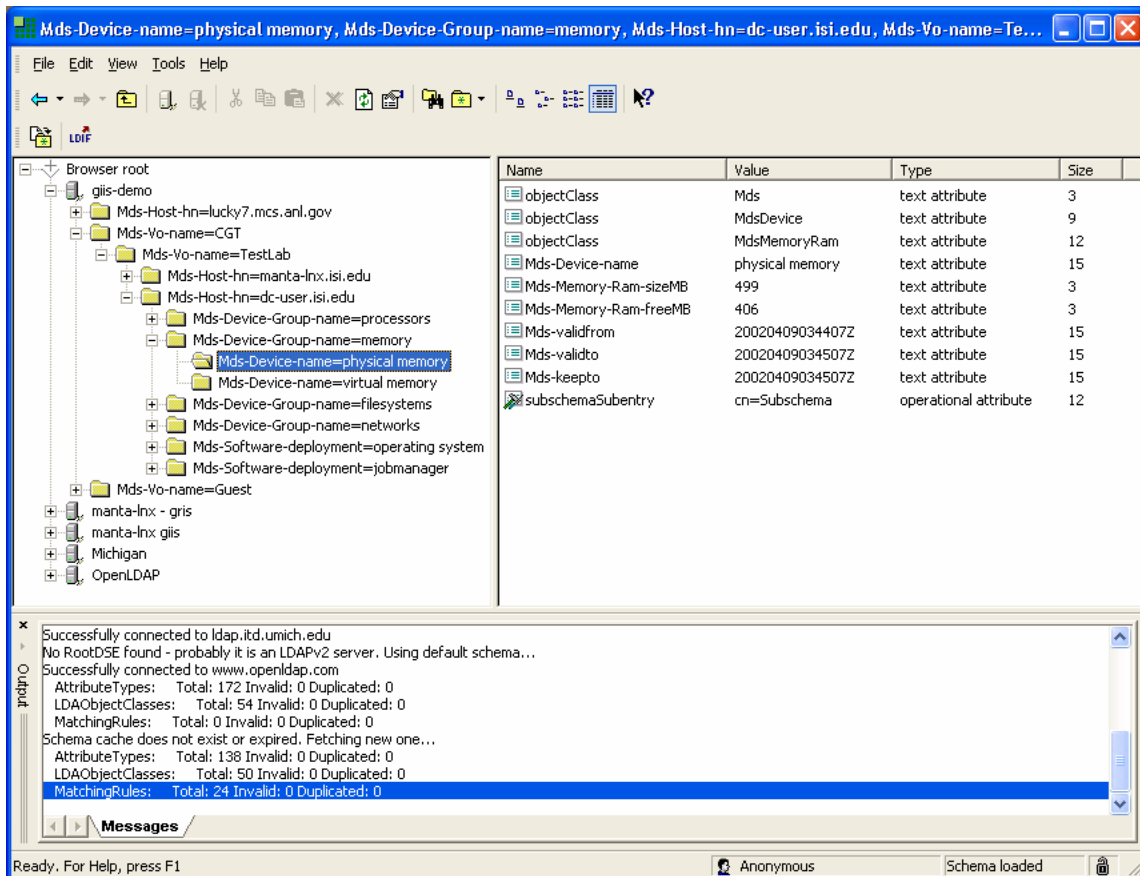
The Softerra LDAP browser can be downloaded from:
http://download.cnet.com/downloads/0-3364666-100-8589915.html?tag=st.dl.1000

More information about the Softerra browser can be obtained from www.softerra.com.

Additional ways of accessing MDS data are described in *Consuming Data From MDS 2.2*, http://www.globus.org/mds/getmdsdata/cmdsdata.html.

Performance Expectations From a GRIS or GIIS Query

The results returned from a grid-info-search command query can be affected by the complexity of a hierarchy involving multiple GRIS and GIIS resources and by the timing values specified for the individual resources.

This section presents an overview of these considerations, and the next section focuses more specifically on setting timing values.

The performance of a query to a GRIS is dependent upon the performance of the information providers that the GRIS accesses, as well as the TTL data for the information they deliver:

- When a query to a GRIS arrives, if the data requested is live and cached, the query will be answered very quickly.

- If the data requested has been flushed from the cache because it has expired, then the GRIS will invoke the information provider or providers that supply the information required by the query. If these providers start up and deliver information quickly, then the GRIS can in turn answer the client query relatively quickly, though it will be a little slower than if the data had been in cache.

- If an information provider takes a long time to deliver its information, either because there is a lot of overhead for it to start up or acquire the data, or because there is a lot of data to deliver, then this will negatively impact the performance the client query sees.

- Further, the GRIS will allow a certain, configurable, time window for the information provider to finish its delivery. As a result, if the provider does not finish within the time window, the GRIS will terminate its link to the provider and move on to the next provider, if any. As a result, data from that provider may never appear in the GRIS. To resolve this, either extend the time window that the GRIS waits, or send less data to the GRIS. In this case, though, the client query to the GRIS will take a long time to return, since it will wait for the timeout window and the amount of time to access any other information providers before it can answer the client query.

- To extend this time window for the information provider, try increasing the value of the `timelimit` parameter for the relevant provider in the etc/grid-info-resource-ldif.conf file. (Refer to *MDS 2.2 Configuration Files* for more details on and an example of this file.)

The performance of a query to a GIIS is dependent upon the performance of the GRIS's that it accesses, as well as the TTL data for the information they deliver:

- When a query to a GIIS arrives, if the data requested is live and cached, the query will be answered very quickly.

- If the data requested has been flushed from cache because it has expired, then the GIIS might query a GRIS that supplies that information. The performance of this sub-query is like any client querying a GRIS, as described above.

- Alternatively, in a GIIS hierarchy, a GIIS query for un-cached data might in turn query a subsidiary GIIS. The performance of this sub-query to another GIIS is based upon whether it has cached data, and if not, the performance of its sub-hierarchy.

In summary, there is no a priori formula for predicting the performance of a query to MDS. The more complex the GIIS hierarchy, the more unpredictable the performance, and in general, the longer that a query might take to answer, depending upon TTL data.

It is possible to mitigate this performance variability by attempting to insure that data queried is normally in cache. One can do this either by increasing the TTL value for the data (but that might not make sense), or by causing the cache to fill by periodically running a script that triggers this. This can be done at any or all levels of a GIIS hierarchy, and at the GRIS level as well. If the caches are filled regularly by scripts running periodically, then client queries will most likely find the data they are looking for in cache.

Timing Considerations

As described in *MDS 2.2 Configuration Files*, the grid-info-resource-register.conf file can list one or more GIIS servers to which a GRIS will register directly. The parameters in this file identify host names, ports and several time values that control the registration messages from a GRIS to a GIIS server.

Several grid-info-resource-register.conf timing parameters are of value in ensuring correct query registration and complete return of data. These parameters are described in the paragraphs below.

Registration period (regperiod) and ttl

Since a GIIS can be either a client or a server, this section and the *Cache ttl* section following define "registrar" as the machine receiving registration data, and "registrant" as the machine sending registration data.

The `regperiod` and `ttl` parameters are both in the grid-info-resource-register.conf file. The registration period is the notification time for service availability. That is, `regperiod` specifies how often the registrant sends out a message saying that it exists. The `ttl` specifies how long the receiving registrar should keep the registration information. A general recommendation for the `ttl` value is twice the registration period. This will reduce the potential problem of building a GIIS but not getting all the expected results in cache.

For example, let's say we have a GRIS that sends out a message saying it exists, and it does this every minute. Then we have a GIIS that keeps this message for two minutes. If there is a network glitch and the GIIS misses one existence message, it still knows that the GRIS exists. This is because `ttl=2xregperiod` in this case. If `ttl=regperiod` or `ttl<regperiod`, the existence message from the GRIS could be missed, and it would appear to the GIIS that the GRIS had disappeared.

Cache ttl (cachettl)

The `cachettl` parameter is in the grid-info-resource-register.conf file. The `cachettl` specifies how long the registrar should keep the registrant's data in the registrar's cache. In other words, the `cachettl` is the registrant's "recommended" amount of time for the registrar to keep its data in cache. The default value of `cachettl` is 30 seconds.

When queries are received by the registrar within the time specified by `cachettl`, the data in the registrar's cache is returned. When queries are received by the registrar after the `cachettl` value has timed out, the registrar requests the data from the next lower level of the hierarchy.

For example, say you have a registrar GIIS with a registrant GRIS and a single information provider, and the GIIS receives a query after `cachettl` times out. The registrar GIIS will then request data from the registrant GRIS. If cached data is available (i.e., has not timed out) from the registrant GRIS, that data will be sent to the registrar GIIS. If cached data is not available from the registrant GRIS, the GRIS invokes the information provider, receives data from the provider, and sends the data to the registrar GIIS.

This process is illustrated in the following diagram:
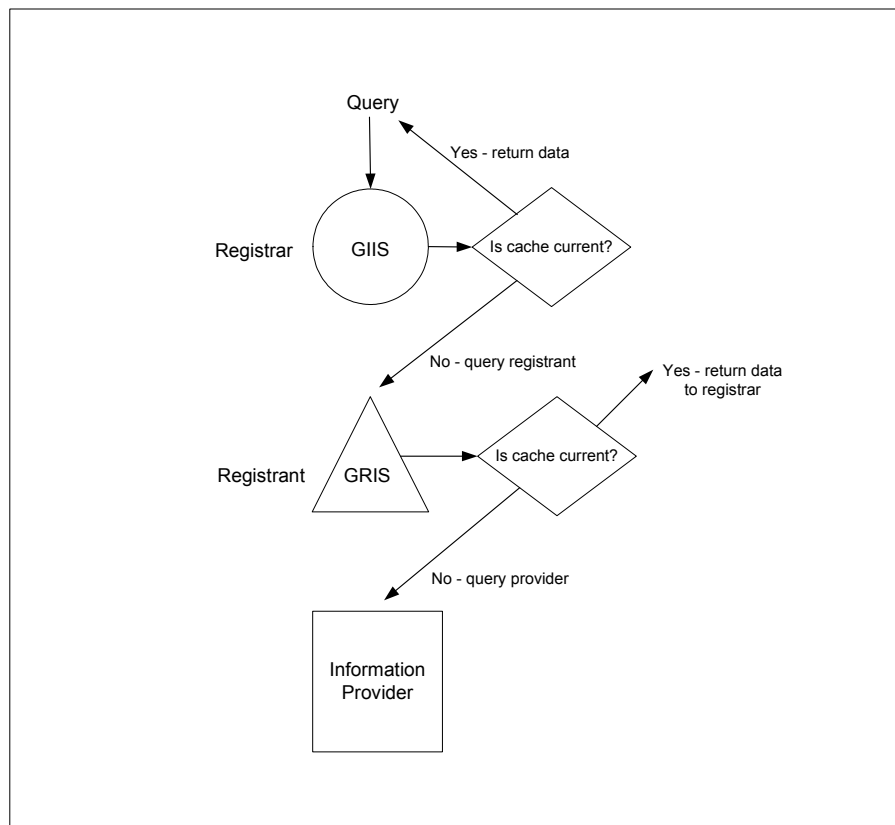
Note that for simplicity, the above example and diagram use a single registrar and a single registrant with one information provider. A hierarchy can consist of multiple registrars and registrants with multiple information providers. With multiple providers, a registrant GRIS checks the `cachetime` (in grid-info-resource-ldif.conf) of each subtree associated with each information provider.

