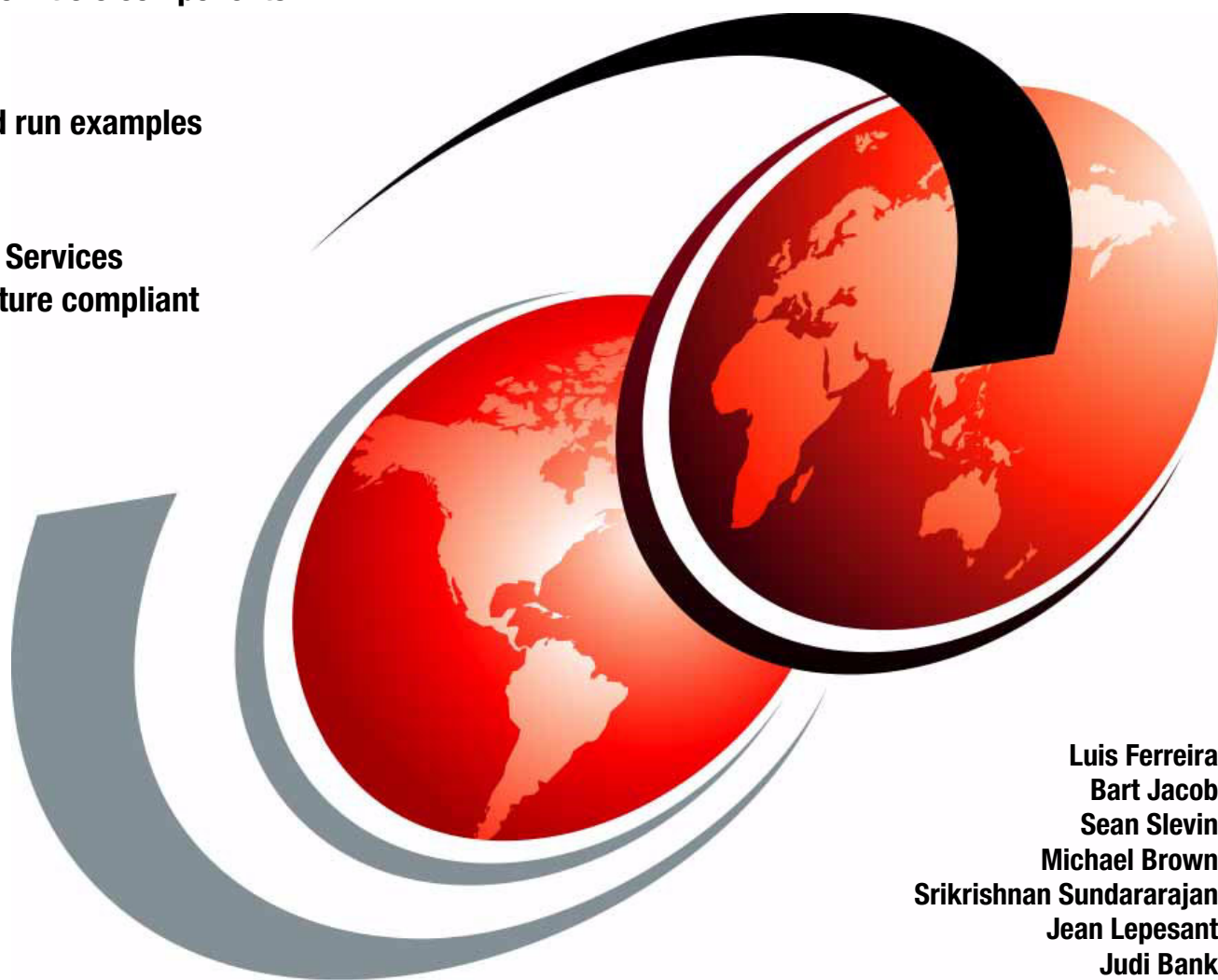IBM

Globus Toolkit 3.0 Quick Start

Globus Toolkit 3.0 components

Install and run examples

Open Grid Services Infrastructure compliant

Luis Ferreira
Bart Jacob
Sean Slevin
Michael Brown
Srikrishnan Sundararajan
Jean Lepesant
Judi Bank

Redpaper

IBM

International Technical Support Organization

Globus Toolkit 3.0 Quick Start

September 2003

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (September 2003)

This edition applies to release 3.0.1 of Globus Toolkit® 3.0.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	Notes®	Tivoli®
DB2®	OS/2®	WebSphere®
developerWorks®	Redbooks™	zSeries®
IBM®	Redbooks(logo) ™	

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redpaper describes the early experiences of the ITSO team with the Globus Toolkit 3.0. This major Grid implementation, hereafter referred to as GT3, is fully compliant with the new Open Grid Services Infrastructure (OGSI) Version 1.0 specification.

The goal of this Redpaper is to provide the critical jump-start for someone who wants to learn about GT3 but has little or no experience with prior Globus releases or grid computing in general. It show you how to implement a GT3 demo or a proof-of-concepts scenario. Also, it illustrates the high-level concepts of the toolkit components and the overall architecture.

The following topics are described in this documentation:

▶ Chapter 1, "Introduction" on page 1 presents an overview of grid computing, OGSI, OGSA, and Globus, explaining a bit of distributed computing history and where GT3 fits.

▶ Chapter 2, "Components" on page 9 presents the features and components provided by GT3 by using a progressive approach, starting at the Web service level up to a real Grid service.

▶ Chapter 3, "Installing on Linux" on page 23 presents a walk-through of the GT3 installation and configuration on machines running Red Hat Linux.

▶ Chapter 4, "Running examples" on page 49 shows you how to run some user-oriented examples. Also, it includes a sub-section oriented towards services operations, using the service browser GUI (Graphic User Interface).

Follow-up Redpapers and Redbooks™ to this Redpaper are planned that go into greater detail about GT3 toolkit programming, enterprise deployment, language support, security, integration, and so on. We suggest you occasionally check the IBM Redbooks Web site for new Grid Redbooks or subscribe to the weekly e-mail newsletter describing new Redbooks, residencies, and workshops at http://www.redbooks.ibm.com/.

Do not forget to check the IBM Grid Computing Web site at http://www.ibm.com/grid/, the Globus Project Web site at http://www.globus.org/, and the Global Grid Forum Web site at http://www.ggf.org/.

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Luis Ferreira, also known as "Luix", is a Senior Software Engineer at IBM Corporation - International Technical Support Organization, Austin Center, working on Linux and grid computing projects. He has 20 years of experience with UNIX®-like operating systems, and holds a MSc Degree in Systems Engineering from Universidade Federal do Rio de Janeiro in Brazil. Before joining the ITSO, Luis worked at Tivoli® Systems as a Certified Tivoli Consultant, at IBM Brasil as a Certified IT Specialist, and at Cobra Computadores as a Kernel Developer and Software Designer. He has written about GT3 concepts, components, installation, configuration, and how to run examples.

Bart Jacob is a Senior Consulting IT Specialist at IBM Corporation - International Technical Support Organization, Austin Center. He has 23 years of experience providing technical support across a variety of IBM products and technologies, including communications,

object-oriented software development, and systems management. He has over 10 years of experience at the ITSO, where he has been writing IBM Redbooks™ and creating and teaching workshops around the world on a variety of topics. He holds a Masters degree in Numerical Analysis from Syracuse University. He wrote the useful CAMgr tool.

Sean Slevin is a Software Engineer in the Grid Computing Initiative in Austin, Texas. He finished his MBA from Texas Tech University in 2002 and also holds a B.B.A. in Management Information Systems from the same school. His areas of expertise include grid computing, database management, and Web applications. He has written on Grid installation and configuration.

Srikrishnan Sundararajan is a Software Engineer with the IBM Linux Technology Center based at Bangalore, India. He holds a Bachelor of Engineering degree in Computer Science and Engineering from Madurai Kamaraj University. Srikrishnan has been exploring grid computing using the Globus Toolkit and has spoken on the architecture and APIs of the IBM Grid Toolbox at IBM DeveloperWorks Live 2003 conference held at New Orleans, LA. He has worked in the areas of Linux Internal, Systems Management product support, and Java™-based e-Business application development. He has Brain Bench certifications in Linux administration and programming, Java, C++, UML, and others. He has written about GT3 examples.

Michael Brown is a Senior Programmer and Sun-certified J2EE architect working in the IBM Linux Integration Center in Austin, Texas, where he is the leader of the teams working in the finance sector and Grid areas. He has more than 25 years of experience as an application developer and enterprise architect on multiple platforms and operating systems, including UNIX, Linux, AIX®, and OS/2®. Michael holds Bachelor of Science (Honours) and Master of Science degrees in Computer Science from the University of Western Ontario in London, Ontario, Canada. He has written about Grid concepts and OGSI.

Jean Lepesant is a IT Specialist working in the Grid Design Center for e-business on demand, IBM Server Group, located in France in Montpellier. He has worked for IBM for 25 years and been involved with grid computing since January 2003. His area of expertise includes application development, first with C language and later with Java.

Judi Bank is a Senior Technical Staff Member of the IBM Systems Group specializing in zSeries® and WebSphere® performance analysis. Judi has 23 years of experience with a large variety of IBM products and technologies, including JES2, zOS operating system, SYSPLEX, Java, WebSphere Commerce Suite, WebSphere Application Server, DB2®, HTTP Server, and, most recently, Globus Toolkit. Her primary responsibility is performance analysis and design. She has lectured extensively at user group sessions like SHARE and EXPO. Judi has a Master of Science degree in Computer Science from Fairleigh Dickinson University. She has helped during all test phases.

Acknowledgements

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with
specific products or solutions, while getting hands-on experience with leading-edge

technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

> **ibm.com**/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks in one of the following ways:

► Use the online **Contact us** review redbook form found at:

> **ibm.com**/redbooks

► Send your comments in an Internet note to:

> redbook@us.ibm.com

► Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

1

Introduction

This chapter presents an overview of grid computing and Globus. It also introduces the two main specifications relevant to GT3 implementation, which are:

- ► Open Grid Services Architecture (OGSA)
- ► Open Grid Services Infrastructure (OGSI)

1

1.1 Introduction to grid computing

You are a scientist, a businessperson, or an engineer. You are always looking for faster ways to do your computer processing, better ways to store and retrieve your data, more efficient ways to interact with your customers or suppliers or clients, all while spending less money. You may be asking yourself these questions, among others:

► How can I analyze the value of an investment portfolio in minutes, rather than hours?

► How can I significantly accelerate the drug discovery process?

► How can I cut the design time of products in half, while reducing the instances of defects?

► How can I efficiently expand and contract to meet cyclical demand?

► How can I unite research teams around the world to take advantage of the most up-to-date knowledge?

► How can I provide and use share data in a reliable and secure way?

► How can I leverage my activities consistently by collaborating with other organizations?

Ever since the first connection was made between two computers, developers have been using that connection to exploit resources. This work started out simply and grew into more complex applications, such as file sharing, print sharing, and e-mail, then grew to distributed computation across a group of networked machines. Generally, these tasks were born from the needs of a single user or developer, and evolved over time through a standardization process or by momentum into a *de-facto* standard to the point where different computers, from different vendors, running applications written in different languages, and running on different operating systems, could all interoperate successfully.

As anyone who has ever had to deploy a new application knows that as an enterprise infrastructure grows, the problem of "islands of servers" comes up. Servers are put in place to handle a specific task. The organization has to purchase computing power and data storage equipment, for the expected peak load, while the average daily load on the system is much less. This obviously wastes money on the initial purchase, electricity, floor space, cooling, and administration. As time goes by, computers are used for more and more sophisticated tasks, and some of these tasks are still ad-hoc or bound to a single-vendor or a single-operating system. Others tasks build new layers on top of existing standards-based platforms, and so on.

Grid computing is the clear answer for the above scenarios. The solution is to coordinate distributed resources, using open standards and general-purpose protocols and interfaces. A set of systems and applications able to provide multiple qualities of services as a single Grid.

Global Grid Forum and Open Grid Services

The Global Grid Forum (GGF) is an open community initiative that aims, as stated in its Web page, "to promote and support the development, deployment, and implementation of Grid technologies and applications via the creation and documentation of 'best practices' - technical specifications, user experiences, and implementation guidelines".

Two main specifications relevant to the discussion in this document are the result of the GGF initiative. They are:

► Open Grid Services Architecture (OGSA)

► Open Grid Services Infrastructure (OGSI)

OGSI's and OGSA's goals are to provide the framework to support the creation of set of searchable services, callable by any system on a Grid. The key is that the definition of the

service interface itself is completely separate from its underlying implementation, which is what gives us the ability to provide OGSA services on virtually any computing device that can connect on a network. The old computer science saying "You can solve any problem by adding one level of indirection" applies here, but the key to the success of OGSA over previous systems is its standards-based design layered on top of existing key standards-based layers.

OGSA services provide a virtualization layer on top of the real computing resource, making it possible to create dynamic management and configuration of virtual organizations.

For detailed information about Grid, OGSA, and OGSI, refer to the document *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration* by Foster, et al, found at:

http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf

1.1.1 General classes of grid computing

The type of jobs done in a grid computing environment can vary widely, as needed by the application. Listed below are some general trends and benefits of the uses of grid computing, along with a brief description of where each fits in the overall Grid world. It is important to notice that all Grid jobs have, at least, a small element of compute and data.

Computation

This is the type of Grid job most people first think about, by dividing a computational task into multiple parallel jobs running simultaneously on computers in the Grid. Distributing computational load around the Grid can have dramatic reductions in overall processing elapsed time and increase overall system utilization. Generally a master system carves up the data set into small pieces and delivers them to systems in the Grid waiting for work. Each system performs its data processing, then returns the result of that processing back to the master system, which collects and consolidates the results into the overall solution.

Data virtualization

Here data storage is the resource. By aggregating data, an intelligent, cross-network file system can harness the islands of unused disk space into a virtual data scenario. All it needs to do is keep track of where a piece of data is written so it can be retrieved successfully.

More sophisticated scenarios could store the data around the Grid redundantly in order to provide a highly available and reliable data system.

Databases play a big role on data grids, specially federated databases, where there are multiple sources of data, and a need to combine the information from these various sources. This can provide a single point of contact and name space for a larger subset of file or data resources. From there, there is a combination of caching and replication technologies to try to ease the burden of remote geographies.