In caching, the clock starts when a registrar receives data, regardless of when the data was generated. The data might be obsolete when it comes to the top-level GIIS, but the GIIS still returns that data.

Note that MDS uses its own caching mechanism, and does not rely on that provided by standard OpenLDAP.

Note also that the proper operation of these timing and registration parameters depends on the synchronization of the system clocks on the machines involved. If the clocks are not synchronized, registration messages may be lost, and partial data or no data may be returned.

Refer to *MDS 2.2: Creating a Hierarchical GIIS* for additional information on timing and registration control.

Diagnostics and Logging

MDS provides capabilities for checking servers registered to a GIIS, writing OpenLDAP and GRIS/GIIS back end messages to a log file, and viewing the MDS system configuration. These capabilities are described in the following sections.

GIIS Registration Status Checking

The status of servers that are registered to a GIIS or from which a GIIS is getting data can be checked with an option on the `grid-info-search` command. If several servers are registered to a GIIS and it appears that not all of them are sending data, `grid-info-search` with the `giisregistrationstatus` option can be used to check the status of those servers.

Note that the term "server" is defined here as a resource specified in MDS with a host name, port number, and suffix (usually "Mds-vo-name=site, o=Grid" or "Mds-vo-name=local, o=Grid" by default).

The command output displays registration objects that include the status type: valid, invalid, or purged. These status types are derived from the `validfrom`, `validto`, and `keepto` parameters, which are generated from the `ttl` parameter in the grid-info-resource-register.conf file. (See *MDS 2.2 Configuration Files* for more details.)

The `validfrom`, `validto`, and `keepto` parameters represent the timeframe during which one server keeps registration messages sent from another server. This timeframe can be illustrated as follows:

```
     validfrom:      validto:       keepto:
        ↓               ↓              ↓
time=|     VALID   |   INVALID  |  PURGED  |
```

An example of these parameters from a grid-info-site-giis.conf file is as follows:

```
Mds-validfrom: 20020522174628Z
Mds-validto: 20020522180128Z
Mds-keepto: 20020522180128Z
```

These parameters show year, month, day, and time in hours, minutes, and seconds.

The `grid-info-search` command with the `giisregistrationstatus` option displays the status of the machines one level below (i.e., the immediate children) of the GIIS being queried.

For example, the following command:

```
grid-info-search -x -b "mds-vo-name=site,o=grid" -s base
giisregistrationstatus
```

displays the status of the default server (as defined in the grid-info.conf file) and a server registered to it. Note that the `-b` and `-s base` options are required on the command for querying from the default server. The `-h` and `-p` options are required for querying from other servers. See *The grid-info-search Command* section earlier in this chapter for the full command syntax.

The command output is as follows. Registration objects are displayed for a GRIS on host1 registering to the GIIS on host1, and for a GRIS on host2 registering to the GIIS on host1.

```
version: 2

#
# filter: (objectclass=*)
# requesting: giisregistrationstatus
#

# site, Grid
dn: Mds-Vo-name=site,o=Grid
objectClass: Mds
objectClass: MdsVoOp
objectClass: MdsService
objectClass: MdsServiceLdap
Mds-Service-type: ldap
Mds-Service-hn: host2
Mds-Service-port: 2135
Mds-Service-Ldap-suffix: Mds-Vo-name=local, o=grid
```

```
Mds-Service-Ldap-sizelimit: 0
Mds-Service-Ldap-timeout: 30
Mds-Service-Ldap-cachettl: 1200
Mds-Bind-Method-servers: ANONYM-ONLY
Mds-Reg-status: VALID

# site, Grid
dn: Mds-Vo-name=site,o=Grid
objectClass: Mds
objectClass: MdsVoOp
objectClass: MdsService
objectClass: MdsServiceLdap
Mds-Service-type: ldap
Mds-Service-hn: host1
Mds-Service-port: 2135
Mds-Service-Ldap-suffix: Mds-Vo-name=local, o=grid
Mds-Service-Ldap-sizelimit: 0
Mds-Service-Ldap-timeout: 20
Mds-Service-Ldap-cachettl: 1200
Mds-Bind-Method-servers: AUTHC-ONLY
Mds-Reg-status: VALID

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

Log File Facility

MDS 2.2 uses the Unix syslog daemon, configured in /etc/syslog.conf, to log OpenLDAP messages and GRIS/GIIS back end messages. The `syslog restart` command causes the syslog daemon to read syslog.conf and direct the messages into a designated log file.

The syslog daemon and configuration file are described in detail in their related `man` pages: `man 3 syslog` and `man 5 syslog.conf`.

You put an entry like:

local4.*   local-log-file

in the /etc/syslog.conf file, and then run `syslog restart`. You run this command as root; the exact path name of the command will vary depending on your particular machine and environment.

The `syslog restart` command publishes messages into the log file name indicated by the local4.* entry in the syslog.conf file.

The logged messages show slapd server messages for items such as machine registration status, success or failure of a search, and slapd operational status. The following is an example of messages in a local-log-file:

```
Jun 12 13:07:58 host1 slapd[16894]: REGISTERING host1:2135 (Mds-Vo-
name=local, o=grid)
Jun 12 13:07:58 host1 slapd[16894]: slapd starting
Jun 12 13:08:03 host1 slapd[16896]: daemon: conn=0 fd=9 connection from
IP=128.9.72.31:35431 (IP=0.0.0.0:22280) accepted.
Jun 12 13:08:03 host1 slapd[16932]: conn=0 op=0 BIND dn="" method=163
Jun 12 13:08:03 host1 slapd[16896]: deferring operation
Jun 12 13:08:03 host1 slapd[16933]: conn=0 op=1 BIND dn="" method=163
Jun 12 13:08:03 host1 slapd[16896]: deferring operation
Jun 12 13:08:03 host1 slapd[16932]: conn=0 op=2 BIND dn="" method=163
Jun 12 13:08:03 host1 slapd[16896]: deferring operation
Jun 12 13:08:03 host1 slapd[16934]: conn=0 op=3 BIND dn="" method=163
Jun 12 13:08:03 host1 slapd[16896]: deferring operation
Jun 12 13:08:03 host1 slapd[16932]: conn=0 op=4 SRCH base="mds-vo-
name=site,o=grid" scope=0 filter="(objectClass=*)"
Jun 12 13:08:03 host1 slapd[16932]: conn=0 op=4 RESULT tag=101 err=0
text=
Jun 12 13:08:03 host1 slapd[16935]: conn=0 op=5 UNBIND
Jun 12 13:08:03 host1 slapd[16935]: conn=-1 fd=9 closed
Jun 12 13:08:11 host1 slapd[16896]: daemon: conn=1 fd=9 connection from
IP=128.9.72.31:35432 (IP=0.0.0.0:22280) accepted.
Jun 12 13:08:11 host1 slapd[16934]: conn=1 op=0 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16896]: deferring operation
Jun 12 13:08:11 host1 slapd[16933]: conn=1 op=1 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16896]: deferring operation
Jun 12 13:08:11 host1 slapd[16932]: conn=1 op=2 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16896]: deferring operation
Jun 12 13:08:11 host1 slapd[16932]: conn=1 op=3 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16896]: deferring operation
Jun 12 13:08:11 host1 slapd[16932]: conn=1 op=4 SRCH base="mds-vo-
name=site,o=grid" scope=2 filter="(objectClass=*)"
Jun 12 13:08:11 host1 slapd[16896]: daemon: conn=2 fd=13 connection
from IP=128.9.72.31:35433 (IP=0.0.0.0:22280) accepted.
Jun 12 13:08:11 host1 slapd[16934]: conn=2 op=0 SRCH base="" scope=0
filter="(objectClass=*)"
Jun 12 13:08:11 host1 slapd[16935]: conn=2 op=1 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16934]: conn=2 op=0 RESULT tag=101 err=0
text=
Jun 12 13:08:11 host1 slapd[16933]: conn=2 op=2 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16896]: deferring operation
Jun 12 13:08:11 host1 slapd[16935]: conn=2 op=3 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16934]: conn=2 op=4 BIND dn="" method=163
Jun 12 13:08:11 host1 slapd[16935]: conn=2 op=5 SRCH base="Mds-Vo-
name=local, o=grid" scope=2 filter="(objectClass=*)"
Jun 12 13:08:12 host1 slapd[16935]: conn=2 op=5 RESULT tag=101 err=0
text=
Jun 12 13:08:12 host1 slapd[16934]: conn=2 op=6 UNBIND
Jun 12 13:08:12 host1 slapd[16934]: conn=-1 fd=13 closed
Jun 12 13:08:12 host1 slapd[16933]: conn=1 op=5 UNBIND
Jun 12 13:08:12 host1 slapd[16932]: conn=1 op=4 RESULT tag=101 err=0
text=
Jun 12 13:08:12 host1 slapd[16932]: SEARCH SUCCEEDED
Jun 12 13:08:12 host1 slapd[16932]: conn=-1 fd=9 closed
Jun 12 14:03:22 host1 slapd[18151]: lt_dlopen failed: (libback_giis.la)
/host1/globus-working-
```

```
3/install/libexec/openldap/gcc32dbg/libback_giis.so: undefined symbol:
lutil_sasl_interact
Jun 12 14:03:22 host1 slapd[18151]: etc/grid-info-slapd.conf: line 12:
failed to load or initialize module libback_giis.la
```

The log file shows you if the search succeeded, if slapd fails to run (and why), if
registration is done, and so forth – all with a time stamp.

MDS System Configuration Information Provider

The MDS system configuration provider is part of the grid-info-mds-core provider. The
system configuration provider generates deployment information such as the service path,
process ID of the current slapd instance, e-mail address of the administrator, any
administrative comments, and LDAP suffixes used by the service. This type of
information is helpful for MDS installation debugging.

The MDS system information provider output can be retrieved by using the `grid-info-`
`search` command to query all objects on a host.

For example, the output from the following command:

```
grid-info-search -x -h giis.globus.org -p 2135
```

contains the following deployment information:

```
# MDS, dc-user.isi.edu, local, grid
dn: Mds-Software-deployment=MDS, Mds-Host-hn=dc-user.isi.edu,Mds-Vo-
name=local,o=grid
objectClass: MdsSoftware
objectClass: MdsService
objectClass: MdsServiceLdap
Mds-Software-deployment: MDS
Mds-Service-type: ldap
Mds-Service-hn: dc-user.isi.edu
Mds-Service-port: 2135
Mds-Service-Ldap-timeout: 30
Mds-Service-admin-contact: jadmin@isi.edu
Mds-Service-Executable-PID: 5757
Mds-Service-Path: /scratch/demo-giis/GL7
Mds-Service-admin-comment: This is giis.globus.org, the top level GIIS
for the Globus Monitoring and Discovery Service.
Mds-Service-Ldap-suffix: Mds-Vo-name=local,o=Grid
Mds-Service-Ldap-suffix: Mds-Vo-name=vo-index,o=Grid
Mds-Service-Ldap-suffix: Mds-Vo-name=giis-demo,o=Grid
Mds-Service-Ldap-suffix: Mds-Vo-name=CGT,o=Grid
Mds-Service-Ldap-suffix: Mds-Vo-name=TestLab,o=Grid
Mds-Service-Ldap-suffix: Mds-Vo-name=Guest,o=Grid
Mds-validfrom: 20020718202422Z
Mds-validto: 20020719082422Z
Mds-keepto: 20020719082422Z
```

The system information provider uses the administrator's e-mail address from the grid-info.conf file and the administrative comments from the grid-info-deployment-comments.conf file.

See *MDS 2.2 Configuration Files* for more details on and examples of grid-info.conf and grid-info-deployment-comments.conf.

Chapter 4. Directory Information Tree (DIT)

The MDS directory structure follows the LDAP model, which consists of a Directory Information Tree hierarchy and object class definitions.

LDAP Definition

LDAP is a client-server protocol for storing information and responding to queries. It is not a database, but is a protocol for database and directory access.

There are two sides to LDAP: client-to-server communications and server-to-server communications. Basic client-to-server communications allow user applications to contact an LDAP server to create, retrieve (the primary function of MDS), modify, and delete data with the standard LDAP commands. Server-to-server communications define how multiple servers on a network share the contents of an LDAP Directory Information Tree (DIT) and how they update and replicate information between themselves.

LDAP was designed to contain small records of information in a hierarchical structure. This structure appears much like the directory tree of a file system, with individual nodes containing attributes and connecting to other subtrees. However, unlike the multi-megabyte files in most user directories, the nodes in an LDAP tree are usually much smaller.

Directory Information Tree (DIT)

The basic structure of LDAP is a simple tree of information. Starting at a root node, it contains a hierarchical view of all its data and provides a tree-based search system for the data (this is one of the most common ways to search data). Resource discovery is performed by a hierarchical search of this tree.

The tree itself is called the Directory Information Tree (DIT). Subtrees of the DIT contain all the information on that LDAP server; the subtrees can be distributed or replicated. The contents of the directory are called object classes and entries. Object classes describe what information can be stored in the directory. Entries group related information. Objects are named by their position in the tree.

Every node in the tree is known as an entry, or Directory Service Entry (DSE). These entries contain the actual records that describe different real and abstract objects in the computing environment, such as users, computers, networks, preferences, etc. The content of a record is stored in an entry as attribute/value (<name, value>) pairs.

The root node of the tree doesn't really exist and can't be accessed directly. There is a special entry called the Root Directory Specific Entry, or rootDSE, that contains a description of the whole tree, its layout, and its contents, but this really isn't the root of the tree itself. Each entry contains a set of properties, or attributes, in which data values are stored. Simply said, each entry is a data structure of variables.

For computational grids, the root we use is "o=grid," where "o" is a descriptor for "organization." The DIT branches out below this root by adding organizations (o), organizational units (ou), resources, etc.

Every node in the DIT structure has a unique path to the root. That path can serve as an unambiguous name to the entry associated with the node.

To refer to each entry in the DIT uniquely, you must use a Distinguished Name (DN). Normally, this is the very first attribute of the entry. The DN presents the full name of the entry within the entire tree. The DN can provide a list of unique attribute names and values along the path from the root of the DIT to an object. The DN provides the path from leaf to root entry in the DIT.

A container is a regular entry that contains a branch point in the DIT. A container can store entries or other containers. Any entry can be used as a container; organization unit (ou) objects are commonly used as containers. A container serves as a component of a DN that specifies part of a path in a search base. You can create a suffix or branch point by placing a container under the suffix or under another container (via an LDIF file, described below).

The *LDAP Data Interchange Format* (LDIF) is commonly used to define records within the DIT. This is typically a text file describing the various elements within a record and how it is organized.

Objects/Schema

As stated above, the LDAP information model is based on entries arranged in a hierarchical tree-like structure. Another way of describing an entry is as a collection of attributes that has a globally unique DN. Each of the entry's attributes has a *type* and one or more *values*. A type is created by associating an object class.

LDAP allows you to control which attributes are required and allowed in an entry, through the use of a special attribute called object class. An object class consists of a name, all of the elements (<name, value> pairs) that must be present in an instance of that object class, and all of the elements that may be present in an instance of that object class.

LDAP has a set of standard object class definitions, which define attributes for an entry and define what values those attributes may contain. Attributes may be required or optional.

The values of the object class attribute determine the schema rules the entry must obey. We define schema here as the collection of attribute type definitions, object class definitions and other information that a server uses to determine how to match a filter or attribute value assertion (in a compare operation) against the attributes of an entry, and whether to permit add and modify operations.

LDAP directory schema can be illustrated as shown in the following figure:
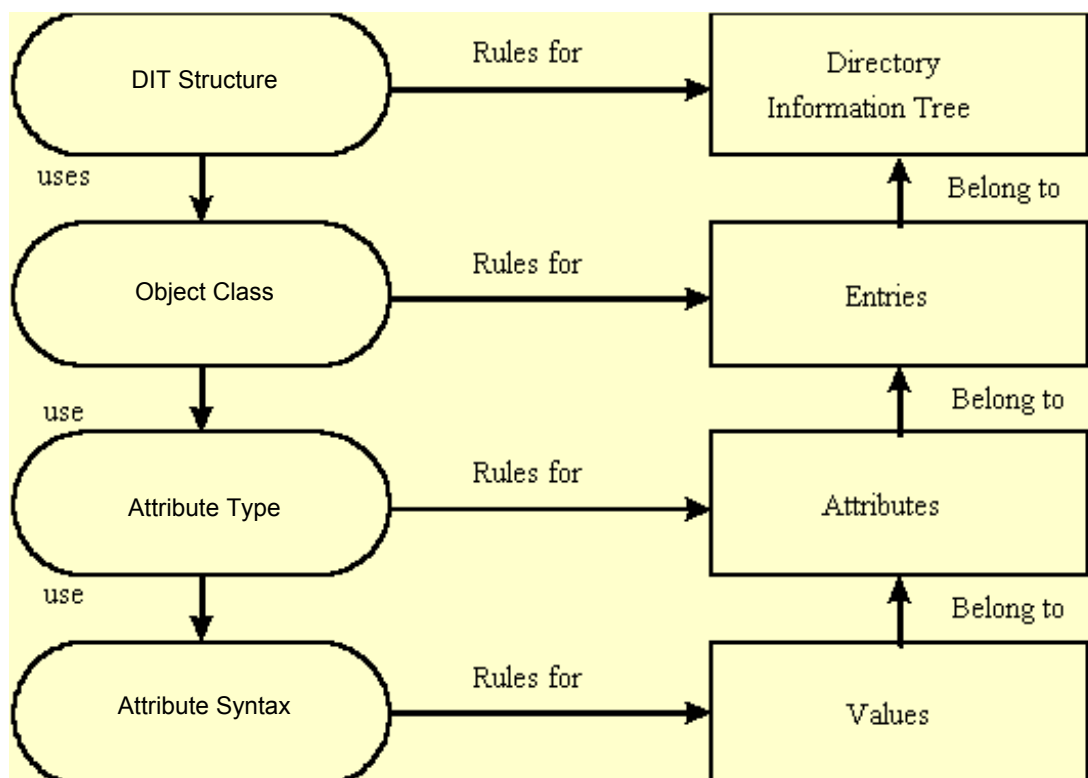
The MDS schema is described later in this chapter.

Query/Operations

LDAP can be considered as a global directory service used to search for information. No matter which LDAP server a client connects to, it sees the same view of the directory. One or more LDAP servers contain the data making up the LDAP directory tree.

The LDAP session is fairly basic in nature. It begins when a client issues a bind operation to the server. The server may require an authentication step at this point. Once the connection between the two parties has been established, the client can send any number of LDAP interrogation commands. Note that MDS does not allow clients to update data via the standard LDAP interface.

LDAP permits the creation of nodes that represent the root of a project hierarchy – these nodes provide search scoping by establishing project "roots" that sit at the top of a hierarchy of project resources. You can create a root entry by importing an LDAP Directory Interchange Format (LDIF) file.

To find a starting point of a DIT, you need to query the root DSE to see what suffixes are present unless you already know the suffix. Two descriptors are commonly used as naming conventions to denote directory suffixes. These descriptors are domain component (dc) and organization (o) and either one can be used.

The LDAP server that provides the top of the Grid DIT (o=grid) can also be used as the top of a set of catalogs that index various objects of interest in the Grid community, and provide naming consistency and discovery for those objects. The DIT is formed using organizations rooted at o=grid.

For example:
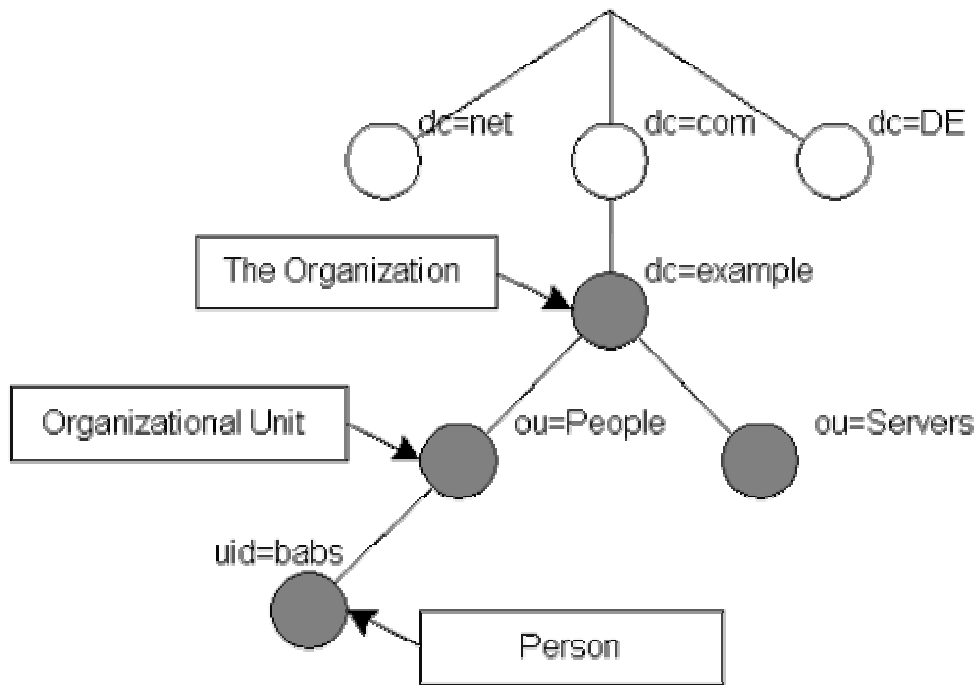
In MDS, as in other LDAP-based applications, we are concerned with searching the DIT for information. A search operation sends a set of parameters to the server that request objects that match the operation. A fully declared search involves eight parameters:

1. The tree or subtree you want to search
2. How many levels of the tree to search
3. How aliases to other parts of the tree should be followed
4. The number of entries to be returned

5. The time limit for the search
6. Whether only the attribute types and not its values are needed
7. A filter operation to be applied to the search list
8. The list of attributes to be returned

MDS 2.2 Information Model and the DIT

The MDS 2.2 information model contains three kinds of information, as follows:

- Structural information

Structural information is represented by a resource hierarchy that maps to objects, which are represented by named positions in the LDAP DIT.

- Merged information

Merged information is represented by the "joining" of parent with child data, and is used to simplify common query patterns.

- Auxiliary information

Auxiliary information uses LDAP auxiliary object classes in a uniform representation of leaf/parent data.

The use of auxiliary information is a way of representing a collection of related attributes, and thus can be combined in ways that structures cannot. An object must have exactly one structural type, but may have zero or more auxiliary types also.

In MDS, this information model is used in the GRIS host object hierarchy, as follows:

```
Mds-Host-name=hostname
   Mds-Software-Deployment=operating system
   Mds-Device-Group-name=processors
      Mds-Device-name=cpu 0
   Mds-Device-Group-name=memory
      Mds-Device-name=physical memory
      Mds-Device-name=virtual memory
   Mds-Device-Group-name=filesystems
      Mds-Device-name=/scratch1
      Mds-Device-name=/scratch2
   Mds-Device-Group-name=networks
      Mds-Device-name=eth0
```

In the above hierarchy, the host consists of collections of devices. Note that the `Mds-Device-Group-name` (as well as the `Mds-Host-name` and `MDS-Software-Deployment`) objects are all structural types. The `Mds-Device-name` objects are auxiliary types. Note that there can be more than one auxiliary type for each category.

The GRIS structural class hierarchy consists of the following structural types defined in MDS:

```
Mds
    Attr: Mds-validfrom (like createtime)
    Attr: Mds-validto (accuracy metadata)
    Attr: Mds-keepto (discard metadata)
MdsHost
MdsDevice
MdsDeviceGroup
MdsSoftwareDeployment
Every MDS object: name, time metadata
```

Each of these structural types has an attribute called `name`, which does not identify an attribute instance, but a set of attributes.

The following are examples of types in the GRIS auxiliary class:

```
MdsCpu
    Attr: Mds-Cpu-vendor               - Once per CPU
    Attr: Mds-Cpu-model
    Attr: Mds-Cpu-speedMHz
MdsCpuCache
    Attr: Mds-Cpu-Cache-L1kB           - Once per CPU
MdsCpuSmp
    Attr: Mds-Cpu-Smp-size             - Once per SMP
MdsCpuTotal
    Attr: Mds-Cpu-Total-count          - Once per MPP

MdsCpuFree                             - Once per SMP
    Attr: Mds-Cpu-Free-1minX100
    Attr: Mds-Cpu-Free-5minX100
    Attr: Mds-Cpu-Free-15minX100

MdsCpuTotalFree                        - Once per MPP
    Attr: Mds-Cpu-Total-Free-1minX100
    Attr: Mds-Cpu -Total-Free-5minX100
    Attr: Mds-Cpu -Total-Free-15minX100
```

Note in the above examples that `MdsCpu`, `MdsCpuCache`, etc. are the class names, and `Mds-Cpu-vendor`, `Mds-Cpu-model`, etc. are the attribute names. LDAP considers the whole string during processing.

Note also that the 1, 5, and 15 minute averages above are normalized by cpu count (fast uptime). LDAP does not support values with units, so MDS embeds it into the attribute type.

MDS 2.2 Core GRIS Providers Hierarchy

The MDS 2.2 information model represents the physical and logical components of a compute resource as a hierarchy of elements. There are only a small number of elemental types, corresponding to LDAP structural object classes and representing little more than their names:

```
class MdsVo
  contains attr Mds-Vo-name
class MdsHost
  contains attr Mds-Host-hn
class MdsDevice
  contains attr Mds-Device-name
class MdsDeviceGroup
  contains attr Mds-Device-Group-name
...
```

A complementary set of "auxiliary" types adds information about the particular elemental instances. The LDAP auxiliary types are special in that they can be added to a structurally typed object to extend it with more information. The MDS 2.2 information model uses this feature to merge information "upward" in the object tree---while a leaf node may contain information about a single resource instance, a parent node may contain the merged information about several instances:

```
dn: Mds-Device-Group-name=memory, ...
  objectclass: MdsMemoryRamTotal
  objectclass: MdsMemoryVmTotal
  objectclass: MdsDeviceGroup
  Mds-Device-Group-name: memory
  Mds-validfrom: 200110030128.12Z
  Mds-validto: 200110030128.12Z
  Mds-keepto: 200110030128.12Z
  Mds-Memory-Ram-Total-sizeMB: 751
  Mds-Memory-Ram-Total-freeMB: 642
  Mds-Memory-Vm-Total-sizeMB: 1600
  Mds-Memory-Vm-Total-freeMB: 1592
  Mds-Memory-Ram-sizeMB: 751
  Mds-Memory-Ram-freeMB: 642
  Mds-Memory-Vm-sizeMB: 1600
  Mds-Memory-Vm-freeMB: 1592

dn: Mds-Device-name=physical memory, Mds-Device-Group-name=memory, ...

  objectclass: Mds
  objectclass: MdsDevice
  objectclass: MdsMemoryRam
  Mds-Device-name: physical memory
  Mds-Memory-Ram-sizeMB: 751
  Mds-Memory-Ram-freeMB: 642
  Mds-validfrom: 200110030128.12Z
  Mds-validto: 200110030128.12Z
  Mds-keepto: 200110030128.12Z
```

```
dn: Mds-Device-name=virtual memory, Mds-Device-Group-name=memory, ...

    objectclass: Mds
    objectclass: MdsDevice
    objectclass: MdsMemoryVm
    Mds-Device-name: virtual memory
    Mds-Memory-Vm-sizeMB: 1600
    Mds-Memory-Vm-freeMB: 1592
    Mds-validfrom: 200110030128.12Z
    Mds-validto: 200110030128.12Z
    Mds-keepto: 200110030128.12Z
```

This ability to merge multiple types allows the parent object to reflect the different types of each children, and therefore contain data from both of them.

The GRIS service provides a hierarchy of such objects for a Symmetric MultiProcessor (SMP) compute resource, merging data upward to the "host object" which contains all information about the host. This merged object is convenient for search filters expressing constraints on multiple data, but it loses some information due to the LDAP data model's inability to distinguish instances of a particular attribute value. For example, if an SMP had two CPUs with speed X, and one CPU with speed Y, the host object would only represent that there is at least one CPU of each speed. One would have to visit the underlying CPU device objects to determine how many (or which) CPUs have a particular speed value.

The core MDS 2.2 GRIS providers create a hierarchy with the following structure:

GRIS Host Objects
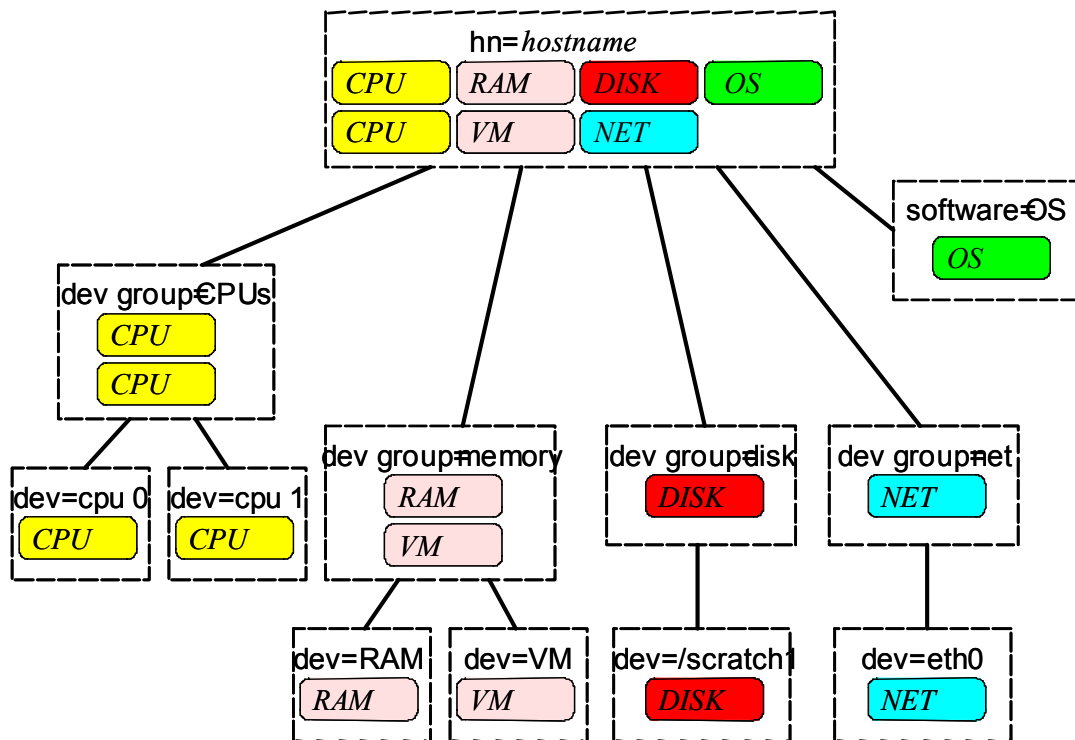
Where the well-known MdsDeviceGroup object names "processors", "memory", "filesystems", and "networks" act as collection points for the instances of devices in those categories. The well-known MdsSoftwareDeployment name "operating system" references information about the booted operating system software on the resource.

The schema design includes support for clusters of SMPs, i.e., distributed memory processors, via other compute elements such as MdsHostNetnode and MdsHostNode to represent networked (or hidden) elements of a cluster, each of which may be an SMP resource. However, the core providers with MDS 2.2 do not include the platform-specific probes necessary to efficiently extract information from such systems.

An index of schema definitions and descriptions of object classes and attribute types can be found in *MDS 2.2 Schemas*.

MDS DIT Namespaces

In the MDS version of the LDAP DIT, there are two reserved name spaces, as follows:

```
Mds-vo-name=local
Mds-vo-name=site
```

The `vo` DN represents the virtual organization described in Chapter 2, *Overview of MDS*. `vo=local` references the GRIS, and `vo=site` references the GIIS. (GRIS and GIIS are described in more detail in Chapter 2.)

The GIIS creates a subtree for each information provider that publishes data into MDS. The GRIS represents physical resources and uses namespaces to represent them – computers, file systems, network interfaces, etc.

Within a `vo`, the organization of namespaces depends on the specifics of the `vo`.

Within the MDS DIT, container names are used to represent groups of entries; then host objects are above the groups. Collection and composition of objects appear as you go upward in the tree.

Hierarchies of GIIS and GRIS resources are described in *MDS 2.2: Creating a Hierarchical GIIS*.

Chapter 5. MDS Security Configuration

This chapter describes the concepts and functions of MDS security and the procedures required to configure MDS security for your Grid computing environment.

The purpose of this chapter is to educate you about MDS security so that you can speak knowledgeably about it with your systems administrator. If you are (or intend on becoming) the administrator for your organization, this chapter will provide initial training for you in MDS security concepts and functions. If you do have an administrator for your organization, please contact them for the specific security requirements in your environment.

This chapter assumes that the Globus Toolkit and MDS have been installed and that default security features have been configured for your environment. Therefore, this chapter can serve as a reference to MDS security concepts and procedures to help you understand them, or to help you reinstall or revise your MDS security setup.

Additional information on Globus Toolkit security is available from the following:

- [Appendix A, Globus Toolkit Security Overview](#)

- Globus web site, at www.globus.org/security.

- *Globus Toolkit 2.2 Installation Instructions*, at
  http://www.globus.org/gt2.2/install.html

- *Globus Toolkit 2.2 Administrator's Guide*, at http://www.globus.org/gt2.2/admin/.

- *Globus Toolkit Tutorials* (slide presentations), at
  www.globus.org/about/events/US_tutorial/index.html.

Overview

This chapter is organized into the following sections:

- MDS Security Requirements – describes security dependencies and relationships with the Globus Toolkit and other software/hardware components, the LDAP service certificate, security-related configuration files, other files required for authentication and authorization, and related environment variables.

- MDS Security Setup Procedures – describes how to set the environment, obtain required certificates, verify installation, and be set up to perform authenticated queries.

- MDS Security Operations – provides an overview of how to configure (or reconfigure) the MDS security environment to grant specific access rights for various users to particular resources. Describes the use of Access Control Lists (ACLs), certificates, proxies, and related files to change authentication and authorization types and levels.

- MDS Queries – describes how to perform authenticated and non-authenticated queries. Provides pointers to documentation for interpreting command output and security-related messages, and for handling queries that return unexpected results because of lack of proper authentication and/or authorization.

MDS Security Requirements

This section describes the external and internal software components, directories, files, and environment variables comprising the MDS security configuration.

Security Dependencies/Relationships

As part of the Globus Toolkit, all of the standards and extensions to standards and protocols on which Toolkit security is based are applicable to MDS as well as to the other Toolkit components. As described in *Appendix A, Globus Toolkit Security Overview*, these include the Grid Security Infrastructure (GSI), Generic Security Service API (GSS-API), Secure Sockets Layer/Transport Layer Security (SSL/TLS), and X.509.

OpenLDAP

MDS 2.2 uses OpenLDAP Version 2.0.22, which implements LDAP Version 3. The security in OpenLDAP is provided by the Simple Authentication and Security Layer (SASL), which also uses GSS-API. SASL is a method for adding authentication support to connection-based protocols. MDS 2.2 uses Cyrus SASL Version 1.5.27. SASL requires libraries that are dynamically loaded at runtime.

SASL is a convenient generic interface for secure application development. By itself, SASL does not provide any security. It relies on underlying technologies to provide the actual identity authentication and message protection services desired by applications communicating over a network. Applications may install and request the use of particular mechanisms or use a default mechanism provided by the SASL implementation.

As described in *Appendix A, Globus Toolkit Security Overview*, the GSS-API security mechanism for MDS 2.2 is GSI, which enables the use of Globus certificates and grid-mapfiles to provide authorization services to information objects.

MDS 2.2 also uses OpenSSL Version 0.9.6b. OpenSSL provides the Secure Sockets Layer (SSL) implementation used by the GSI. OpenSSL is an open-source implementation of SSL used to build the GSS-API.

MDS runs on the OpenLDAP slapd directory server. Consequently, MDS supports the same security features as OpenLDAP and slapd, as follows:

- Authentication services through Cyrus SASL, which in turn supports GSS-API.

- Privacy and integrity protections through the use of TLS. slapd's TLS implementation utilizes OpenSSL software.

- Access control to database entries, based on LDAP authorization information, IP address, domain name, and other criteria. slapd supports both static and dynamic access control information.

The security permissions an information 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, or the provider must implement its own privilege elevation, e.g., by running a setuid executable. Refer to the *MDS 2.2 GRIS Specification Document: Creating New Information Providers* for details on custom information providers.

GRIS and GIIS Considerations

MDS 2.2 uses a consolidated slapd instance for GRIS and GIIS services on a given machine. This requires only a single port, which should be 2135. While it is recommended that port 2135 be used for all MDS services, you can select alternate ports by editing all the 2135 occurrences in $GLOBUS_LOCATION/etc/*.conf. See the *MDS 2.2 Configuration Files* web page for more details.

When querying a GIIS or GRIS on a slapd server, both authentication and authorization need to be considered.

All queries to the slapd server are authenticated through GSI via SASL unless the user specifies the `-x` option (anonymous bind) on the `grid-info-search` (or `ldapsearch`) command. (See *Chapter 3, Using MDS*, for more details on `grid-info-search`.) The authentication process checks the user's identity to make sure they are who they say they are; this authentication occurs in the slapd server's front end.

The authentication process follows a sequence of checking the settings of various environment variables (described later in this chapter) and making sure that the proper proxy and certificate files are specified by these variables. If the authentication sequence fails at any point, the query will not return the requested information.

After authentication by the slapd server, queries are authorized by the GIIS and GRIS implementations in the back end of the server. For queries to any GIIS or GRIS, authorization is performed against the grid-mapfile to make sure that the user is allowed to access the information they are requesting. Authorization against the grid-mapfile depends on the setting of the `anonymousbind` parameter in the grid-info-slapd.conf file. (The grid-mapfile and grid-info-slapd.conf files are described later in this chapter.)

If `anonymousbind` is set to `no`, authorization is performed against the grid-mapfile. Queries with the `grid-info-search` command should be entered without the `-x` option in this case; otherwise an error message will result. If `anonymousbind` is set to `yes`, no authorization is performed against the grid-mapfile. Queries with the `grid-info-search` command can be entered either with or without the `-x` option in this case.

Mutual Authentication

MDS provides mutual authentication between GIIS and GRIS servers as well as between GIIS servers in a hierarchy. This feature is intended to keep rogue servers from affecting a GIIS hierarchy. Mutual authentication is performed on outbound channels, as in a server registering to the one above it in a hierarchy.

Mutual authentication is enabled via the `bindmethod` parameter in the grid-info-resource-register.conf file. This parameter specifies the binding method from the upper-level GIIS as one of the following:

AUTHC-ONLY: Bind only with GSI authentication.
AUTHC-FIRST: Try authentication first; if that doesn't work, use anonymous binding. Authentication is tried first with every registration attempt.
ANONYM-ONLY: Use anonymous binding only. This is the default.

Note that the `bindmethod` parameter is required in the grid-info-resource-register.conf file. Without this parameter in the file, registration will not work.

Mutual authentication works only for servers in an MDS 2.2 environment. That is, there cannot be mutual authentication between MDS 2.1 and 2.2 resources.

A related policy statement may be included in the grid-info-site-policy.conf file to restrict mutual authentication to a particular host or hosts and to a specific port on a host or hosts.

See *Configuration Files* below for more information on grid-info-resource-register.conf and grid-info-site-policy.conf. Refer to *MDS 2.2: Creating a Hierarchical GIIS* for more information on policy statements.

Platforms

MDS security operations are also dependent on the particular operating system used in your environment. For example, SASL requires dynamically loaded libraries at runtime. This dependency is an issue for environments that do not support dynamic libraries, such as the Unicos operating system. Please contact your system administrator for any platform-related MDS security issues in your environment, or contact the MDS development group.

Service Certificate

For authenticated access to GSI-enabled services of the Globus Toolkit, you need a user certificate. If you are using Toolkit components (other than MDS) that require a gatekeeper for your host, you need a host certificate as well.

For authenticated access to MDS, the server needs an X.509-compatible LDAP service certificate. You obtain an LDAP service certificate (as well as the other types of certificates) via the `grid-cert-request` command, as described under *MDS Security Setup Procedures* later in this chapter.

Configuration Files

The MDS 2.2 installation creates several configuration files of interest to system administrators and programmers. These files are located in $GLOBUS_LOCATION/etc. Three of these configuration files are of particular importance to MDS security: grid-info-slapd.conf, grid-info-site-policy.conf, and grid-info-resource-register.conf.

The grid-info-slapd.conf file sets basic access control rules and sets anonymous binding. Additionally, this file designates the GIIS and GRIS provider components to OpenLDAP, establishes LDAP and MDS information schema, and defines back ends supported by the slapd server. The `anonymousbind` and `access to` parameters in this file set anonymous binding and access control, respectively. Refer to *MDS 2.2 Configuration Files* for an example grid-info-slapd.conf file.

The grid-info-site-policy.conf file controls the acceptance of registration messages by a GIIS. This file can be used to create an open policy where all registrants are welcome, or a closed system whereby only specified resources can register with a GIIS. The default is for the GIIS to accept registrations only from itself, and from port 2135. This file must be modified from the default in a hierarchical GIIS environment. Refer to *MDS 2.2 Configuration Files* for an example grid-info-site-policy.conf file, and to *MDS 2.2: Creating a Hierarchical GIIS* for more details on the use of this file.

The grid-info-resource-register.conf file lists the GIIS servers to which a GRIS or "child" GIIS will register directly. The default is to register to the local GIIS on the host. This file identifies host names, ports, and time values that control registration messages from a GRIS or GIIS to a GIIS server. Refer to *MDS 2.2 Configuration Files* for an example grid-info-resource-register.conf file, and to *MDS 2.2: Creating a Hierarchical GIIS* for more details on the use of this file.

An additional configuration file, grid-info-server-env.conf, sets up the X509* and other environment variables for use in mutual authentication.

Other Important Files

The following directories and files are also relevant to the MDS (and Globus Toolkit) security configuration:

- /etc/grid-security
- /etc/grid-security/ldap
- $GLOBUS_LOCATION/etc/ldap
- /etc/grid-security/certificates
- $HOME/.globus
- /tmp
- $GLOBUS_LOCATION/libexec/grid-info-slapd and
  $GLOBUS_LOCATION /etc/grid-info-server-env.conf

These directories and files are described in the following sections.

/etc/grid-security

The /etc/grid-security directory holds configuration information and private information used by the GSI libraries. It should be on local disk (i.e., not NFS-mounted). This directory and its contents should not be writable by any untrusted users (generally this means only by root). All of this directory's contents may be world-readable.

Relevant to MDS in /etc/grid-security is the grid-mapfile, described below.

grid-mapfile

This file defines a set of authorized MDS clients by mapping grid subject names (Distinguished Names) to local user account names. This mapping may be one-to-one or several-to-one, but cannot be several-to-several.

It is the system administrator's responsibility to generate the grid-mapfile in the /etc/security directory, and to verify that the Distinguished Names (DNs) in the file are owned by and match the local user names.

The grid-mapfile is a plain text file, containing quoted DNs (credential names that are the subjects of X.509 certificates) and unquoted local user names.

The default location of the grid-mapfile is in /etc/grid-security. The GRIDMAP environment variable in the /etc/grid-info-server-env.conf (described later in this section) should point to this location or wherever the file resides.

The GIIS back end checks the grid-mapfile only when `anonymousbind` is set to `no` (i.e., security is enabled) in the grid-info-slapd.conf file. The following is an example grid-mapfile:

```
"/O=Grid/O=Globus/OU=isi.edu/CN=Jake Clayman" jake
"/O=Grid/O=Globus/OU=isi.edu/CN=Marv Peppler" marv
"/O=Grid/O=Globus/OU=isi.edu/CN=Diaz Santiago" diaz
"/O=Grid/O=Globus/OU=isi.edu/CN=Cynthia Juarez" cynthia
"/O=Grid/O=Globus/OU=isi.edu/CN=David Chung" david
```

/etc/grid-security/ldap

The /etc/grid-security/ldap directory contains the LDAP service certificate and key, in the files described below.

ldapcert.pem

This file contains the X.509 certificate used by the LDAP server during mutual authentication with connecting clients. This file is a matched pair with the service key in ldapkey.pem described below. The ldapcert.pem file should be world-readable. ldapcert.pem is set as the value of the X509_USER_CERT environment variable in the grid-info-server-env.conf configuration file described later in this section.

ldapkey.pem

This file contains the X.509 private key used by the LDAP server during mutual authentication with connecting clients. The private key in this file corresponds to the service certificate in ldapcert.pem described above. The ldapkey.pem file must be readable by the user account that runs MDS. ldapkey.pem is set as the value of the X509_USER_KEY environment variable in the grid-info-server-env.conf configuration file described later in this section.

Note that ldapcert.pem and ldapkey.pem are the equivalent of hostcert.pem and hostkey.pem, which are used by other Globus Toolkit components (such as GRAM) that run a gatekeeper and require a separate host certificate and key. If required, the host certificate and key must be requested separately via the `grid-cert-request` command.

Name/Location Conformance for ldapcert.pem and ldapkey.pem

Note that it is the system administrator's responsibility to ensure that the name and location of the service certificate and service key files conform to the X.509 environment variable values set in the grid-info-server-env.conf configuration file.

This conformance can be achieved by using the `-service`, `-dir`, and `-prefix` arguments on the `grid-cert-request` command to specify, respectively, an LDAP service certificate, a directory for the service certificate and key, and file names for the service certificate and key. The use of `grid-cert-request` is described later in this chapter under *MDS Security Setup Procedures*. Alternately, you can manually change the certificate and key file names as required and move them to the appropriate location.

$GLOBUS_LOCATION/etc/ldap

If MDS cannot find the LDAP service certificate and key files (ldapcert.pem and ldapkey.pem) in the /etc/grid-security/ldap directory as described above, it will look for them in the $GLOBUS_LOCATION/etc/ldap directory. You'll need to put the certificate and key files in this directory if it is to be used. Name/location conformance for the service certificate and key files also applies here.

/etc/grid-security/certificates

The /etc/grid-security/certificates directory contains the X.509 certificates of all trusted Certificate Authorities (CAs) and their signing policies. The signing policies are the names for which the CAs issue certificates.

For a user to authenticate with GSI to any GSI-based service, the certificate of the CA that signed the user's certificate and the related signing policy must be in this directory on the server. Similarly, the certificate of the CA that signed the service certificate and the related signing policy must be in this directory on the client. All the contents of this directory are used by both GSI-based clients and GSI-based services and should be world-readable. Note that /etc/grid-security/certificates is the default directory. If necessary, change the value of the X509_CERT_DIR environment variable to point to the directory you are using.

The CA's certificates are stored in files whose names are based on a hash of the CA identity for quick location, for example, 42864e48.0. The signing policy files are named after the same hash with ".signing_policy" appended, for example, 42864e48.signing_policy.

Refer to *Adding a New Trusted CA to a Globus Installation* (www.globus.org/security/v2.0/adding_trusted_ca.html) for more information.

$HOME/.globus

The security-related files in the user's home directory are as follows:

usercert.pem

This file contains the user's certificate, which holds the subject name, public key, and CA signature. See the *MDS Security Setup Procedures* section later in this chapter for an example of a user certificate contained in this file.

userkey.pem

This file contains the user's private key, readable only by the user and encrypted using the user's passphrase. See the *MDS Security Setup Procedures* section later in this chapter for an example of a user's private key contained in this file.

$/tmp

The temporary directory can contain one or more proxy files, which are temporary files containing unencrypted proxy private keys and certificates (readable only by the user's account).

$GLOBUS_LOCATION/libexec/grid-info-slapd and $GLOBUS_LOCATION/etc/grid-info-server-env.conf

In $GLOBUS_LOCATION/libexec, grid-info-slapd is a script called when MDS is started with the `globus-mds start` command. (The grid-info-slapd script should not be confused with the grid-info-slapd.conf configuration file.) The grid-info-slapd script calls the grid-info-server-env.conf configuration file to set up the MDS environment variables such as those for the service certificate and key. The grid-info-slapd script then calls the slapd server, which reads the grid-info-slapd.conf file and determines all the other configuration files to read.

The service certificate and key are set by default in $GLOBUS_LOCATION/etc/grid-info-server-env.conf to /etc/grid-security/ldap/ldapcert.pem and ldapkey.pem.

The following is an example of the grid-info-slapd script:

```
#! /bin/sh

. ${GLOBUS_LOCATION}/libexec/globus-script-initializer

. ${GLOBUS_LOCATION}/etc/grid-info-server-env.conf

exec ${GLOBUS_LOCATION}/libexec/slapd "$@"
```

The following is an example of the grid-info-server-env.conf file:

```
#! /bin/bash

. ${GLOBUS_LOCATION}/libexec/globus-script-initializer

if [ ! -z "${GRID_SECURITY_DIR}" ] &&
[ -r "${GRID_SECURITY_DIR}/ldap/ldapkey.pem" ] &&
[ -r "${GRID_SECURITY_DIR}/ldap/ldapcert.pem" ] ; then
X509_USER_CERT=${GRID_SECURITY_DIR}/ldap/ldapcert.pem
X509_USER_KEY=${GRID_SECURITY_DIR}/ldap/ldapkey.pem
elif [ -r "/etc/grid-security/ldap/ldapkey.pem" ] &&
[ -r "/etc/grid-security/ldap/ldapcert.pem" ] ; then
X509_USER_CERT=/etc/grid-security/ldap/ldapcert.pem
X509_USER_KEY=/etc/grid-security/ldap/ldapkey.pem
secconfdir="/etc/grid-security"
elif [ -r "${GLOBUS_LOCATION}/etc/ldap/ldapkey.pem" ] &&
[ -r "${GLOBUS_LOCATION}/etc/ldap/ldapcert.pem" ] ; then
secconfdir="${GLOBUS_LOCATION}/etc"
X509_USER_CERT=${GLOBUS_LOCATION}/etc/ldap/ldapcert.pem
X509_USER_KEY=${GLOBUS_LOCATION}/etc/ldap/ldapkey.pem
fi
# it is possible that we may reach the end of this if without matching,
# if no certificate/key pair found anywhere.
# This is ok.

X509_RUN_AS_SERVER=true

GRIDMAP=${sysconfdir}/grid-mapfile

LD_LIBRARY_PATH=${GLOBUS_LOCATION}/lib:${LD_LIBRARY_PATH}
SASL_PATH=${GLOBUS_LOCATION}/lib/sasl

export X509_USER_CERT
export X509_USER_KEY
export X509_RUN_AS_SERVER
export GRIDMAP
export LD_LIBRARY_PATH
export SASL_PATH
```

The environment variables set in this file are described in the next section.

It is the system administrator's responsibility to ensure that the name and location of the service certificate and service key files conform to the X.509 environment variable values set in the grid-info-server-env.conf configuration file.

This conformance can be achieved by using the -service, -dir, and -prefix arguments on the grid-cert-request command to specify, respectively, an LDAP service certificate, a directory for the service certificate and key, and file names for the service certificate and key. The use of grid-cert-request is described later in this chapter under *MDS Security Setup Procedures*. Alternately, you can manually change the certificate and key file names as required and move them to the appropriate location.

Environment Variables

The environment variables used in the Globus Toolkit and MDS security configuration are described in the following paragraphs. The only variable you need to set is GLOBUS_LOCATION. All other environment variables are set automatically by MDS in the /etc/grid-info-server-env.conf file.

GLOBUS_LOCATION

This variable points to the directory in which the Globus Toolkit and related services are installed. This directory should be owned by the user named "globus." The GSI security libraries use GLOBUS_LOCATION as one place to look for the trusted certificates directory. The location $GLOBUS_LOCATION/share/certificates is used if X509_CERT_DIR is not set and /etc/grid-security and $HOME/.globus/certificates do not exist.

X509_USER_CERT

This variable points to the location of the certificate files. For users, the default is $HOME/.globus/usercert.pem. For servers, the default is /etc/grid-security/hostcert.pem. This variable also defines the path to the ldapcert.pem file in the grid-info-server-env.conf file (described above) that is run when MDS is started.

X509_USER_KEY

This variable points to the location of the private key files. For users, the default is $HOME/.globus/userkey.pem. For servers, the default is /etc/grid-security/hostkey.pem. This variable also defines the path to the ldapkey.pem file in the grid-info-server-env.conf file (described above) that is run when MDS is started.

X509_CERT_DIR

This variable points to the location of the trusted certificates directory, which contains CA certificates and signing policy files. The default is /etc/grid-security/certificates (described earlier in this chapter).

X509_USER_PROXY

This variable points to the location of the user proxy credentials. The default is /tmp/x509up_u<uid>. This variable should be left undefined unless you are an expert user and know exactly what you are doing.

GRIDMAP

This variable points to the location of the grid-mapfile. The default is /etc/grid-security/grid-mapfile. If the grid-mapfile is in another location, make sure that GRIDMAP points to it.

X509_USER_DELEG_PROXY

This variable is set by the GSI security libraries to point to the location of credentials that it receives during proxy delegation. Application servers usually then copy the value of this variable to X509_USER_PROXY, and users generally never see it. Manually setting this value has no effect.

X509_RUN_AS_SERVER

If this variable is set (to any value), it causes the GSI security libraries to ignore proxy credentials and use host certificates instead, unless X509_USER_PROXY is explicitly set. The intent is for X509_RUN_AS_SERVER to be used with servers that should always use a given host certificate and private key.

X509_CERT_FILE

This variable points to a file of trusted CA certificates. X509_CERT_FILE can be used to explicitly force the GSI security libraries to use a given file containing trusted CA certificates. Use of this variable is discouraged due to lack of use and testing.

SASL_PATH

This variable points to the location of the SASL plug-in libraries used for authentication. SASL_PATH is in the $GLOBUS_LOCATION/libexec/grid-info-slapd script described above and has a default value of `${GLOBUS_LOCATION}/lib/sasl`.

LD_LIBRARY_PATH

This variable points to the location of shared system libraries. LD_LIBRARY_PATH is in the $GLOBUS_LOCATION/libexec/grid-info-slapd script described above and has a default value of:

```
LD_LIBRARY_PATH=${GLOBUS_LOCATION}/lib:${LD_LIBRARY_PATH}
```

MDS Security Setup Procedures

This section describes the procedures you need to follow to set up (or revise) your security environment for MDS. This section assumes that the Globus Toolkit and MDS have already been installed in your environment, as described in the *Globus Toolkit 2.2 Installation Instructions*.

For purposes of illustration, let's also assume that this setup involves a client (a GRIS) running on one host and a server (a GIIS) running on another host. (The procedure would essentially be the same if the GRIS and GIIS were on the same machine.)

The essence of MDS security setup is to obtain the required certificates and to create a proxy for authenticated access. The basic steps are as follows:

1. Set the environment.
2. Obtain necessary certificates.
3. Create a proxy for authenticated access.
4. Start MDS and verify authenticated access.

These steps are described in detail in the following sections.

1. Set the Environment

You need to set your environment before using MDS (or any other) commands from the Globus Toolkit. You need to do this on both the client and server machines.

First, make sure you have set GLOBUS_LOCATION to the location of your Toolkit installation. This directory should be owned by a user named "globus."

There are two environment scripts, called GLOBUS_LOCATION/etc/globus-user-env.sh and GLOBUS_LOCATION/etc/globus-user-env.csh. Enter the one that corresponds to the type of shell you are using.

For example, in csh or tcsh, you enter:

```
source $GLOBUS_LOCATION/etc/globus-user-env.csh
```

In sh, bash, ksh, or zsh, you enter:

```
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

2. Obtain Necessary Certificates

For authenticated access to MDS, each Globus Toolkit user requires a user certificate and each server requires an LDAP service certificate. These certificates allow the user and the server, respectively, to participate in the authentication process. (Each host requires a host certificate if it is running a gatekeeper or other service besides MDS.)