Data virtualization can be applied on data mining or business intelligence systems. In this case, not only a file system or database is virtualized, but an organization's entire collection of data is brought into a coherent whole. This might also include a combination of data and computational grid.

Resiliency

This is a catch-all for the use of network-wide redundancy of all classes of resources in a system in a high availability scenario. By using a certain level of redundancy, business can

continue in the face of many types of hardware, software and network failures, without the burden of paying for unnecessary idle resources.

1.1.2 Industry specific areas of use

Given the generalized sample scenarios above, we highlight some potential uses for Grid technology in different industries, but it is not restricted to only those areas. Grid computing capability is growing everyday, and we believe that someday, all systems and applications will run in some kind of Grid-like environment. The following list is far from complete, but represents potential for the present and future:

Finance
- ▶ Derivative analysis
- ▶ Statistics
- ▶ Portfolio risk
- ▶ Insurance policy cost
- ▶ Real-time stock market analysis

Life sciences
- ▶ Drug screening
- ▶ Protein folding
- ▶ Protein sequencing

Medicine
- ▶ Record management
- ▶ Automated analysis and diagnosis
- ▶ Research into the nature of disease

Government/Academia
- ▶ Dispersed research center collaboration
- ▶ Weather forecasting
- ▶ General high-performance computing

Energy
- ▶ Seismic analysis
- ▶ Oil field simulation

Manufacturing
- ▶ Product design
- ▶ Simulation
- ▶ Modelling
- ▶ Finite element analysis (anything involving flow (air, water, fuel, and so on)

Telecommunications/Media
- ▶ Video rendering
- ▶ Network gaming
- ▶ Content distribution
- ▶ Dynamic bandwidth for new classes of applications

Electronics
- ▶ Chip layout optimization
- ▶ Board layout optimization

► Circuit simulation

1.2 Introduction to Globus, OGSI, and OGSA

Now that we have put distributed computing and its uses into perspective, it is time to introduce all of the relevant standards that Globus is based on.

1.2.1 Previous generations of the Globus Toolkit

Globus Toolkit Version 1 (GT1) and Version 2 (GT2) are the predecessors of GT3. Services and protocols offered by these versions have evolved to support Grid services with more pervasive services and servers. Grid services are Web services that conform to a specific set of conventions defined by OGSI. GT2 is still supported as current users make the transition to GT3.

The technologies used to build GT2 components include Grid Resource Allocation Management (GRAM), Monitoring and Discovery Service (MDS), and Grid File Transfer Protocol (GridFTP). All of these components utilize the Grid Security Infrastructure (GSI) protocol for security at the connection layer. For more information about GT2 and GT3 components mapping, refer to 2.4, "GT2 and GT3 pillars" on page 20.

1.2.2 GT3

GT3 is an implementation of the Open Grid Services Infrastructure (OGSI) Version 1.0, a new specification that the Globus Project played a key role in defining, hosted at http://www.globus.org/. The focus of this Redpaper is on this version.

Globus is using OGSI as their infrastructure for their GT3 base services. They also add some management services. OGSI represents the reference open source standard implementation of the Open Grid Services Infrastructure standard. Figure 1-1 shows the key areas identified as the basis for grid computing. At least one service of each pillar should be included in most of Grid implementations. These key areas are resource management, information services, and data management.
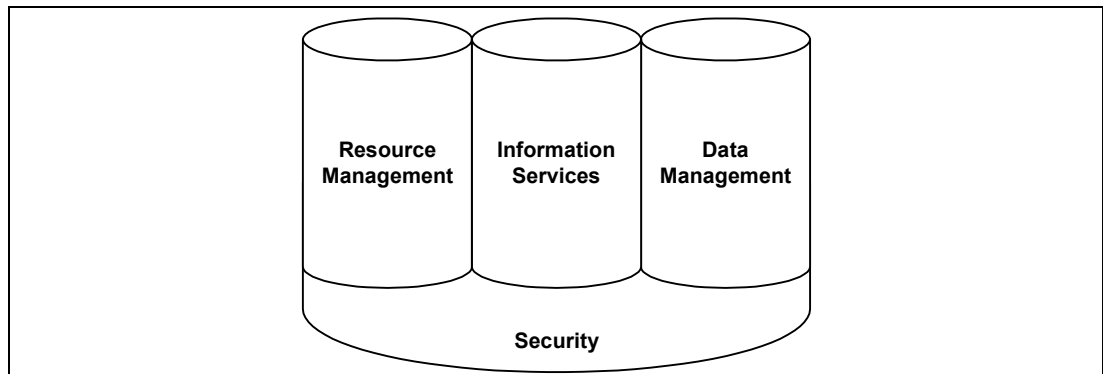
Figure 1-1   Grid computing key areas

1.2.3 Open Grid Services Architecture (OGSA)

OGSA defines the blue print of how a Grid should look appear, including the infrastructure. It also defines the programming model of Grid services. It provides instructions on how to build

a Grid service, by outlining the required components needed to build and deliver an enterprise-class Grid solution.

The OGSA itself is described by the GGF as follows

> "Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration requires the early definition of a core set of interfaces, behaviors, resource models, bindings, and so forth: what we call the OGSA Platform. The OGSA working group within the Global Grid Forum has been formed to define this OGSA Platform by (a) specifying, in broad but somewhat detailed terms, the scope of important services required to support both e-science and e-business applications, (b) identifying a core set of such services that are viewed as essential for many Grid systems and applications, and (c) specifying at a high-level the functionalities required for these core services and the interrelationships among those core services."

Here is a list of some items added by OGSA:

- ► Identity establishment and authentication negotiation
- ► Policy expression and negotiation
- ► Service discovery, monitoring, and management
- ► Service level agreement negotiation and monitoring
- ► Virtual organization membership management and communication
- ► Hierarchical service collection
- ► Data resource integration into computations
- ► Distributed resource management across heterogeneous platforms
- ► Seamless Quality of Service delivery
- ► Common base for autonomic management solutions
- ► Common infrastructure building blocks to avoid "stovepipe solution towers"
- ► Open and published Interfaces
- ► Leverage industry-standard integration technologies: SOAP, XML, and so on
- ► Seamless integration with existing IT resources

OGSA architecture layers can be implemented by different products, developed by multi-ISVs, software developers, organizations, open communities, and computer manufacturers, around the world.

OGSA Grid Service Programming Model

When developing a new service for deployment within an OGSA-compliant system, you must follow the Grid service programming model. The programming model provides capabilities essential to distributed/grid computing and all OGSA services adhere to specified service interfaces and behaviors. Some of those capabilities are required and others are optional.

- ► Factory
- ► Registry
- ► Discovery
- ► Life cycle
- ► Query service data
- ► Notification
- ► Reliable invocation

A Grid service can be visualized as the instantiation of the interfaces described above. The important thing to remember is the only contact between a Grid service and its users (applications running on the Grid) is the service interface. These service interfaces are defined by the existing Web Services Description Language (WSDL). Several enhancements to WSDL have been identified for OGSI requirements and are currently being added to the WSDL standard. For more information on WSDL, consult http://www.w3.org/.

1.2.4 Open Grid Services Infrastructure (OGSI)

OGSI is the concrete specification of OGSA infrastructure. It is the middleware, the "Java 2 Platform" for Grid services. It defines how to build a Grid service, outlining the mechanisms for creating, managing, and exchanging information for Grid services.

The OGSI itself is described by the GGF as follows:

"Building on both Grid and Web services technologies, the Open Grid Services Infrastructure (OGSI) defines mechanisms for creating, managing, and exchanging information among entities called Grid services. Succinctly, a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviors) that define how a client interacts with a Grid service. These conventions, and other OGSI mechanisms associated with Grid service creation and discovery, provide for the controlled, fault resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications."

1.3 Summary

After reading this chapter, you may be overwhelmed with acronyms, standards, and layers upon layers of software, keep in mind that it is just distributed computing; distributed computing built on modern open standards, designed to allow heterogeneous access to Grid resources, but distributed computing nonetheless.

The IT industry has been doing custom, single-purpose projects for decades. OGSA allows us to concentrate on solving high-level problems by providing a strong infrastructure, toolkit, and third-party involvement.

For further information about grid computing, solutions, and other initiatives, check the IBM Grid Computing Web site at http://www.ibm.com/grid/.

2

Components

In this chapter, we introduce the features provided by the GT3. This is done using a progressive approach, starting at the Web service level up to a real Grid service.

The following points will be addressed:

► Web services
► Grid services
► GT3 organization

2.1 Web services

Web services are the foundation for Grid services, which are the basis of OGSI, OGSA, and therefore, GT3. It is necessary to understand the Web services architecture to understand Grid services and GT3.

2.1.1 Overview

Web services are a distributed computing technology that allows the creation of applications based on the client/server model. What makes Web services special?

► Web services are platform and language independent.

► Web services use open and known protocols, such as HTTP.

Figure 2-1 shows a macro flow of how Web services work. Imagine a Web application querying a database located in a central site. Web services use standard Simple Object Access Protocol (SOAP) protocol and Extensible Markup Language (XML) grammar to communicate; the application can be written in any language, such as C, C++, or Java, and running on any platform, such as Linux, AIX, or Windows®. Hypertext Transfer Protocol (HTTP) is the primary protocol for Web services. This makes the service request and response transmission over a very well-known transport protocol. For more information on XML, SOAP, and HTTP, consult the World Wide Web Consortium (W3C) at http://www.w3.org.
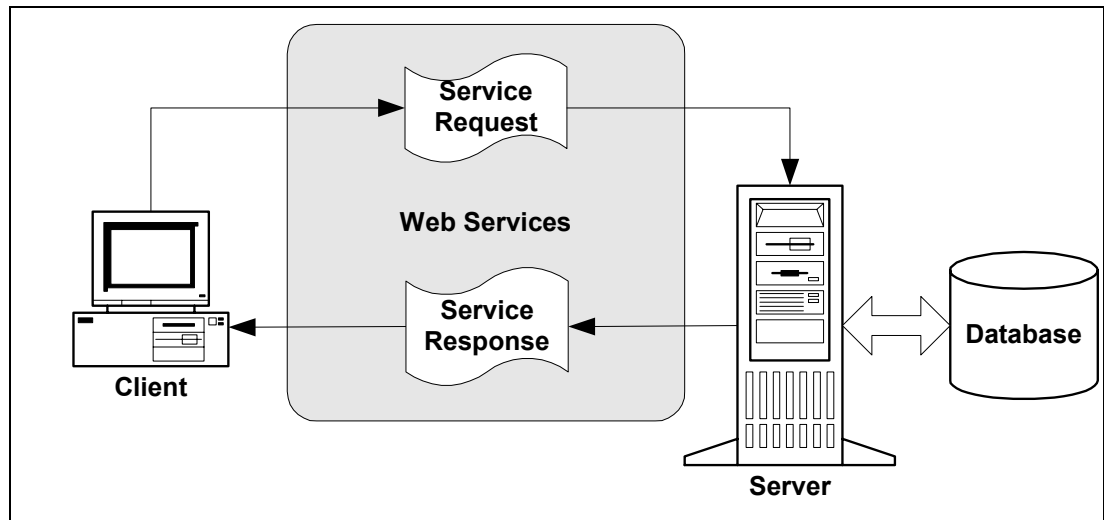
Figure 2-1   Distributed service call

While technologies such as Common Object Request Broker Architecture CORBA) and Enterprise JavaBeans (EJB) are oriented to very dependent clients and servers, Web services are oriented to clients that do not have previous knowledge of the Web service until they invoke it.

Web services do not address transient and stateful services. In most cases, Web services do not remember values from one call to another. However, in distributed systems, there are some scenarios when transient and stateful services are required. These are addressed by Grid services.

2.1.2 Web service invocation

Let us take a look at the macro steps involved in a complete Web service invocation. Note that GT3 takes a slightly different approach than a Web service. We will review a Web service scenario and than present the differences with Grid services.

Figure 2-2 illustrates a Web service invocation:

1. The client tries to discover where the desired Web service is up and running by calling the Universal Description, Discovery, and Integration (UDDI) Registry.

2. The UDDI Registry replies by providing the address in the form of an Uniform Resource Identifier (URI). It points to one of the servers that hosts the desired Web service. The URI is just like a Uniform Resource Locator (URL) in Web pages. Web services are addressed with URIs because they are generally hosted within a Web container. Here is an example of a URI:

   `http://webservices.example-site.com/application/desired-service`

3. The client now knows the address of the Web service (URI), but it does not know how to invoke it. So, the client asks the Web server how to invoke the Web service.

4. The Web server replies with a Web Services Description Language (WSDL) document that can be used to figure out how to invoke the desired Web service. The WSDL fully describes the interface of the Web service.

5. The client now knows how to invoke the desired Web service. The invocation may be done by many protocols. Simple Object Access Protocol (SOAP), a protocol for exchanging information in a decentralized and distributed environment, is commonly used. The encapsulated message is written in XML. For more information on SOAP, consult the W3C at http://www.w3.org/.

6. The Web service replies with a SOAP answer, encapsulating the response message in XML grammar.
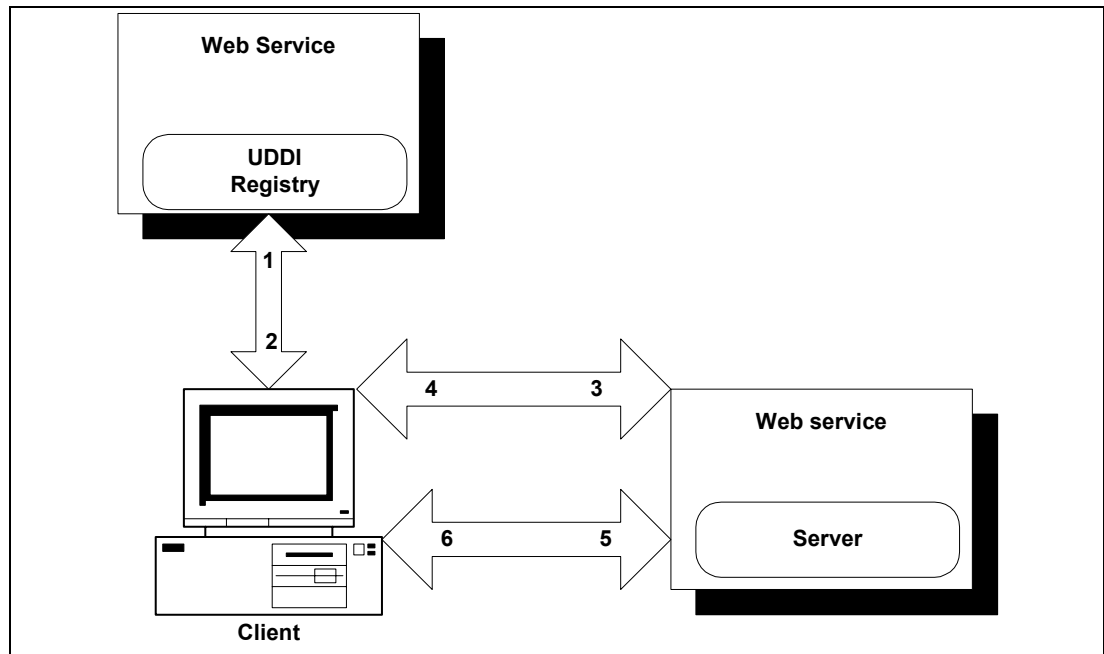
Figure 2-2   Web service invocation

2.1.3 Structure of a Web service application

Developers of Web services normally do not need to worry about the underlying elements of the Web implementation, such as SOAP and WSDL. Tools are available to help them automatically generate program stubs. These stubs are responsible for interpreting requests and forwarding them to the proper destination.

Figure 2-3 shows the components that may be involved in a Web service request:

1. Client invokes a Web service method by calling the client stub. The stub will turn this request into a SOAP request.
2. The SOAP request is sent to the server using the HTTP protocol.
3. The server stub receives the SOAP request, converts it, and delivers it to the Web service method invoked by the client.
4. The server replies with the result of the operation to the stub, which will turn it into a SOAP response.
5. The SOAP response is sent over the network using the HTTP protocol.
6. The client stub receives the SOAP response, converts it, and delivers it to the client.
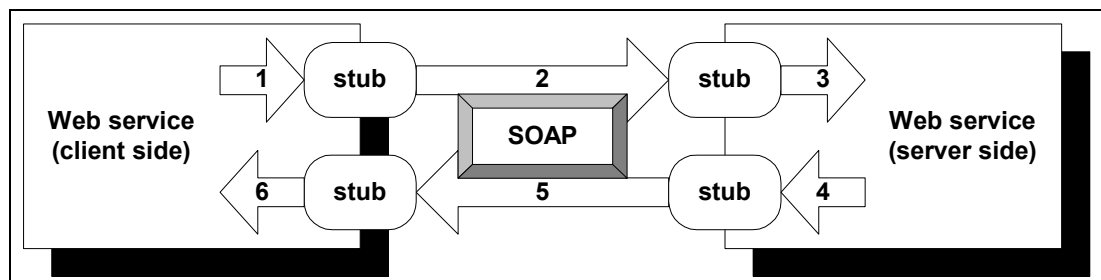
Figure 2-3   Web service application structure

2.2 Grid services

As stated before, Grid services are built upon Web services, and OGSI defines the additional mechanisms for creating and managing Grid services. Some concepts, such as naming, service data, notification, and life cycle, are part of the key features defined by OGSI. Table 2-1 summarizes those features.

Table 2-1   Features at a glance

Concept	Description
Naming	► Unique name for services instances ► Discover operation done by naming
Service data	► Service data is associated with any Grid service instance ► Operations to set and query the service data values ► Notifications on service data changes
Notification	Interfaces for registering and delivering notifications and also subscriptions

Concept	Description
Life cycle	▶ Grid services instances can be created by Factories ▶ Grid services instances can be explicitly or "soft" destroyed

For more information about the full list of Grid services features (which includes stateful and inheritance) and OGSI specification, refer to `http://www.ggf.org/ogsi-wg`.

2.2.1 Naming

In 2.1.3, "Structure of a Web service application" on page 12, we saw that Web services are addressed with URIs. Since Grid service are Web services, they are also addressed with URIs. However, as defined in OGSI, a "Grid service URI" is called a Grid Service Handle (GSH). In order to meet the requirements to communicate with the service, GSH must be resolved to a Grid Service Reference (GSR).

▶ GSH: Points to a Grid Service.
▶ GSR: Specifies how to communicate with the Grid service.

GSH and GSR

Each GSH must be unique and point to a Grid service instance. There cannot be two Grid service instances with the same GSH. A GSH can be thought of as a permanent network pointer to a particular Grid service instance.

The GSH does not provide sufficient information to allow a client to access the service, so the client needs to "resolve" a GSH into a GSR. The GSR contains all information that a client requires to communicate with the service.

The GSR is not a permanent pointer to a Grid service instance because a GSR may become invalid for various reasons, for example, the Grid service instance may be moved to a different server.

GT3 provides a mechanism called *HandleResolver* to support client resolution of a GSH into a GSR. GSR can take many different forms, but since GT3 uses a SOAP connection to communicate with a Grid Service, the GSR is a WSDL file (remember that WSDL describes the Web service interface: what methods it has, and so on).

▶ GSH: HTTP/S URL based GSH scheme
▶ GSR: SOAP over HTTP/S, so GSRs are in WSDL format

2.2.2 Service data

Service data is probably one of the most important concepts of Grid services. Service data is a structured collection of information that is associated with an instance of a Grid service. It is a mechanism to expose a service instance's state data to the service requestors. Therefore, these requestors (clients or servers) are able to query, update, and change those Service Data Elements (SDE).

Every Grid service instance by default has a few standard SDEs, and each one can be of a different type. There is a list of standard SDEs as part of the OGSI specification. For a detailed description of those standard SDEs, refer to the OGSI specification at `http://www.ggf.org/ogsi-wg`. In addition to the standard service data, APIs are available to allow dynamic creation of services data that corresponds to the data information in the WSDL of your service instance.

Grid service instances hold the value of the service data, which can be queried at anytime or be associated to a callback notification when its value changes. For that, OGSI provides an interface to query for SDEs or subscribe to notifications (see 2.2.3, "Notifications" on page 14).

2.2.3 Notifications

Notifications create a mechanism to allow a notification source to deliver a message to a notification receiver (also known as a notification sink). In GT3, the notification cycle is managed by a subscription management service.

As illustrated in Figure 2-4 on page 15, the notification framework defines the following components:

1. Subscription request

   A message sent to the notification source containing the location of the notification sink to which notification messages are to be sent, and an initial lifetime for the subscription source. A subscription request causes the creation of a Grid service instance called a subscription.

2. Subscription expression

   An XML document that describes the rules and format of the notification message, such as notification destination, and when it should be sent.

3. Grid service subscription instance

   A service instance created during the subscription operation. This instance manages the subscription properties.

4. Notification source

   A grid service instance that sends notifications.

5. Notification message

   An actual callback notification message. Notifications can also be sent to post service data value modifications. So, in this case, notifications use a service data concept behind the scenes. When a service instance wants to receive a notification associated to a particular SDE, the service needs to be subscribed in order to be notified of subsequent changes to the target instance's service data.

6. Notification sink

   A grid service instance that receives notifications.
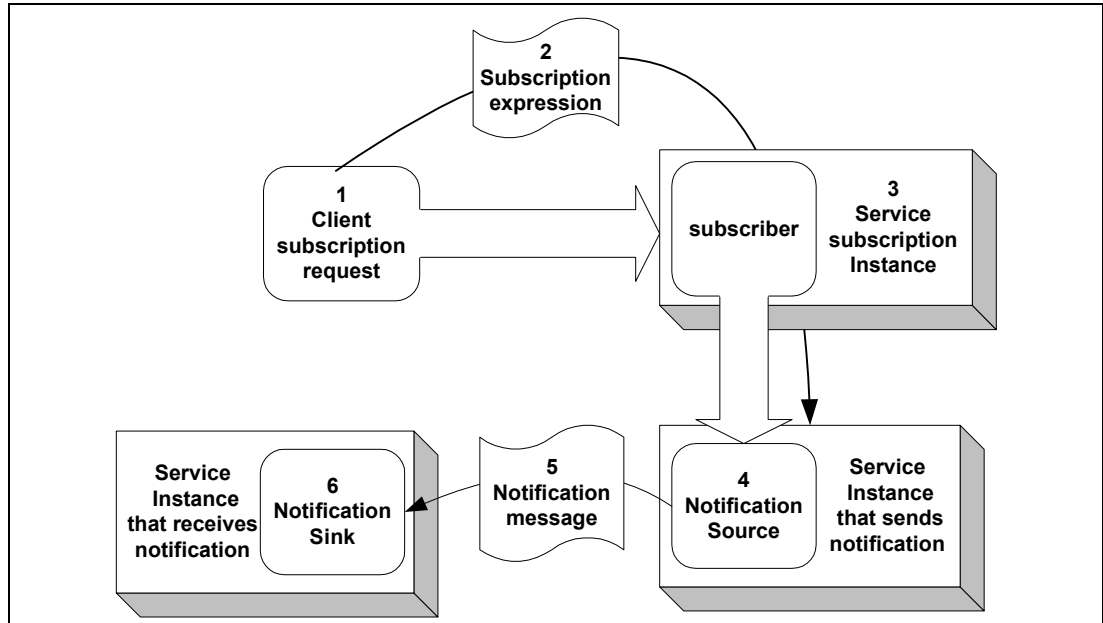
Figure 2-4   Notification framework

2.2.4 Life cycle

The OGSI specification defines the life cycle of any Grid service instance to be "demarcated by the creation and destruction of that service instance". It is one of the core properties of a Grid service. Actually, the mechanism is part of the hosting environment. A Grid service also support notifications of lifetime related events.

Creating a Grid service instance may be done by requesting a Factory to create an instance for the client, while destroying it may be done by invoking a method on the service instance itself. Also, a service instance may be destroyed by what is termed as "soft-state" approach, where the client registers interest in a service for a period of time and when the period expires; if no reaffirmation of interest is made, the service is terminated.

Factory

A Factory is a mechanism to create instances of Grid services. It is part of the OGSI specification and it is implemented via a Factory interface through a class of Grid services.

An operation to create a Grid service returns the GSH associated to requested service, but this information is not enough to invoke the service. GSR is also necessary. The client needs to obtain the GSR, "resolving" the GSH. While GSH is a permanent reference, GSR is limited for period of time.

Figure 2-5 on page 16 illustrated how Factory can be used to create an instance of a Grid service:

1. Client discovers a Factory by querying the registry service to get the handle to the Factory service.

2. Client calls an operation, on the Factory, to create an instance of a Grid service.

3. Factory creates a new instance of the Grid service.

4. Factory returns the GSH of the new Grid instance to the client.

5. Client and server interact as result of the initial call.
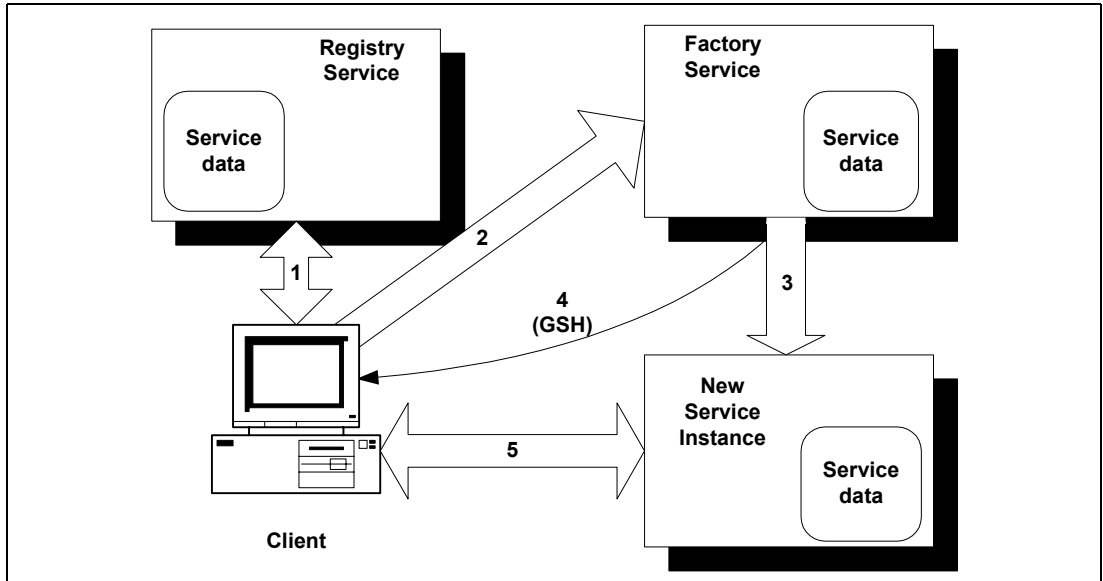
Figure 2-5   The Factory concept

2.3 GT3 organization

This section describes the components which GT3 is built upon. Figure 2-6 illustrates the building blocks representing those components that are included in the GT3 implementation. The blocks in gray represent GT3 core.
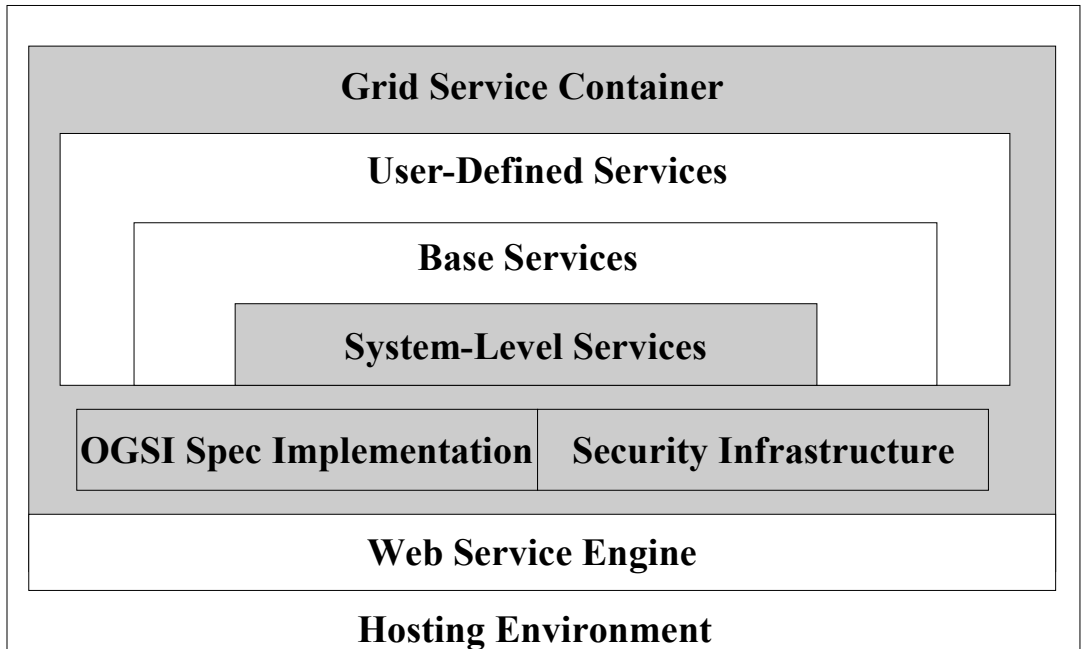
Figure 2-6   GT3 implementation

2.3.1 Hosting environment, Web Service Engine, and Grid Service Container

A hosting environment is where the GT3 runs. It is the server environment, such as an operating system or an application server. Currently, GT3 supports the following types of hosting environments associated with GT3 Service Containers that can be selected during the installation/configuration time:

► Embedded Container
► Stand-alone Container
► J2EE Web Container (Servlet)
► J2EE Enterprise JavaBeans Container (EJB)

As presented Figure 2-6 on page 16, GT3 Grid Service Container is implemented upon the OGSI specification and Security Infrastructure specification. The gray area in Figure 2-6 on page 16 represents the GT3 core. For more information about OGSI specification and Security infrastructure specification, refer to http://www.ggf.org.