As described in the *Globus Toolkit 2.2 Installation Instructions*, you obtain a user, host, or service certificate with the `grid-cert-request` command. For your user certificate, you issue this command from the client machine. For the LDAP service certificate required by MDS, you issue this command from the server machine in the following form:

```
% grid-cert-request –service ldap –host my.host.fqdn
```

In the command format above:

- `-service ldap` specifies that a certificate be created for the LDAP server.
- `-host my.host.fqdn` specifies the fully qualified domain name of the host that will run the LDAP server.

For a description of all the options available for `grid-cert-request`, enter the command with the `-help` or `-usage` options.

After running this command, the key is generated immediately and placed in /etc/grid-security/ldap/ldapkey.pem. The key file should be readable only by its owner.

The following specific example shows a request for an LDAP service certificate for dc-user.isi.edu and the resulting output:

```
$ grid-cert-request -service ldap -host dc-user.isi.edu

Using configuration from /etc/grid-security/globus-host-ssl.conf
Generating a 1024 bit RSA private key
.....................+++++
.....................+++++
writing new private key to '/etc/grid-security/ldap/ldapkey.pem'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
```

```
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:Level 1 Organization [Globus]:Name (e.g.,
John M. Smith) []:

A private ldap key and a certificate request has been generated
with the subject:

/O=Grid/O=Globus/CN=ldap/dc-user.isi.edu

----------------------------------------------------------

The private key is stored in /etc/grid-security/ldap/ldapkey.pem
The request is stored in /etc/grid-security/ldap/ldapcert_request.pem

Please e-mail the request to the Globus CA ca@globus.org
You may use a command similar to the following:

 cat /etc/grid-security/ldap/ldapcert_request.pem | mail ca@globus.org

Only use the above if this machine can send AND receive e-mail. if not,
please
mail using some other method.

Your certificate will be mailed to you within two working days.
If you receive no response, contact Globus CA at ca@globus.org
```

After you enter the `grid-cert-request` command for the LDAP service certificate, do the following:

1. Rather than following the e-mailing instructions in the `grid-cert-request` output, instead use your regular user mail agent to send an e-mail to ca@globus.org; copy and paste the contents of /etc/grid-security/ldap/ldapcert_request.pem into the message. Do not include this file as an attachment. Do not be concerned about the 0 byte file named ldapcert.pem that is generated by the `grid-cert-request` program. It is just a place holder; your real certificate will arrive in the mail.

2. Within two business days, your LDAP service certificate will be mailed to you. When it arrives, read the contents of the e-mail and save the entire e-mail to /etc/grid-security/ldap/ldapcert.pem. This file must be owned by the user account that will run MDS. The file should have permissions 600.

Note that the Globus CA mentioned above is provided for convenience only. For the most effective security, your environment should have its own CA.

Refer to the *Globus Toolkit 2.2 Installation Instructions* for more details on e-mailing your certificate request and receiving your certificate via return e-mail.

To see the details of existing certificates, use the `grid-cert-info` command. Enter the `-help` and `-usage` options for details on this command.

3. *Obtain a Proxy for Authenticated Access*

After you have obtained your LDAP service certificate, the next step is to verify authenticated access to MDS. You do this by creating a proxy with the `grid-proxy-init` command, issued on the client machine. You also need to check the grid-mapfile (in etc/grid-security on the server) to make sure that it correctly identifies a set of authorized MDS clients (by mapping grid subject names (Distinguished Names) to local user account names) for your environment. See the grid-mapfile section earlier in this chapter for more information.

Enter the `grid-proxy-init` command as in the following example:

```
% grid-proxy-init
```

The example above does not require the `-cert` and `-key` options, because the user certificate and key are in their standard locations: ~/.globus/usercert.pem for the certificate and ~/.globus/userkey.pem for the key. For a description of all the options available for `grid-proxy-init`, enter the command with the `-help` or `-usage` options.

The output of the `grid-proxy-init` command appears as in the following example:

```
Your identity: /O=Grid/O=Globus/OU=isi.edu/CN=Leon Kuntz
Enter GRID pass phrase for this identity:
Creating proxy ...................................................... Done
Your proxy is valid until Mon Apr  1 23:37:45 2002
```

You enter your passphrase in response to the prompt, and your proxy is created as shown above. Note that if you forget your passphrase, you'll need to request a new user certificate via `grid-cert-request`.

A file with the name of, for example, X509up_u3491 (in /tmp) shows you that the proxy is running.

To see the details of existing proxies, use the `grid-proxy-info` command. Enter the `-help` and `-usage` options for details on this command.

The creation of this proxy effectively "logs you in" to MDS. If you need to terminate the proxy for some reason before the period for which the proxy is valid (default or otherwise) expires, you can enter `grid-proxy-destroy` from the client machine.

Note that after your proxy expires (or after your log out of your client), you need to re-enter `grid-proxy-init` as described above. By default, a proxy is good for 12 hours; you can change the default with the `-h` option on the `grid-proxy-init` command.

4. Start MDS and Verify Authenticated Access

The last step in your MDS security setup is to make sure that your configuration is functioning properly.

From the server, start MDS 2.2 with the following command:

```
% GLOBUS_LOCATION/sbin/globus-mds start
```

This command starts the slapd server for the GRIS. It does not require the GLOBUS_LOCATION environment variable to be set. (Note that `globus-mds start` is backwards-compatible with `SXXgris start`.)

Next, enter the following security-enabled query from the client, to output the details of all objects on the GRIS:

```
% grid-info-search -b "Mds-Vo-name=local,o=Grid"
```

See the *MDS Queries* section later in this chapter for more details on performing authenticated and non-authenticated queries.

MDS Security Operations

This section provides an overview of how to configure (or reconfigure) the MDS security environment to grant specific access rights for various users to particular resources. Authentication and authorization types and levels can be granted and changed through the use of Access Control Lists (ACLs), certificates, proxies, and related files.

Access Control

The essence of security is authentication (are you who you say you are) and authorization (are you allowed to access the resources you are requesting for the tasks you want to perform).

Authorization grants access based on authenticated identity. For MDS 2.2, authorization is performed through screening against the grid-mapfile and static ACLs (stored in the grid-info-slapd.conf configuration file).

The access control process is illustrated in the following diagram:

```
                          ┌──────────────┐
                          │Authentication│
                          └──────────────┘
              Anonymous                    GSI

         ┌──────────┐              ┌──────────┐
         │  Allow   │              │  Allow   │
         │anonymous?│              │anonymous?│
         └──────────┘              └──────────┘
      No              Yes     Yes              No

   Reject              ┌──────────────┐   ┌──────────┐
                       │              │   │In mapfile?│
                       │ Process query│   └──────────┘
                       │(for each datum)│          No
                       │              │      Yes     Reject
                       └──────────────┘

         ┌──────────┐
         │ Allowed  │
         │ access?  │
         └──────────┘
      Yes              No

   Yield datum         Skip
   in result           down

   Object dynamic
   ACLS
```

```
                                          generalized
                                          authentication subject
                                          (includes SASL/GSS-GSSAPI)
OpenLDAP:    read/write

<what>     <access>  by  <who>          location in another    interesting to
                                        LDAP directory          organizations
DIT object             LDAP groups      describing group        with LDAP
DIT tree                                of subjects             institutional
objectclass                                                     services
attributetype
```

grid-mapfile

The grid-mapfile resides in the /etc/grid-security directory and defines a set of authorized MDS clients by mapping grid subject names (Distinguished Names) to local user account names. The grid-mapfile is described in more detail earlier in this chapter, under *Other Important Files*.

Static Access Control

Static access control is provided by ACLs in the grid-info-slapd.conf file. Any changes to the ACLs in this configuration file require that the server be restarted before the changes take effect.

Within grid-info-slapd.conf, access control is set by the `access to` parameter. A simple example of this would be `access to * by * write`, which grants write access to all entities and attributes by all (anonymous and authenticated) users. You can also use the `access to` parameter to grant access much more specifically. For example:

```
access to * by dn="uid=/O=Grid/O=Globus/OU=isi.edu/(.*)" write
```

grants access to anyone with an ISI credential.

As described in detail in the *OpenLDAP 2.0 Administrator's Guide* (www.openldap.org/doc/admin/slapdconfig.html), the access directive in the slapd configuration file has the following form:

```
<access directive> ::= access to <what>
        [by <who> <access> <control>]+
```

where `<what>` selects the entries and/or attributes to which the access applies, `<who>` specifies which entities are granted access, and `<access>` specifies the type of access granted. Multiple `<who>` `<access>` `<control>` triplets are supported, allowing many entities to be granted different access to the same set of entries and attributes.

For the entities and attributes to which the access applies (`<what>`), entries can be selected by a regular expressing matching the entry's Distinguished Name or by a filter matching some attribute(s) in the entry.

For the entities being granted access (`<who>`), they can be specified as all users (`*`), anonymous (non-authenticated) users (`anonymous`), authenticated users (`users`), users associated with a target entry (`self`), or users matching a regular expression (`dn=<regex>`).

For the type of access granted (`<access>`), the levels of access can be specified as `none`, `auth` (bind), `compare`, `search`, `read`, or `write`. Refer to the *OpenLDAP 2.0 Adminstrator's Guide* for much more detail on specifying access control.

Delegated and Limited Proxies

Another way of controlling authentication and authorization is through the use of proxies.

A proxy certificate can be defined as a temporary binding of a new key pair to an existing user identity. The use of proxy certificates allows an entity to temporarily delegate their rights to remote processes or resources on the Internet.

A proxy can be delegated to produce a second-level proxy credential, and that delegated proxy can have its rights limited to control access to resources in specified ways.

Proxy delegation is the remote creation of a user proxy. This results in a new private key and an X.509 proxy certificate, signed by the original key. The following occur when a proxy is delegated:

- A new key pair is generated remotely on the server.
- The proxy certificate and public key are sent to the client.
- The client signs the proxy certificate and returns it.
- The server (usually) puts the proxy in the /tmp directory.

Delegation allows remote processes to act (authenticate) on behalf of the user; a remote process thereby can "impersonate" the user. Delegation also avoids having to send passwords or private keys across the network.

A full or unrestricted proxy is one that has been created by `grid-proxy-init` or is a proxy created from such a proxy by normal delegation mechanisms; all of its authority can be delegated. A limited proxy is one that is created from a full proxy when it is delegated with the limited delegation mechanism.

A limited proxy can be used to restrict the rights of a delegated proxy to a subset of those rights associated with the issuer. These restrictions can be used, for example, to only make requests to a specific server or only to authorize specific operations on specific resources. The certificate for the delegated proxy contains the embedded restriction policy.

During delegation, a client can elect to delegate only a limited proxy rather than a full proxy. For example, the GRAM (job submission) client does this. Each service decides whether it will allow authentication with a limited proxy. For example, the job manager service requires a full proxy; the GridFTP server allows either a full or a limited proxy to be used.

If necessary, delegation tracing can be performed to see through what entities a proxy certificate can be delegated. This can be an audit, to retrace footsteps, or an authorization, to deny from bad entities. Tracing can then add information to the signed proxy certificate about each entity to which the proxy is delegated. This does not guarantee proper use of the proxy, however; it just tells you which entities were purposely involved in a delegation.

For more information on proxies and delegation, refer to the security sections of the Globus web site: www.globus.org/security and www.globus.org/security/v2.0.