Figure 2-7 illustrates the implementation layers of the hosting environment. It is built upon the container that follows the OGSI specification and a Web engine, such as Apache AXIS. The AXIS engine is responsible for managing SOAP over HTTP communication. It acts as a SOAP message listener and performs the SOAP request/response serialization and de-serialization, service configuration, and deployment. For more information on AXIS, refer to http://ws.apache.org/axis.
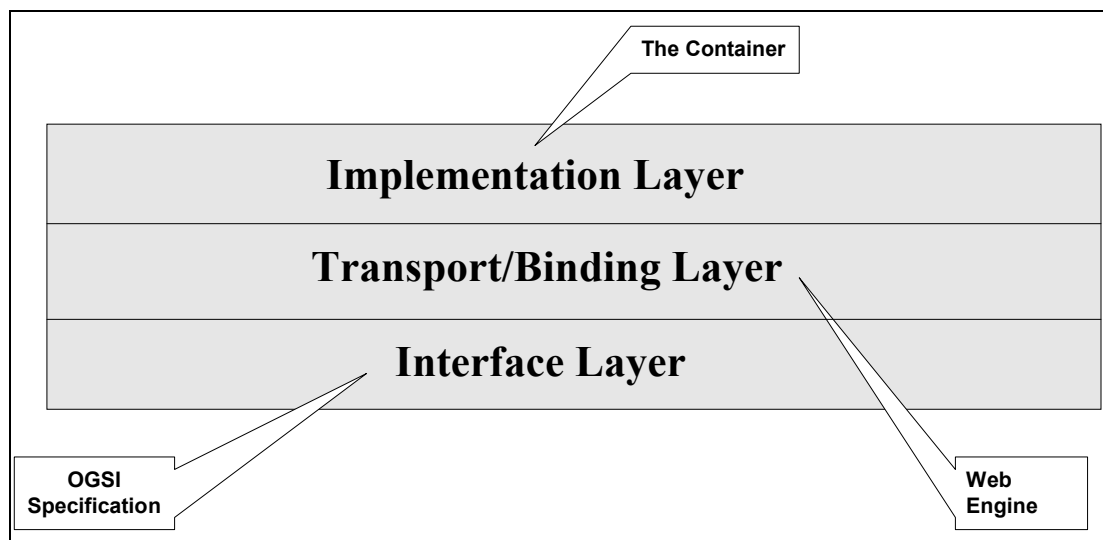
Figure 2-7   Hosting environment layers

Embedded hosting environment

An embedded hosting environment is a hosting environment used mainly on clients or on lightweight servers to allow creation and management of Grid services. For example, a subscription operation creates a lightweight hosting environment to run the subscription service instance on the client side.

See Figure 2-8 on page 18 for an example of an embedded hosting environment.
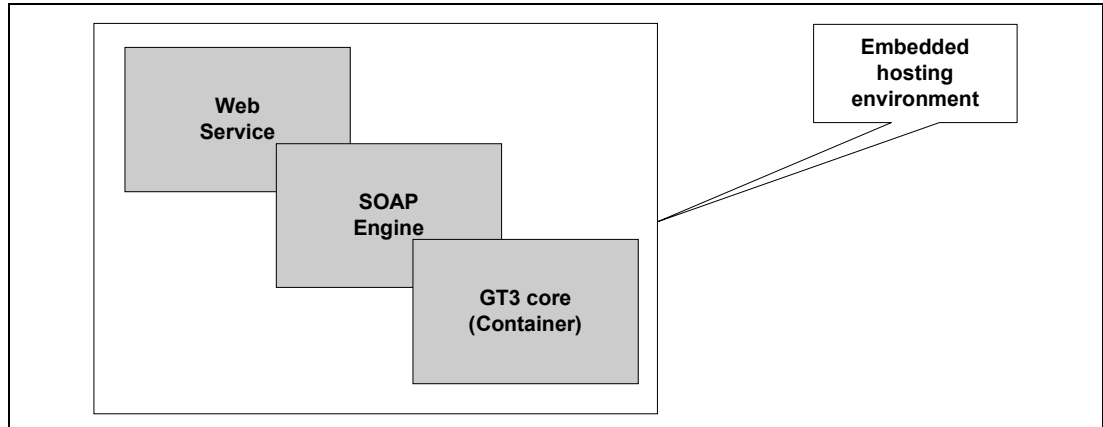
Figure 2-8   Embedded hosting environment

Stand-alone hosting environment

A stand-alone hosting environment is basically the embedded hosting environment and a front-end command line, which starts and stops the hosting environment. For more information on how to start and stop a stand-alone hosting environment, refer to the `globus-start-container` and `globus-stop-container` commands in 4.1, "Start hosting environment container" on page 50, and in the *User's Guide Core Framework* at http://www-unix.globus.org/toolkit/3.0/ogsa/docs/users_guide.html.

Figure 2-9 gives an example of a stand-alone hosting environment.
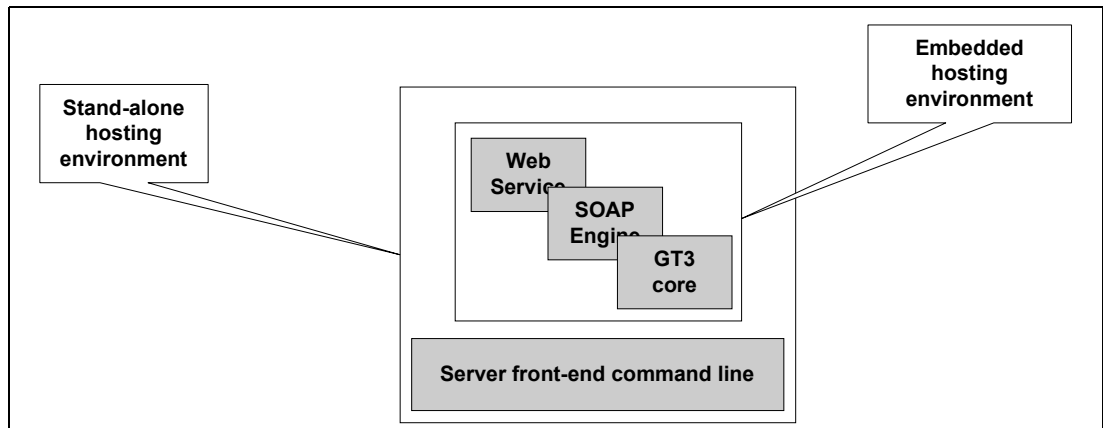
Figure 2-9   Stand-alone hosting environment

Servlet hosting environment

A servlet hosting environment is essentially the embedded hosting environment running inside a Java engine (Web container), such as Tomcat or WebSphere Application Server. This Web container uses the Web services provided as part of the Java engine, instead the standard one provided by GT3.

Figure 2-10 on page 19 gives an example of a servlet hosting environment.
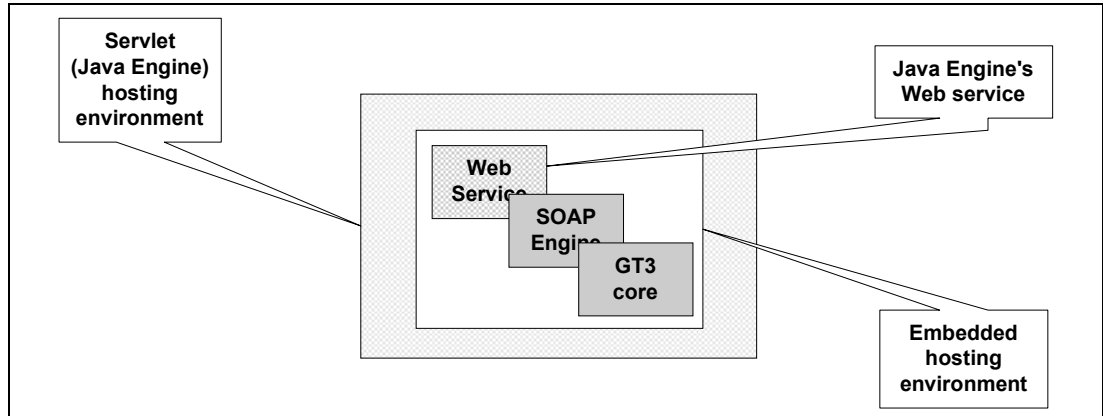
Figure 2-10   Servlet hosting environment

For more information about Tomcat and WebSphere, refer to the Apache Jakarta Project at
http://jakarta.apache.org/tomcat/ and the WebSphere software platform Web page at
http://www.ibm.com/websphere/, respectively.

EJB hosting environment

The EJB hosting environment is essentially the embedded hosting environment running
inside a EJB application server (EJB container), such as WebSphere Application Server. For
more information about EJB application servers, refer to the following Web sites:

http://www.ibm.com/developerworks/java/library/co-tipejbs.html?dwzone=java
http://www.ibm.com/websphere/

2.3.2 System level services

System level services are general purpose services that facilitate the use of Grid services in
production environments. The GT3 core distribution included the following system level
services, which are geared towards administrators who set up the GT3 environment:

► Administration Service

    It allows for "pinging" and shutdown of the hosting environment.

► Logging service

    It allows you to manage log filters at run time and monitor log buffers.

► Management service

    It allows for the discovery of the current load and for destroying, activating, and
    deactivating services.

2.3.3 GT3 base services

This section describe the base services provided by GT3. The base services are not part of
the GT3 core. These services need to be installed:

► Job management
► Index services
► Reliable File Transfer (RFT) services (multiRFT)

Job management services

Job management services provide a way of submitting and monitoring jobs remotely, that is,
submitting jobs in the Grid. It follows the interfaces defined in OGSI by using WSDL, which is

based on XML. Job management services provide a client-side command called
`managed-job-globusrun` that invokes the Master Managed Job Factory Service (MMJFS) to
submit a job. For more information about job management, refer to the Globus Resource
Management Web page at:

http://www-unix.globus.org/developer/resource-management.html

Index services

Index services are mainly used in discovery operations. Basically, they provide a way to
query and produce service data. Index services provide a client-side command called
`ogsi-find-service-data`. The command allows you to query any service data element from
any Grid service. For more information about Index Services, refer to the Globus Information
Services Web page at http://www.globus.org/ogsa/releases/final/docs/infosvcs/.

RFT services

RFT, also known as multiRFT, is part of the Data Management implementation, along with
Grid File Transfer Protocol (GridFTP) and Replica Relocation Service (RLS). It provides the
interface for reliable file transfers on Grid servers. A RFT client side Java based program is
provided (RFTClient). For the server side, Data Management utilizes the same GridFTP
daemon provided in GT2. For more information about Data Management, refer to the Globus
Data Management Services Web page at:

http://www-unix.globus.org/developer/data-management.html

2.4 GT2 and GT3 pillars

The organization of GT2 consists of three pillars on top of a security infrastructure known as
Grid Security Infrastructure (GSI), as illustrated in Figure 2-11. These pillars are:

► Grid Resource Allocation Management (GRAM)
► Monitoring and Discovery Service (MDS)
► Grid File Transfer Protocol (GridFTP)
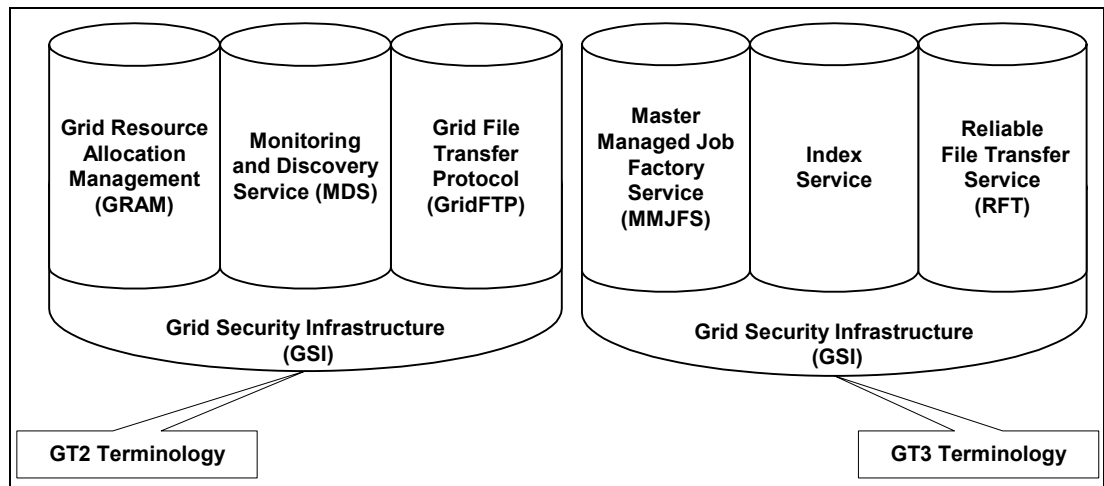
Figure 2-11   Terminology

GRAM

This resource management pillar provides support for:

- ▶ Resource allocation
- ▶ Submitting jobs (remotely running executable files and receiving results)
- ▶ Managing job status and progress

MDS

This information services pillar provides support for collecting information in the Grid and for querying this information, based on the Lightweight Directory Access Protocol (LDAP).