MDS Queries

This section describes how to perform authenticated and non-authenticated queries. This section also provides pointers to documentation for interpreting command output and security related messages, and for handling queries that return unexpected results because of lack of proper authentication and/or authorization.

Authenticated Queries

The MDS query command, `grid-info-search`, by default issues security-enabled (SASL/GSI-GSSAPI authentication turned on) queries. This means that the GSI security mechanisms described in this chapter are activated to verify your identity and the resources to which you have access. Mutual authentication via certificates, proxies, and public/private key encryption/decryption are part of this process; the security mechanisms also check the grid-mapfile to make sure that it contains an entry mapping your certificate

subject name to your local user name. Additional security operations may also be part of your particular environment.

Entering a search command without the `–anonymous` or `–x` option automatically performs a security-enabled query. For example, entering:

```
% grid-info-search –h giis-demo.globus.org –p 8463 –b 'Mds-Vo-
name=site, o=Grid'
```

performs a security-enabled query of all objects on the GIIS at host giis-demo.globus.org, on port 8463.

For more information on performing authenticated queries and on dealing with related issues and problems, refer to the Globus Toolkit Error FAQ and MDS FAQ pages, at www.globus.org/about/faq/errors.html and www.globus.org/mds/FAQ.html, respectively.

Anonymous Queries

The MDS query commands can also be used to issue security-disabled (SASL/GSI-GSSAPI authentication turned off) queries by using the `–anonymous` or `–x` option on the command. For example, entering:

```
% grid-info-search –x –h giis-demo.globus.org –p 8422 –b 'Mds-Vo-
name=local, o=Grid'
```

performs an anonymous query of all objects on the GRIS at host giis-demo.globus.org, on port 8422.

Depending on the specifics of the MDS security configuration in your environment, there may be little point in performing anonymous queries. Anonymous queries may be useful to give you a quick look at basic resources, but it may not be possible for you to performed detailed queries of specific resources on specific machines, particularly in a hierarchical GIIS environment. In some environments, you may not be able to perform any queries at all, or even start MDS. Consult with your system administrator about the possibility and value of anonymous access in your environment.

Appendix A. Globus Toolkit Security Overview

This appendix describes the basic concepts of Globus Toolkit security, including the Grid Security Interface (GSI), certificates, proxies, and programs for credential management.

Globus Toolkit Security Mechanism

The security mechanism used by the Globus Toolkit is the Grid Security Infrastructure (GSI), which enables the use of certificates and various files to provide authentication and authorization services. The Toolkit implements GSI protocols and APIs to address Grid security needs.

GSI is a library for providing generic security services for applications that will be run on the Grid. GSI provides programs to facilitate login to a variety of sites, while each site has its own flavor of security measures. GSI provides a means of simplifying multiple remote logins.

GSI enables secure authentication and communication over an open network. GSI provides a number of useful services for Grids, including mutual authentication and single sign-on. The single sign-on authentication service identifies a user for access to multiple Grid resources via one sign-on procedure. GSI also provides local control over access rights and mapping from global to local user identities.

Standards Used/Extended by GSI

GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) communication protocol. Extensions to these standards have been added for single sign-on and delegation (via X.509 proxy certificates). The Globus Toolkit's implementation of the GSI adheres to the Generic Security Service API (GSS-API), which is a standard API for security systems promoted by the Internet Engineering Task Force (IETF).

GSS-API is independent of any specific security implementation. It is the IETF standard for adding authentication, delegation, message integrity, and message confidentiality to applications, for secure communication between two parties over a reliable channel. GSS-API separates security from communication, which allows security to be easily added to existing communication code, effectively placing transformation filters on each end of the communication link. All of the Globus Toolkit components use GSS-API.

The Globus Toolkit provides GSS-API on GSI protocols, as well as various tools for credential management, login/logout, and so forth. The Toolkit also provides a wrapper around GSS-API, called globus-gss-assist, to make GSS-API easier to use. globus_gss_assist hides some of the gross details of GSS-API, conforms to Globus

Toolkit conventions, and still separates security from communication. Refer to
http://www.globus.org/Security/gss_assist.html for more information.

Certificates

A central concept in GSI authentication is the *certificate*. Every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service.

A GSI certificate contains four primary pieces of information, as follows:

- A subject name, which identifies the person or object represented by the certificate.

- The public key belonging to the subject.

- The identity of a Certificate Authority (CA) that has signed the certificate to verify that the public key and the identity both belong to the subject.

- The digital signature of the named CA.

The signature of the CA proves that the certificate came from the CA, vouches for the subject name, and vouches for the binding of the public key to the subject.

Note that a third party (a CA) is used to certify the link between the public key and the subject in the certificate. In order to trust the certificate and its contents, the CA's certificate must be trusted. The link between the CA and its certificate must be established via some non-cryptographic means, or else the system is not trustworthy. GSI certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the IETF.

Mutual Authentication

If two parties have certificates, and if both parties trust the CAs that signed each other's certificates, then the two parties can prove to each other that they are who they say they are. This is known as *mutual authentication*. The GSI uses the Secure Sockets Layer (SSL) for its mutual authentication protocol, which is described below. (SSL is also known by a new, IETF standard name: Transport Layer Security, or TLS.)

Before mutual authentication can occur, the parties involved must first trust the CAs that signed each other's certificates, and in addition must have copies of the CAs' certificates and trust that the certificates really belong to the CAs.

Private and Public Keys

Public key cryptography relies not on a single key (a password or a secret "code"), but on two asymmetric keys. These keys are numbers that are mathematically related in such a way that if either key is used to encrypt a message, the other key must be used to decrypt it. Also important is the fact that it is next to impossible to obtain the second key from the first one and/or any messages encoded with the first key.

By making one of the keys available publicly (a *public key*) and keeping the other key private (a *private key*), a person can prove their ownership of the private key simply by encrypting some data. If the data can be decrypted using the public key, the person must have used the private key to encrypt the message.

The core GSI software provided by the Globus Toolkit expects the user's private key to be in a file in the local computer's storage. To prevent other users of the computer from stealing the private key, the file that contains the key is encrypted via a password (also known as a *passphrase*). To use the GSI, the user must enter the passphrase required to decrypt the file containing their private key.

Certificate Types

For authenticated access to the Globus Toolkit, you need a certificate for yourself; i.e., a *user certificate*. Additionally, you need certificates for the different Globus Toolkit services you want to run.

If you plan on running your own gatekeeper, you also need a *host certificate*. When submitting a job through Globus, you authenticate against a gatekeeper, which runs on a remote host. The gatekeeper will map your certificate against a local user and start a job manager as that user. The job manager will start up and monitor the requested job. Gatekeepers and job managers are described in more detail later in this section.

In addition to your Globus Toolkit user certificate, authenticated access to MDS requires an X.509-compatible LDAP *service certificate*.

You obtain each of these certificates via the `grid-cert-request` command. Detailed procedures for doing so are described later in the *MDS Security Setup Procedures* section of *Chapter 5, MDS Security Configuration*.

Proxies

The GSI provides a *delegation* capability: an extension of the standard SSL/TSL protocol that reduces the number of times the user must enter their passphrase. If a Grid computation requires that several Grid resources be used (each requiring mutual authentication), or if there is a need to have agents (local or remote) requesting services

on behalf of a user, the need to re-enter the user's passphrase can be avoided by creating a *proxy*.

A proxy is defined as a temporary binding of a new key pair to an existing user identity. The use of proxies allows an entity to temporarily delegate their rights to remote processes or resources on the Internet.

Delegation permits the remote creation of a user proxy, which results in a new private key and an X.509 proxy certificate, signed by the original key. This allows a remote process to authenticate on behalf of the user, and avoids sending passwords or private keys across the network. The remote process in essence "impersonates" the user.

During delegation, a proxy can be designated as a "full" proxy, with the same access rights as those of the user certificate, or as a "limited" proxy, with a subset of those access rights. Limited or "restricted" proxies are described in more detail in the *MDS Security Operations* section of *Chapter 5, MDS Security Configuration*.

A proxy consists of a new certificate (with a new public key in it) and a new private key. The new certificate contains the user's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the user, rather than by a CA. The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies are temporary, local, short-lived credentials for use by Grid computations.

Therefore, a proxy certificate:

- Defines how a short-term, restricted credential can be created from a normal, long-term X.509 credential.

- Is a special type of X.509 certificate that is signed by the normal end entity certificate or by another proxy.

- Supports single sign-on and delegation through "impersonation."

The proxy's private key must be kept secure, but because the proxy isn't valid for very long, it doesn't have to kept quite as secure as the owner's private key. Since a proxy is short lived, any potential compromise to the proxy is also short lived, which limits possible damage and reduces the administrative burdens of revoking and reissuing certificates. It is thus possible to store the proxy's private key in a local storage system without being encrypted, as long as the permissions on the file prevent anyone else from looking at it easily. Once a proxy is created and stored, the user can use the proxy certificate and private key for mutual authentication without entering a password.

When proxies are used, the mutual authentication process differs slightly. The remote party receives not only the proxy's certificate (signed by the user), but also the user's certificate. During mutual authentication, the user's public key (obtained from their certificate) is used to validate the signature on the proxy certificate. The CA's public key

is then used to validate the signature on the user's certificate. This establishes a chain of trust from the CA to the proxy through the user.

The creation of a proxy allows you to "log on" to the Grid, to verify authenticated access and run Globus Toolkit programs like MDS. You use the `grid-proxy-init` command to create a proxy. If you want to terminate a proxy before it expires (to "log out" of the Grid), you can use the `grid-proxy-destroy` command. These commands are described in more detail in the *MDS Security Setup Procedures* section of *Chapter 5, MDS Security Configuration*.

Grid "log on" with the `grid-proxy-init` command can be illustrated as follows:

User certificate file	
Pass Phrase → **Private Key (Encrypted)**	→ **User Proxy certificate file**
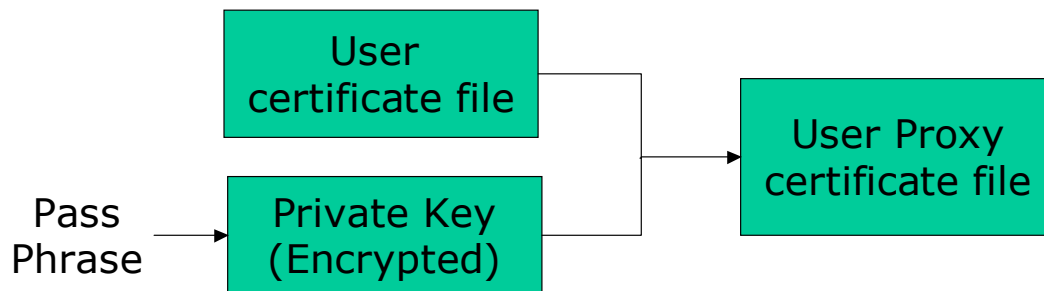
Programs for Credential Management

The Globus Toolkit provides the following command-line programs for credential acquisition and management:

grid-cert-request: obtains a new certificate.
grid-cert-info: displays the details of an existing certificate.
grid-cert-renew: renews an expired certificate.
grid-proxy-init: obtains a new proxy.
grid-proxy-info: displays the details of an existing proxy
grid-proxy-destroy: terminates an existing proxy before its automatic expiration time.

Full syntax for each of these programs can be displayed by using the `-help` or `-usage` options at the command line.