GridFTP

This data management pillar provides support to transfer files among machines in the Grid and for the management of these transfers.

As illustrated on Figure 2-11 on page 20, some terms in GT3 have changed, but the fundamental pillars of the architecture remain the same.

Table 2-2 shows the relationship between GT2 components and GT3 services.

Table 2-2   GT2 versus GT3

GT2 components	GT3 services
GRAM (gatekeeper)	MMJFS
MDS GIIS	Index services
MDS GRIS	SDE in MMJFS
GridFTP server	GridFTP server
GRAM Reporter	SDE in Managed Job Service
globus-url-copy	Reliable File Transfer Service

2.5 How to write a service

The goal of this Redpaper is to provide the critical jump-start for someone who wants to learn about GT3 but has little or no experience with prior Globus releases or grid computing in general. It is *not* intended to describe how to write a Grid service or application. For that task, we recommend the following online documents:

- ▶ *Java Programmer's Guide Core Framework*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/java_programmers_guide.html

- ▶ *Grid Service Development Tools Guide*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/tools_guide.html

- ▶ *The Globus Toolkit 3 Programmer's Tutorial*, found at:

  http://www.casa-sotomayor.net/gt3-tutorial/

3

Installing on Linux

This chapter presents the steps necessary to install and configure GT3 on machines running Red Hat Linux.

The following topics will be discussed:

▶ Lab environment
▶ Acquiring the necessary software
▶ Installing the software
▶ Configuring the software
▶ Testing the installation

At the time of writing, GT3 Version 3.0.1 of the Toolkit was just released, and this document is based on that version.

3.1 Lab environment

This section gives an overview of the configuration of the software and hardware used in our lab. We built a very small scenario. It is the simplest Grid environment, intended to help illustrate the concepts and components behind the Grid and GT3. An Ethernet LAN and three Intel® Pentium® machines were used. In Figure 3-1, we illustrate this environment with the host names and the functionality of each machine. The host names are x1 and x2. Also, an infrastructure server, called m0, was set. The machines should have a clock speed of 1 GHz, 512 MB of memory, and hard drives totaling 20 GB.

If you have more than three machines available, you can build a bigger scenario for Proof-of-Concepts (PoC) proposals and/or demos. For that, you simply include more servers, such as x3, x4, and so on.
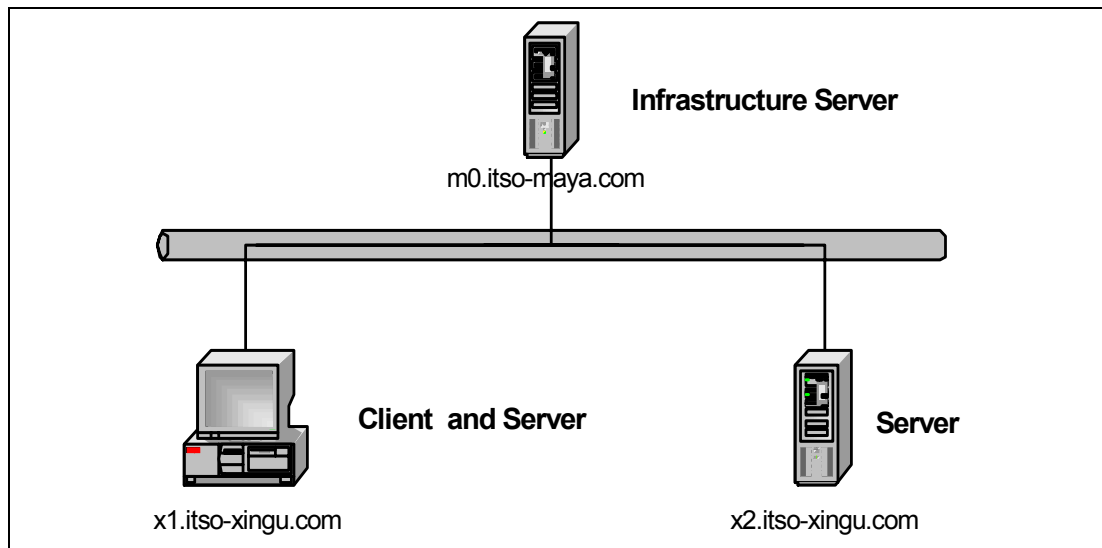
Figure 3-1   Hardware environment and software functions of each machine

3.1.1 Naming and addressing planning

Table 3-1 summarizes the names of the machines to be used in the Grid, their IP addresses, and the software to be installed on them.

Table 3-1   Host names and IP addressing

Host name	IP	Description
x1.itso-xingu.com	192.168.0.241	Globus server and Client machine
x2.itso-xingu.com	192.168.0.242	Globus server
m0.itso-maya.com	192.168.0.10	Infrastructure server

3.1.2 Certificate authority

The Grid environment needs an existing certificate authority. A simple certificate authority is installed as part of the m0 setup. For more information about m0 setup, refer to Appendix A, "Infrastructure server setup" on page 73.

3.1.3 Users and groups

We recommend that you define the users and groups that you want to use before implementation. Table 3-2 contains the list of user and group IDs used in our lab.

Table 3-2   IDs and passwords

user ID	group ID	password	Activities
root	root	\<password\>	Super user needs.
globus	globus	\<password\>	Globus Toolkit environment. For installation and execution of the Toolkit.
itso	itso	\<password\>	End user environment. For jobs execution on the Grid.

Note: It is important that the user ID under which you plan to run the GRAM master managed job Factory is a member of the group ID that owns the launch_uhe_setuid program. For more information, refer to the MMJFS configuration at:

http://www-unix.globus.org/toolkit/3.0/ogsa/docs/admin/configuration.html

3.1.4 Directories

Table 3-3 presents the Globus directories' definitions used in the lab environment on machines x1 and x2.

Table 3-3   Globus directories

Directory	Ownership	Description
/usr/local/globus	globus:globus	Base directory of Globus
/home/globus/install	globus:globus	Installation directory

3.2 Software installed

GT3 needs several files, or tools, in order to complete the installation. Depending on your type of environment, some tools may not be necessary. The up-to-date list of the required and optional tools can be found in the User's Guide Core Framework at http://www-unix.globus.org/toolkit/3.0/ogsa/docs/users_guide.html.

Table 3-4 lists the files, or tools, we used in our lab. For more information about the tools, refer to Appendix C, "Software installed" on page 91.

The download of those files must be done during the m0 setup. For more information, refer to "Populate installation image repository" on page 77.

Table 3-4   Tools used in the lab environment

Tool	Description
Java SDK	The underlying code of GT3 is written in Java, so it is necessary to install the Java platform on any machines running the toolkit.
Apache Ant	A Java-based build tool that is required for the GT3 installation. The toolkit is composed of many Java classes that need to be combined or built to form a functioning program. Apache Ant is used to execute an Ant-based build script that automates the build process.

Tool	Description
Junit	Java-based testing framework that facilitates regression tests. The junit.jar class needs to be included with Ant for the GT3 installation to run tests.
PostgreSQL	Open source relational database management system released by the PostgreSQL Global Development Group that is compliant with the JDBC Java API.
Globus Toolkit	Globus Toolkit bundle Version 3.0.1.

We recommend you take a look at the Globus Toolkit Advisories Web site, which includes fix packages (for security and bugs) and also some packages of enhancements. This Web site is located at:

http://www-unix.globus.org/toolkit/advisories.html

Note: Although this documentation uses Java SDK Version 1.4.1, we suggest using Version 1.4.2.

3.3 Setting up the environment and requirements

This section is a cookbook for setting up the Grid environment with GT3. There are dependencies that require the installation to be performed in this order. The major steps to set up the Grid environment include:

► Install and configure the infrastructure server on m0. Go to Appendix A, "Infrastructure server setup" on page 73 to learn how to set up the m0 machine.

► Install the Red Hat Linux workstation package on the x1 and x2 machines.

► Mount the required NFS directories on x1 and x2.

► Install GT3 and its prerequisites on x1 and x2.

► Configure GT3 on x1 and x2.

Attention: Do not forget to follow all the steps of this installation section on all Grid servers involved in your environment, in our case, x1 and x2.

In order to facilitate the comprehension, all examples will be shown for machine x1.

3.3.1 Install and configure Linux

Install Linux on all of the machines that will be part of the Grid. Select the **Workstation** package and **No firewall**. Do not use DHCP. Each system should use a fixed network IP address with a corresponding host name, as defined in Table 3-1 on page 24.

Note: At the time this Redpaper was written, the binary components of the GT3 installation package had a conflict when building with a glibc library on Red Hat Version 9.0.

As the globus-grim component is built statically, the source package must be used when Red Hat Version 9.0 is being used. Therefore, we decided to describe the installation of the binary package on Red Hat Version 8.0 and the installation of the source package on Red Hat Version 9.0. Choose the one that matches your needs.

Configure NTP

For the NTP configuration, you can choose the m0 machine at 192.168.0.10.

Configure user and group IDs

Use Table 3-2 on page 25 to create user IDs at install time; otherwise, the IDs can be added later.

Add users that will be used to install both the toolkit and the certificate authority, as well as submit jobs. In our installation, we called the users *globus* and *itso*. Users can be added by using the `adduser` command followed by the `passwd` command to give the user a password.

As root, issue the following commands for this purpose:

```
# adduser globus
# passwd globus
# adduser itso
# passwd itso
```

Copy /etc/hosts

In our lab, we decided to use a flat scheme for the name server and not use DNS, so the /etc/hosts file created during the infrastructure server setup can be copied from m0.

As root, issue the following command to copy /etc/hosts from m0:

```
# scp 192.168.0.10:/etc/hosts /etc/hosts
```

Mount m0 directories

NFS is used to export the necessary directory structure provided by the m0 machine. For more information about the directory structure of m0, refer to Table A-5 on page 76.

As root, issue the following commands to mount the necessary directories:

```
# mkdir -p /mnt/m0
# mount m0:/expgt3 /mnt/m0
# mkdir -p /mnt/CA
# mount m0:/CA /mnt/CA
```

3.3.2 Java SDK

The underlying code of GT3 is written in Java, so it is necessary to install the Java platform on any machines running the toolkit. For our installation, we chose to install Sun Microsystem's Java Software Developer Kit (SDK), as you can see in "Tools" on page 92. This package includes the Java APIs and classes necessary for the toolkit to function properly.

More information about Java SDK can be found at the following Web site:

http://java.sun.com

You may download and install the Java Runtime Edition (JRE) version as opposed to the SDK. However, the SDK includes a compiler and debugger that are necessary if you choose to build your own binaries from the source code at a later date. For this reason, we chose to install the SDK.

The file downloaded from Sun is an executable that is first run on m0 to create a RPM. Next, the RPM is installed using the `rpm` command. Make sure the RPM package of SDK is presented on m0.

As root, issue the following commands for this purpose.

```
# cd /mnt/m0/sun-java/
# rpm -ivh j2sdk-1_4_1_03-fcs-linux-i586.rpm
```

The necessary environment variable (JAVA_HOME) setup will be performed in a later step in 3.3.4, "Set environment variables" on page 29.

3.3.3 Apache Ant and Junit

Apache Ant is a Java-based build tool that is required for the GT3 installation. The toolkit is composed of many Java classes that need to be combined or built to form a functioning program. Apache Ant is used to execute an Ant-based build script that automates the build process. In addition, if the toolkit needs to be re-built at any point, Apache Ant can skip actions that have already been completed.

More information about Apache Ant can be found at the following Web site:

http://ant.apache.org

Junit is a Java-based testing framework that facilitates regression tests. The junit.jar class needs to be included with Ant for the GT3 installation, since we run tests from source. More information about Junit can be found at the following Web site:

http://www.junit.org

The installation of Apache Ant requires only one command. To install, you simply go to the directory where apache-ant-1.5.3-1-bin.tar.gz is located and use the **tar** command. A new directory containing the decompressed files will be created for you automatically.

To complete the installation, junit.jar needs to be placed in the lib directory of Ant. First, unzip junit3.8.1.zip using the **unzip** command, and then copy it to /usr/local/apache-ant-1.5.3-1/lib.

As root, issue the following commands for this purpose:

```
# cd /usr/local/
# tar -xzvf /mnt/m0/apache/apache-ant-1.5.3-1-bin.tar.gz
# unzip /mnt/m0/junit/junit3.8.1.zip
# cp junit3.8.1/junit.jar apache-ant-1.5.3-1/lib
```

Example 3-1 shows the output of the install process.

Example 3-1   Install Apache Ant output

```
[root@x1 root]# cd /usr/local/
[root@x1 local]# tar -xzvf /mnt/m0/apache/apache-ant-1.5.3-1-bin.tar.gz
apache-ant-1.5.3-1/bin/ant
apache-ant-1.5.3-1/bin/runant.pl
apache-ant-1.5.3-1/bin/antRun
apache-ant-1.5.3-1/bin/runant.py
apache-ant-1.5.3-1/bin/antRun.pl
apache-ant-1.5.3-1/bin/complete-ant-cmd.pl
apache-ant-1.5.3-1/
apache-ant-1.5.3-1/bin/
apache-ant-1.5.3-1/lib/

...(author omits lines)

apache-ant-1.5.3-1/etc/log.xsl
apache-ant-1.5.3-1/LICENSE.xerces
apache-ant-1.5.3-1/KEYS
apache-ant-1.5.3-1/LICENSE.sax
apache-ant-1.5.3-1/LICENSE.dom
apache-ant-1.5.3-1/WHATSNEW
apache-ant-1.5.3-1/welcome.html
apache-ant-1.5.3-1/LICENSE
```

Example 3-2 shows the output of the installation process for Junit.

Example 3-2   Install Junit output

```
[root@x1 local]# unzip /mnt/m0/junit/junit3.8.1.zip
Archive:  /mnt/m0/junit/junit3.8.1.zip
   creating: junit3.8.1/
   creating: junit3.8.1/doc/
   creating: junit3.8.1/doc/cookbook/
   creating: junit3.8.1/doc/cookstour/
   creating: junit3.8.1/doc/faq/
   creating: junit3.8.1/doc/testinfected/
   creating: junit3.8.1/javadoc/

...(author omits lines)

  inflating: junit3.8.1/junit/tests/runner/TextRunnerTest$1.class
  inflating: junit3.8.1/junit/tests/runner/TextRunnerTest.class
  inflating: junit3.8.1/junit/tests/runner/TextRunnerTest.java
  inflating: junit3.8.1/junit/tests/WasRun.class
  inflating: junit3.8.1/junit/tests/WasRun.java
  inflating: junit3.8.1/junit.jar
  inflating: junit3.8.1/README.html
  inflating: junit3.8.1/src.jar
```

3.3.4 Set environment variables

Environment variables need to be set, as shown in Table 3-5, in order for the location of Java and Ant to be known throughout the system.

Table 3-5   Variables

Variable	Description	Value
JAVA_HOME	Base directory of Sun Java	/usr/java/j2sdk1.4.1_03
ANT_HOME	Base directory of Apache Ant	/usr/local/apache-ant-1.5.3-1

Tip: Instead of placing the export statements in /etc/profile, they may be placed in .bashrc in order to be accessible only by a specific user. Placing the statements in .bashrc requires that they be copied into the .bashrc file of every user that needs access to the Globus environment. In a production environment, we might choose to place the statements in .bashrc in order to restrict access to the environment by unauthorized users.

Make sure you place the variable declarations in the correct position, prior to the existing export PATH statement.

As root, use your preferred editor to modify the /etc/profile file, as in Example 3-3 on page 30.

Example 3-3   New /etc/profile

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

JAVA_HOME=/usr/java/j2sdk1.4.1_03
ANT_HOME=/usr/local/apache-ant-1.5.3-1
PATH=$JAVA_HOME/bin:$ANT_HOME/bin:$PATH
export JAVA_HOME
export ANT_HOME

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

...(author omits lines)
```

Next, the `ant` command is used to build the code. Before running the `ant` command, source /etc/profile so that the necessary variables are set.

As root and globus, issue the following commands for this purpose.

```
# . /etc/profile
```

3.3.5 JDBC and PostgreSQL

A Java Database Connectivity (JDBC) compliant database must be used by GT3 for the RFT service and RLS. Although GT3 includes PostgreSQL JDBC and is used in our lab installation, any JDBC compliant database could be able to be configured for use by GT3. JDBC is a API for Java that allows access to a wide range of SQL databases. JDBC is similar to the open standard API Open Database Connectivity (ODBC), which is aligned with The Open Group. More information about The Open Group can be found at `http://www.opengroup.org/`.

More information about JDBC can be found at `http://java.sun.com/products/jdbc/`.

PostgreSQL is an open source relational database management system released by the PostgreSQL Global Development Group, which is compliant with the necessary JDBC Java API.

More information about PostgreSQL can be found at `http://www.postgresql.com`.

Check previous PostgreSQL installation

The Red Hat distribution includes the PostgreSQL RPMs and may be installed by choosing either the server installation type or choosing to install all of the RPMs in a custom installation. For this reason, if you chose not to install a fresh copy of Linux, we recommend that you check to see if the RPMs are already installed on your system.

As root, issue the following command for this purpose:

```
# rpm -qa | grep sql
```

If the PostgreSQL RPMs are listed, continue with configuring /etc/init.d/postgresql, which is discussed in "Configure PostgreSQL" on page 31.

Install PostgreSQL

PostgreSQL consists of 11 RPMs, but only three of them are required to install a functioning database. If you wish, you may chose to install of all the RPMs, as illustrated in "Tools" on

page 92. The installation process consists of installing the RPMs and then making changes to two configuration files.

> **Attention:** The RPM packages for Red Hat Version 9.0 and Red Hat Version 8.0 are not the same. Make sure you get the right package. The examples and commands presented in this section show the installation on Red Hat Version 9.0.

First, install the RPMs by using the `rpm` command. Several RPMs are dependent upon other packages and must be installed in the *exact* order shown in Example 3-4.

In Red Hat Version 9.0, as root, issue the commands in for this purpose.

Example 3-4   Install PostgreSQL RPMs 7.3.3 on Red Hat Version 9.0

```
# rpm -ivh /mnt/m0/postgresql/postgresql-libs-7.3.3-1PGDG.i386.rpm
# rpm -ivh /mnt/m0/postgresql/postgresql-7.3.3-1PGDG.i386.rpm
# rpm -ivh /mnt/m0/postgresql/postgresql-server-7.3.3-1PGDG.i386.rpm
```

Example 3-5 shows the output of these commands.

Example 3-5   Install PostgreSQL RPMs 7.3.3 on Red Hat Version 9.0 output

```
[root@x1 root]# rpm -ivh /mnt/m0/postgresql/postgresql-libs-7.3.3-1PGDG.i386.rpm
Preparing... ######################################### [100%]
1:postgresql-libs ######################################### [100%]
[root@x1 root]# rpm -ivh /mnt/m0/postgresql/postgresql-7.3.3-1PGDG.i386.rpm
Preparing... ######################################### [100%]
1:postgresql ######################################### [100%]
[root@x1 root]# rpm -ivh /mnt/m0/postgresql/postgresql-server-7.3.3-1PGDG.i386.rpm
Preparing... ######################################### [100%]
1:postgresql-server ######################################### [100%]
```

> **Note:** The PostgreSQL installation process creates the user postgres. This user will be used later during the creation of the database.

Configure PostgreSQL

Next, the postgresql configuration file located in /etc/init.d needs to be altered in order to allow clients to connect via TCP/IP-based connections. After opening the file in any editor, search for the line that starts with the `postmaster` command. It is around line 157 on PostgreSQL Version 7.3.3 running on Red Hat Version 9.0. As presented in Example 3-6 and Example 3-7 on page 32, the -i flag needs to be added after the -o flag, and `'-p ${PGPORT}'` needs to be removed from the line.

Example 3-6   Original line on /etc/init.d/postgresql

```
...(author omits lines)

su -l postgres -s /bin/sh -c "/usr/bin/pg_ctl -D $PGDATA -p /usr/bin/postmaster -o '-p
${PGPORT}' start  > /dev/null 2>&1" < /dev/null

...(author omits lines)
```

As root, use your preferred editor to modify the postgresql file, as shown in Example 3-7 on page 32.

Example 3-7   Modified line on /etc/init.d/postgresql

```
...(author omits lines)

su -l postgres -s /bin/sh -c "/usr/bin/pg_ctl -D $PGDATA -p /usr/bin/postmaster -o -i start
> /dev/null 2>&1" < /dev/null

...(author omits lines)
```

Start PostgreSQL

After saving the file, start the database.

As root, issue the following command for this purpose:

```
# /etc/init.d/postgresql start
```

To complete the installation of PostgreSQL, the database needs to be told to listen for requests on the host machine. This is accomplished by altering the pg_hba.conf file (host-based authentication), which is located in the /var/lib/pgsql/data directory. Machines that want to access the database must be defined in the pg_hba.conf file. The host stanza (or record) contained in the file must match the users and IP address authentications. For example, in order to allow any user from any machine with an 192.168.x.x address to access any database, the following host record should be:

```
host       all        192.168.0.0   255.255.0.0    trust
```

Do not forget to remove the # character (comment character) from the beginning of the line. In Version 7.3.3, we found the host stanza in line 48:

```
...(author omits lines)
#host       all        127.0.0.1     255.255.255.255    trust
...(author omits lines)
```

As root, use your preferred editor to modify the /var/lib/pgsql/data/pg_hba.conf host stanza file, as follows:

```
...(author omits lines)
host       all        192.168.0.0   255.255.0.0  trust
...(author omits lines)
```

3.4 Globus Toolkit Installation

This section presents the installation and configuration of GT3. There are dependencies that require the installation to be performed in this order. More information about the installation process can be found in the Globus Toolkit 3.0 Documentation Web page at:

http://www-unix.globus.org/toolkit/documentation.html

At the time this Redpaper was written, the binary components of the GT3 installation package had a conflict when building with the glibc library on Red Hat Version 9.0. So, two installation processes are presented here: one using the binary package (with Red Hat Version 8.0) and the other one with a source package (with Red Hat Version 9.0). Choose the one that better fits your environment: 3.4.1, "Binary package installation with Red Hat Version 8.0" on page 33, or 3.4.2, "Source package installation with Red Hat Version 9.0" on page 35.

The Globus packages used to perform the following installations are described in Table C-5 on page 93.

In our lab environment, we installed and configured GT3 in both machines: x1 and x2.

> **Tip:** Once you have a configured and stable environment (after the Globus Toolkit installation and configuration), we recommend you back up the $GLOBUS_LOCATION directory; if there are problems due to additional configurations, restoring a backup would be a faster way to recover the system.

3.4.1 Binary package installation with Red Hat Version 8.0

As a security precaution, GT3 needs to be installed by a non-root user. The reason for this lies in the fact that many of the services are designed to be run with limited access to the Globus environment; installing as root negates this important feature.

> **Tip:** Instead of switching users repeatedly with the `su - <username>` command, you may find it useful to have a separate window opened for each user. For example, open a new window for root, globus, and itso.

Untar GT3 binary package

The software is installed in a similar fashion to Java. First, you must create the globus directories, as presented in Table 3-3 on page 25. Also, the ownership of the directories need to be changed. Then, switch to globus user and "untar" the Globus binary package.

First, as root, issue the following commands to create the directory:

```
# mkdir -p /usr/local/globus
# chown globus:globus /usr/local/globus
```

Then, as the globus user, issue the following commands to untar the file:

```
# su - globus
$ mkdir install
$ cd install
$ tar -xzvf /mnt/m0/globus/gt3.0.1-linux-installer.tar.gz
```

Example 3-8 contains a partial output of the file.

Example 3-8   Untar GT3 output

```
[root@x1 root]# mkdir -p /usr/local/globus
[root@x1 root]# chown globus:globus /usr/local/globus
[root@x1 root]# su - globus
[globus@x1 globus]$ mkdir install
[globus@x1 globus]$ cd install
[globus@x1 install]$ tar -xzvf /mnt/m0/globus/gt3.0.1-linux-installer.tar.gz
gt3.0.1-linux-installer/
gt3.0.1-linux-installer/install-gt3
gt3.0.1-linux-installer/install-gt3-mmjfs

...(author omits lines)

gt3.0.1-linux-installer/globus_grim-0.3.tar.gz
gt3.0.1-linux-installer/ogsa_cbindings_src_bundle_src.tar.gz
gt3.0.1-linux-installer/globus_libxml2-0.2.tar.gz
gt3.0.1-linux-installer/mmjfs-all-src-1.1-src_bundle.tar.gz
gt3.0.1-linux-installer/globus_gsi_sysconfig-0.10.tar.gz
gt3.0.1-linux-installer/globus-rls-server-2.0.8-src_bundle.tar.gz
```

Install GT3 binary package

The toolkit will be installed using the gt3-installer script. We recommend capturing the output log in a file. To create a log of the installation process for debugging, add the **tee** command to the end of the installation command, as shown in Example 3-9. A log file will be created and placed into the current directory, which can be opened by any editor.

As the globus user, issue the commands in Example 3-9 for this purpose.

Example 3-9   Install toolkit

```
$ cd ~/install/gt3.0.1-linux-installer
$ ./install-gt3 /usr/local/globus | tee installgt3.log
```

Note: The installation can take anywhere from 15 minutes to several hours, depending upon the processor and available memory of the machine.

Note: We experienced some problems during the installation. On occasion, the installation execution froze and it was necessary to "kill" the frozen process and re-issue the installation command. We were not able to determine the cause of this problem.

Example 3-10 contains the output of the commands ran in Example 3-9.

Example 3-10   Install toolkit output

```
[globus@x1 install]$ cd ~/install/gt3.0.1-linux-installer
[globus@x1 gt3.0.1-linux-installer]$ ./install-gt3 /usr/local/globus | tee installgt3.log
Build environment:
ant is hashed (/usr/local/apache-ant-1.5.3-1/bin/ant)
java is /usr/java/j2sdk1.4.1_03/bin/java
gcc is /usr/bin/gcc

Building GPT ...
build_gpt ====> installing GPT into /usr/local/globus
build_gpt ====> building support/Compress-Zlib-1.16
build_gpt ====> building support/Archive-Tar-0.22
build_gpt ====> building support/PodParser-1.18

...(author omits lines)

Creating...
        /usr/local/globus/libexec/grid-hostinfo-script-initializer
        /usr/local/globus/libexec/grid-hostinfo-common
        /usr/local/globus/libexec/grid-hostinfo-cpu*
        /usr/local/globus/libexec/grid-hostinfo-fs*
        /usr/local/globus/libexec/grid-hostinfo-mem*
        /usr/local/globus/libexec/grid-hostinfo-net*
        /usr/local/globus/libexec/grid-hostinfo-platform*
        /usr/local/globus/libexec/grid-hostinfo-os*
        /usr/local/globus/libexec/grid-hostinfo-host
        /usr/local/globus/libexec/grid-hostinfo-partial-host
        /usr/local/globus/libexec/grid-hostinfo-partial-subcluster
        /usr/local/globus/libexec/grid-hostinfo-partial-cluster
        /usr/local/globus/libexec/grid-hostinfo-runtime
        /usr/local/globus/etc//globus-host-providers.conf
Done
```

3.4.2 Source package installation with Red Hat Version 9.0

As a security precaution, GT3 needs to be installed by a non-root user. The reason for this lies in the fact that many of the services are designed to be run with limited access to the Globus environment; installing as root negates this important feature.

> **Tip:** Instead of switching users repeatedly with the `su - <username>` command, you may find it useful to have a separate window opened for each user. For example, open a new window for root, globus, and itso.

Untar GT3 source package

The software is installed in a similar fashion to Java. First, you must create the globus directories shown in Table 3-3 on page 25. Also, the ownership of the directories need to be changed. Then, switch to globus user and "untar" the Globus source package.

First, as root, issue the following commands to create the directory:

```
# mkdir -p /usr/local/globus
# chown globus:globus /usr/local/globus
```

Then, as the globus user, issue the following commands to untar the file:

```
# su - globus
$ mkdir install
$ cd install
$ tar -xzvf /mnt/m0/globus/gt3.0.1-source-installer.tar.gz
```

You should get the output shown in Example 3-11.

Example 3-11   Untar GT3 source output

```
[root@x1 root]# mkdir -p /usr/local/globus
[root@x1 root]# chown globus:globus /usr/local/globus
[root@x1 root]# su - globus
[globus@x1 globus]$ mkdir install
[globus@x1 globus]$ cd install
[globus@x1 install]$ tar -xzvf /mnt/m0/globus/gt3.0.1-source-installer.tar.gz
gt3.0.1-source-installer/
gt3.0.1-source-installer/install-gt3
gt3.0.1-source-installer/install-gt3-mmjfs
gt3.0.1-source-installer/gt3tutorial-0.2.tar.gz
gt3.0.1-source-installer/OGSI.NET.zip
gt3.0.1-source-installer/gpt-2.2.10-src.tar.gz
gt3.0.1-source-installer/globus_proxy_wrapper-1.0.tar.gz
gt3.0.1-source-installer/globus_gsi_credential-0.9.tar.gz
gt3.0.1-source-installer/globus_grim-0.3.tar.gz
gt3.0.1-source-installer/schedulers/
gt3.0.1-source-installer/schedulers/globus_gram_job_manager_setup_condor-1.3.tar.gz
gt3.0.1-source-installer/schedulers/globus_gram_job_manager_setup_lsf-1.3.tar.gz
gt3.0.1-source-installer/schedulers/globus_gram_job_manager_setup_pbs-1.5.tar.gz
gt3.0.1-source-installer/schedulers/globus_gram_job_manager_setup_remote-1.1.tar.gz
gt3.0.1-source-installer/schedulers/mjs_condor_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mjs_fork_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mjs_lsf_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mjs_pbs_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mmjfs_condor_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mmjfs_fork_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mmjfs_condor_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mmjfs_fork_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/mmjfs_lsf_setup-0.0.tar.gz
```

```
gt3.0.1-source-installer/schedulers/mmjfs_pbs_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/rips_condor_provider_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/rips_lsf_provider_setup-0.0.tar.gz
gt3.0.1-source-installer/schedulers/rips_pbs_provider_setup-0.0.tar.gz
gt3.0.1-source-installer/globus_grim_devel-0.2.tar.gz
gt3.0.1-source-installer/gt3_hostinfo-2.0.tar.gz
gt3.0.1-source-installer/ogsa_client_findServiceData-0.3.tar.gz
gt3.0.1-source-installer/gt3-all-src-1.0-src_bundle.tar.gz
gt3.0.1-source-installer/ogsa_cbindings_src_bundle_src.tar.gz
gt3.0.1-source-installer/globus_libxml2-0.2.tar.gz
gt3.0.1-source-installer/globus-data-management-client-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-data-management-sdk-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-data-management-server-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-information-services-client-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-information-services-sdk-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-information-services-server-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-resource-management-client-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-information-services-server-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-resource-management-client-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-resource-management-sdk-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-resource-management-server-2.4.1-src_bundle.tar.gz
gt3.0.1-source-installer/globus-rls-server-2.0.8-src_bundle.tar.gz
gt3.0.1-source-installer/globus_gsi_sysconfig-0.10.tar.gz
gt3.0.1-source-installer/mmjfs-all-src-1.1-src_bundle.tar.gz
[globus@x1 install]$
```

Install GT3 source package

The toolkit will be installed using the **install-gt3** command in /usr/local/globus. We recommend capturing the output log in a file. To create a log of the installation process for debugging, add the **tee installgt3.log** command to the end of the installation command, as shown in Example 3-12. A file named installgt3.log will be created and placed into the current directory, which can be opened in any editor.

As the globus user, issue the following commands for this purpose:

```
$ cd ~/install/gt3.0.1-source-installer
$ ./install-gt3 /usr/local/globus | tee installgt3.log
```

Note: The source installation can take several hours depending upon the processor and available memory of the machine.

You should get the output shown in Example 3-12.

Example 3-12   Install toolkit with source output

```
[globus@x1 globus]$ cd ~/install/gt3.0.1-source-installer/
[globus@x1 gt3.0.1-source-installer]$ ./install-gt3 /usr/local/globus | tee installgt3.log
Build environment:
ant is hashed (/usr/local/apache-ant-1.5.3-1/bin/ant)
java is /usr/java/j2sdk1.4.1_03/bin/java
gcc is /usr/bin/gcc

Building GPT ...
build_gpt ====> installing GPT into /usr/local/globus
build_gpt ====> building support/Compress-Zlib-1.16
build_gpt ====> building support/Archive-Tar-0.22

...(author omits lines)
```

```
****************************************************************************
Note: To complete setup of the GSI software you need to run the
following script as root to configure your security configuration
directory:

/usr/local/globus/setup/globus/setup-gsi

For further information on using the setup-gsi script, use the -help
option.  The -nonroot can be used on systems where root access is
not available.
****************************************************************************

setup-ssl-utils: Complete

running /usr/local/globus/setup/globus/setup-globus-gram-job-manager...
Creating state file directory.
Done.
Reading gatekeeper configuration file...
Warning: Host cert file: /etc/grid-security/hostcert.pem not found.  Re-run
        setup-globus-gram-job-manager after installing host cert file.
Determining system information...
Creating job manager configuration file...
Done
running /usr/local/globus/setup/globus/setup-globus-job-manager-fork...
configure: warning: Cannot locate mpirun
loading cache ./config.cache
checking for mpirun... no
updating cache ./config.cache
creating ./config.status
creating fork.pm
[globus@x1 globus]$
```

> **Note:** On GT3 installation, you may receive a warning message similar to the following:
>
> ```
> Warning: Host cert file: /etc/grid-security/hostcert.pem not found. Re-run
>         setup-globus-gram-job-manager after installing host cert file.
> ```
>
> This error occurs because you have not set up the Grid security yet. This will be done later
> in this chapter. This issue can be cleared up by running the
> **setup-globus-gram-job-manager** command after retrieving the certificates and before
> installing MMJFS.

3.5 Globus Toolkit post-installation activities

This section presents the post installation activities that must be performed after GT3
installation. The major steps are:

► Set environment variables
► Substitute xalan.jar
► Set the proper CA
► Configure GSI

3.5.1 Set environment variables

Environment variables need to be set in order for the location of Globus to be known
throughout the system. In addition, the globus-user-env.sh script needs to run. This is
accomplished by adding the lines in Table 3-6 on page 38 and Example 3-13 on page 38 into

/etc/profile. Make sure you place the variable declarations in the correct position prior to the existing export PATH statement.

Table 3-6   GLOBUS_LOCATION variable

Variable	Description	Value
GLOBUS_LOCATION	Base directory of Globus	/usr/local/globus

Example 3-13   To be added into /etc/profile

```
PATH=$PATH:$JAVA_HOME/bin:$ANT_HOME/bin:$GLOBUS_LOCATION/bin
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

As root, use your preferred editor to add these lines into /etc/profile as follows:

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

JAVA_HOME=/usr/java/j2sdk1.4.1_03
ANT_HOME=/usr/local/apache-ant-1.5.3-1
GLOBUS_LOCATION=/usr/local/globus
PATH=$PATH:$JAVA_HOME/bin:$ANT_HOME/bin:$GLOBUS_LOCATION/bin
export JAVA_HOME
export ANT_HOME
export GLOBUS_LOCATION

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

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

...(author omits lines)
```

Once these lines are added to /etc/profile file, you must source /etc/profile so that the necessary variables are set.

As root and globus, issue the following command to source /etc/profile:

```
# . /etc/profile
```

3.5.2 Substitute xalan.jar

The GT3 package provides a newer version of the xalan.jar, which is used by GT3 security libraries. Therefore, Java classes in the xalan.jar file must be included in the Java directory.

As root, issue the following commands for this purpose:

```
# mkdir -p $JAVA_HOME/jre/endorsed
# cp $GLOBUS_LOCATION/endorsed/xalan.jar  $JAVA_HOME/jre/endorsed
```

3.5.3 Configure GSI

Before configuring Grid Security Infrastructure (GSI) make sure the certificate authority is successfully installed and configured. See "Create a certificate authority" on page 80.

In our lab environment, we used the m0.itso-maya.com machine as a certificate authority. Table 3-7 on page 39 describes the values to be entered for each prompt.

Table 3-7   GSI setup prompts

Prompt	Enter	Description
`Do you wish to continue (y/n)`	y	Starts setup

`(1) Base DN for user certificates`
`(2) Base DN for host certificates`
`(q) save, configure and Quit`
`(c) Cancel`
`(h) Help`
	1	Request user DN modification

`Enter the Base Distinguish Name (DN)` `for user certificates`	ou=XINGU, o=ITSO	Modify user DN

`(1) Base DN for user certificates`
`(2) Base DN for host certificates`
`(q) save, configure and Quit`
`(c) Cancel`
`(h) Help`
	2	Request host DN modification

`Enter the Base Distinguish Name (DN)` `for host certificates`	o=ITSO	Modify host DN

`(1) Base DN for user certificates`
`(2) Base DN for host certificates`
`(q) save, configure and Quit`
`(c) Cancel`
`(h) Help`
	q	To save, configure GSI and quit

> **Note:** The values presented here to configure the CA and GSI are for demonstration purposes only. In a production environment, a deep analysis involving the security and system management teams must take place.

As root, issue the following command to configure GSI:

`# $GLOBUS_LOCATION/setup/globus/setup-gsi`

You should get an output similar to Example 3-14.

Example 3-14   GSI setup output

```
[root@x3 root]# $GLOBUS_LOCATION/setup/globus/setup-gsi
setup-gsi: Configuring GSI security
Installing /etc/grid-security/certificates//grid-security.conf.42864e48...
Running grid-security-config...

  G S I  :  C O N F I G U R A T I O N   P R O C E D U R E


Before you use the Grid Security Infrastructure, you should first
define the DN (distinguished name) that should be used for your
organization's X509 certificates.  If you do not define a DN,
a default DN will be assigned to you.

This script will ask some questions about site specific
information. This information is used to configure
the Grid Security Infrastructure for your site.

For some questions, a default response is given in [].
Pressing RETURN in response to such a question will enable the default.
This script will overwrite the file --
```

```
         /etc/grid-security/certificates//grid-security.conf.42864e48


Do you wish to continue (y/n) [y] :
=======================================================================


(1) Base DN for user certificates
        [ ou=itso-xingu.com, o=Globus, o=Grid ]
(2) Base DN for host certificates
        [ o=Globus, o=Grid ]


=======================================================================
(q) save, configure the GSI and Quit
(c) Cancel (exit without saving or configuring)
(h) Help
=======================================================================


1
Enter the Base Distinguish Name (DN) for user certificates [ ou=itso-xingu.com, o=Globus,
o=Grid ] :
ou=XINGU, o=ITSO
=======================================================================


(1) Base DN for user certificates
        [ ou=XINGU, o=ITSO ]
(2) Base DN for host certificates
        [ o=Globus, o=Grid ]


=======================================================================
(q) save, configure the GSI and Quit
(c) Cancel (exit without saving or configuring)
(h) Help
=======================================================================


2
Enter the Base Distinguish Name (DN) for host certificates [ o=Globus, o=Grid ] :
o=ITSO
=======================================================================


(1) Base DN for user certificates
        [ ou=XINGU, o=ITSO ]
(2) Base DN for host certificates
        [ o=ITSO ]


=======================================================================
(q) save, configure the GSI and Quit
(c) Cancel (exit without saving or configuring)
(h) Help
=======================================================================


q
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

For our installation, we accepted the default distinguished name (DN) by typing q and
pressing Enter.

Copy CA certificate/public key

In order to use the certificate authority provided on m0, we must copy the CA certificate/public key file from there. This file has a special file name convention, as described in "Public key" on page 83.

As root, issue the following command to find the "hash" file name of the certificate/public key of the CA:

```
# ssh m0 ls -l /CA
```

You should get an output similar to Example 3-15.

Example 3-15   Checking the certificate/public key filename output

```
[root@x1 root]# ssh m0 ls -l /CA
root@m0's password:
7000141d.0
IN
OUT
PROCESSED
demoCA
openssl.cnf
```

> **Note:** In our particular case, the file name of the certificate/public key of the CA is 7000141d.0. You might have a different file name in your environment.

As root, issue the following command to copy the certificate/public key of the CA to the Grid security certificate directory (/etc/grid-security/certificates):

```
# scp m0:/CA/7000141d.0 /etc/grid-security/certificates/
```

Finalizing GSI setup

Depending on each environment, some additional customization needs to be performed to finish the GSI setup and CA configuration. Since the scope of this documentation does not focus on security, we only present the minimum tasks necessary to run a demonstration environment.

The security configuration files that may be modified to match your environment are located in the /etc/grid-security/certificates directory and they are described in Table 3-8. These files are delivered by the tar.gz ball of GT3. The `setup-gsi` command copies them into the right places, and you can make this configuration the default configuration by linking the files to /etc/grid-security.

Table 3-8   Security configuration files

Security files	Description
<hash>.0	CA certificate/public key.
<hash>.signing_policy	Signing policies for CA (not used in our demo environment, because the signature process does not required policy; see the policy_anything flag in "Sign a certificate" on page 88).
globus-host-ssl.conf.<hash>	Create during the GSI setup. It contains configuration parameters used on host certificate requests.
globus-user-ssl.conf.<hash>	Create during the GSI setup. It contains configuration parameters used on user certificate requests.

Security files	Description
grid-security.conf.<hash>	Create during the GSI setup. It contains configuration parameters for GSI.
../globus-host-ssl.conf	Link to /etc/grid-security/certificates//globus-host-ssl.conf.<hash>.
../globus-user-ssl.conf	Link to /etc/grid-security/certificates//globus-user-ssl.conf.<hash>.
../grid-security.conf	Link to /etc/grid-security/certificates//grid-security.conf.<hash>.

3.6 Globus Toolkit configuration

Several items need to be configured to complete the installation of GT3, including:

- ► Request and sign certificates
- ► Install MMJFS
- ► Create a database
- ► Create two Grid security files

Each of these tasks will be covered separately in the following sections.

3.6.1 Requesting and signing certificates

A certificate is necessary for each user (known as a user certificate) who will use the Grid, as well as for each machine (known as a host certificate) that is serving in the Grid.

The following procedure explains the necessary steps to create and sign the certificates:

1. Request the user certificate for the end user.

2. Request the host certificate for the machines.

3. Send the certificates to the CA.

4. Have the CA sign the certificates.

5. Retrieve the signed certificates from the CA.

Request the user certificate

A user certificate is requested by issuing the **grid-cert-request** command. In our lab, the user (or ID) name is itso, as described in the Table 3-2 on page 25. This needs to be done just once, regardless of the number of Grid servers involved in the Grid.

Attention: The request of the user certificate is done in itso user.

As the itso user, issue the following command for this purpose:

```
$ grid-cert-request
```

In our installation, we named the user requesting the certificate itso to match the user name. The output of the command is shown in Example 3-16 on page 43.

Example 3-16   Request user certificate output

```
[root@x1 root]# su - itso
[itso@x1 itso]$ grid-cert-request

Enter your name, e.g., John Smith: itso
Enter PEM pass phrase: <passphrase>

...(author omits lines)
```

After issuing the **grid-cert-request** command, a file named usercert_request.pem is stored in the /home/itso/.globus directory.

Request the host certificate

A host certificate is requested by issuing the **grid-cert-request** command with the -service and -host parameters. The host certificate is requested in a similar fashion to the way the user certificate was requested.

> **Attention:** The request of the host certificate is done by the root user, and must be done for every Grid server involved in the Grid, such as x1 and x2. See Table 3-1 on page 24.

As root, issue the following command for this purpose:

```
grid-cert-request -service host -host x1.itso-xingu.com
```

Example 3-17 gives the output of this command.

Example 3-17   Request host certificate output

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

/O=ITSO/CN=host/x1.itso-xingu.com

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

The private key is stored in /etc/grid-security/hostkey.pem
The request is stored in /etc/grid-security/hostcert_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/hostcert_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
[root@x1 certificates]#
```

For more information about host names, refer to Table 3-1 on page 24.

> **Note:** If you need to reuse the **grid-cert-request** command to get a new certificate, the -force parameter can be used as follows:
>
> ```
> # grid-cert-request -service host -host x1.itso-xingu.com -force
> ```

After issuing the `grid-cert-request` command, a file named hostcert_request.pem is stored in the /etc/grid-security/ directory.

Send the certificates to the CA

The certificate request is made by copying the unsigned certificate to the /CA/IN directory of the CA machines (in our case, at m0.itso-maya.com). See Table A-5 on page 76 for details.

Because the certificate file names are the same regardless of the machines or users, we must copy them to the CA following the naming convention described in Table A-4 on page 75.

As root, issue the following commands for this purpose:

```
# scp /home/itso/.globus/usercert_request.pem m0:/CA/IN/x1usercert_request.pem
# scp /etc/grid-security/hostcert_request.pem m0:/CA/IN/x1hostcert_request.pem
```

You should get an output similar to Example 3-18.

Example 3-18   Send the certificates' output

```
[root@x1 root]# scp /home/itso/.globus/usercert_request.pem \
m0:/CA/IN/x1usercert_request.pem
usercert_request.pem 100% |*****************************|  1251        00:00
[root@x1 root]# scp /etc/grid-security/hostcert_request.pem \
m0:/CA/IN/x1hostcert_request.pem
hostcert_request.pem 100% |*****************************|  1243        00:00
```

Sign the certificates

Now the CA needs to sign the certificate. Log into the m0.itso-maya.com machine and sign the certificates. For that task, you can use the camgr tool, as presented in Figure 3-2 and explained in Appendix B, "Managing a certificate" on page 85.
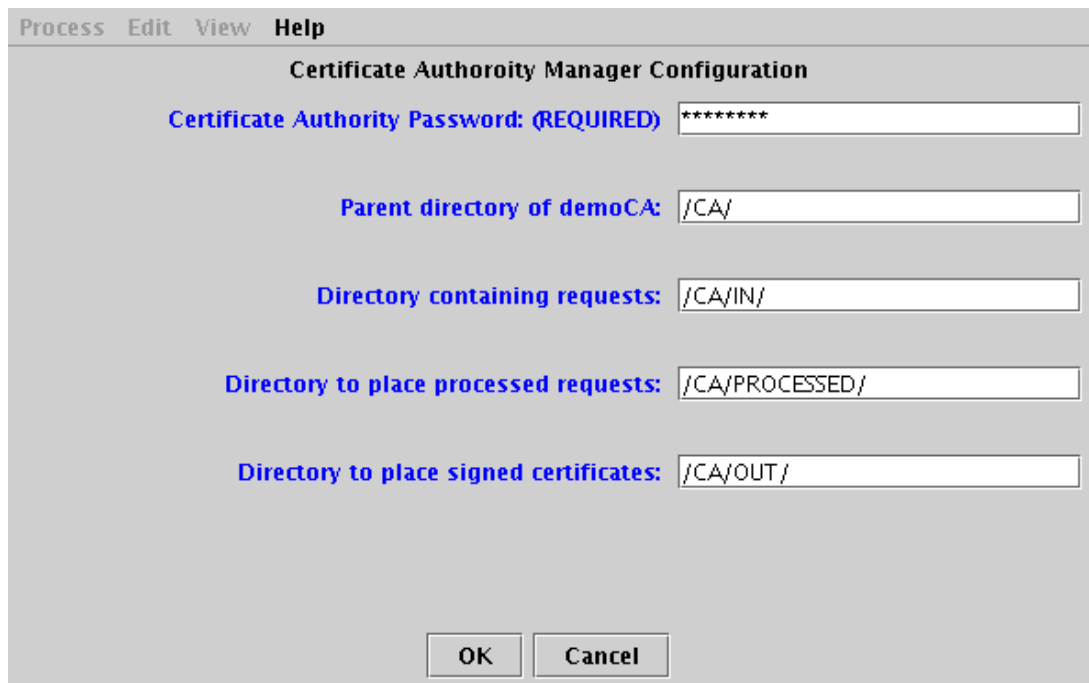

Figure 3-2   Running the camgr command

> **Attention:** The certificate signature is done by the CA, in our case, the m0.itso-maya.com machine.

As root on the m0 machine, issue the following commands for this purpose:

```
# camgr
```

To sign the certificates, follow instructions the presented in "Sign a certificate" on page 88.

Retrieve the signed certificates

The signed certificate is stored by the camgr tool in the /CA/OUT directory of the CA machine (in our case, m0 machine). You need to pay attention and copy/retrieve the signed certificate from the proper location, as described on Table A-5 on page 76.

> **Note:** Now you need to go back to the Grid server you are working, whether x1 or x2.

As root, issue the following commands for this purpose:

```
# scp m0:/CA/OUT/x1usercert_cert.pem /home/itso/.globus/usercert.pem
# scp m0:/CA/OUT/x1hostcert_cert.pem /etc/grid-security/hostcert.pem
```

You should get an output similar to Example 3-19.

Example 3-19   Retrieve signed certificates' output

```
[root@x1 root]# scp m0:/CA/OUT/x1usercert_cert.pem /home/itso/.globus/usercert.pem
root@m0's password: <password>
x1usercert_cert.pem 100% |****************************|  1251       00:00
[root@x1 root]# scp m0:/CA/OUT/x1hostcert_cert.pem /etc/grid-security/hostcert.pem
root@m0's password: <password>
x1hostcert_cert.pem 100% |****************************|  1251       00:00
```

> **Tip:** More information about certificate requests, configuring a CA, and GSI configuration files can be found at:
>
> http://www.globus.org/security/config.html

3.6.2 Install MMJFS

After configuring the certificates, you are able to install the Master Managed Job Factory Service (MMJFS) by issuing the **install-gt3-mmjfs** command script. MMJFS is the single point for submitting jobs. It is responsible for exposing the virtual GRAM service to the outside world. For more information about MMJFS, refer to "Job management services" on page 19.

As shown previously, we recommend capturing the output log in a file by adding | **tee installmmjfs.log** to the end of the installation command.

As the globus user, issue the following commands to install the MMJFS *source* bundle:

```
$ cd ~/install/gt3.0.1-source-installer
$ ./install-gt3-mmjfs /usr/local/globus | tee installmmjfs.log
```

Or, as the globus user, issue the following commands to install MMJFS *binary* bundle:

```
$ cd ~/install/gt3.0.1-linux-installer
$ ./install-gt3-mmjfs /usr/local/globus | tee installmmjfs.log
```

Either way, you should get an output similar to Example 3-20.

Example 3-20   Install MMJFS source bundle output

```
[globus@x1 globus]$ cd ~/install/gt3.0.1-linux-installer
[globus@x1 gt3.0.1-linux-installer]$ ./install-gt3-mmjfs /usr/local/globus | tee
installmmjfs.log
Build environment:
ant is hashed (/usr/local/apache-ant-1.5.3-1/bin/ant)
java is /usr/java/j2sdk1.4.1_03/bin/java
gcc is /usr/bin/gcc

gpt-build ====> CHECKING BUILD DEPENDENCIES FOR globus_core
SKIPPING REBUILD of gcc32dbg
WARNING: "/usr/local/globus//" not found
gpt-build ====> CHECKING BUILD DEPENDENCIES FOR grim
gpt-build ====> Changing to
/home/globus/installbin/gt3.0.1-linux-installer/BUILD/grim-0.3-src//grim

...(author omits lines)

deployServer:
     [echo] deploying server config...

testClientDeployAvailable:

deployClient:

generateUndeploy:

setGarID:

testPostDeployAvailable:

setAbsoluteGlobusLocation:

postDeploy:
   [delete] Deleting directory /usr/local/globus/build/gar
     [copy] Copying 1 file to /usr/local/globus

BUILD SUCCESSFUL
Total time: 12 seconds
```

3.6.3 Change the ownership and access permission

Upon completion, run the setperms.sh script located in the /usr/local/globus/bin directory to change the ownership of some Globus files under $GLOBUS_LOCATION/bin directory. This is necessary to allow some Globus tools to run as root.

As root, issue the following commands for this purpose:

```
# cd $GLOBUS_LOCATION/bin
# ./setperms.sh
```

3.6.4 Create the PostgreSQL database

Before creating the database, make sure the postgres daemon is running.

You can use the **ps** command for verification:

```
# ps -aux | grep postgres
```

If the postgres daemon is not working, refer to "Start PostgreSQL" on page 32, to start it.

Change to user postgres, which was created when the database is installed, and use the **createuser** command to create a new database user named globus. Allow the new postgres user to both create databases as well as more users by typing y at both prompts.

First, go to user postgres and issue the **createuser** command to create the database user called globus (see Example 3-21). Reply y in both prompts.

Example 3-21   Create database user

```
# su - postgres
$ createuser globus
```

You should get an output similar to Example 3-22.

Example 3-22   Create database user output

```
[root@x1 globus]# su - postgres
bash-2.05a$ createuser globus
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER
```

> **Note:** Do not confuse Linux users and database users. In our case, we have a Linux user called globus and then we created a database user called globus as well.

Next, create the database by using the **createdb** command as the globus user, to be used by the RFT service. The database created will be called ogsa and the schema used to create its tables is called rft_schema_ogsa.sql. This file is located in /usr/local/globus/etc/. Create the database using the **createdb** command and create its tables using the **psql** command.

As globus user, issue the following commands for this purpose:

```
$ createdb ogsa
$ psql -d ogsa -f $GLOBUS_LOCATION/etc/rft_schema_ogsa.sql
```

You should get an output similar to Example 3-23.

Example 3-23   Create database for RFT output

```
-bash-2.05b$ createdb ogsa
CREATE DATABASE
-bash-2.05b$ psql -d ogsa -f $GLOBUS_LOCATION/etc/rft_schema_ogsa.sql
CREATE SEQUENCE
CREATE SEQUENCE
psql:/usr/local/globus/etc/rft_schema_ogsa.sql:14: NOTICE:  CREATE TABLE / PRIMARY KEY will
create implicit index 'request_pkey' for table 'request'
CREATE TABLE
psql:/usr/local/globus/etc/rft_schema_ogsa.sql:37: NOTICE:  CREATE TABLE / PRIMARY KEY will
create implicit index 'transfer_pkey' for table 'transfer'
CREATE TABLE
```

```
CREATE TABLE
CREATE TABLE
-bash-2.05b$
```

3.6.5 Create Grid security files

Two security files need to be created from scratch using any text editor. These files essentially contain a listing of users that are to be given access to the Globus environment and are located in the /etc/grid-security directory. They are listed in Table 3-9.

Table 3-9   Security files

Security file name	Description
grid-mapfile	A flat file that maps the Grid users to the distinguished name of the user certificate. Each line represents an authorized user of the resource.
grim-port-type.xml	An XML document that maps the user ID to the Grid service name

Any user that is allowed to request services needs to have the information placed in both of these files. For our lab environment, we will include information regarding the itso user in these files.

As root, use your preferred editor to create the grid-mapfile (Example 3-24).

Example 3-24   Example of /etc/grid-security/grid-mapfile

```
"/O=ITSO/OU=XINGU/CN=itso" itso
"/O=ITSO/OU=XINGU/CN=itso" user1
"/O=ITSO/OU=XINGU/CN=itso" user2
"/O=ITSO/OU=XINGU/CN=itso" user3
```

Note: The contents of grid-mapfile comes from the user certificate. To verify the accuracy of the information, you may want to take a look at the certificate itself, which may be opened in any text editor or retrieved through execution of the following command:

```
# grid-cert-info -s -f /home/itso/.globus/usercert.pem
```

This information is also known as the distinguished name.

As root, use your preferred editor to create the grim-port-type.xml (Example 3-25).

Example 3-25   Example of grim-port-type.xml

```
<authorized_port_types>
<port_type username="itso">http://www.globus.org/namespaces/managed_job/managed_job/
ManagedJobPortType</port_type></authorized_port_types>
```

Attention: In Example 3-25, the entire <port_type username...> statement must be written on the same line. No blanks or carriage return characters are allowed.

4

Running examples

At this stage, all GT3 files have been installed and configured and it is time to test the installation by running a few examples. Before running the examples, we must start a hosting environment container. Here we use only a stand-alone container.

This chapter is split in two main sub-sections. The first sub-section is oriented to an end user, where you see the basic end-user commands on the command line interface (CLI). Although this document is not directly written for developers, the second main sub-section is oriented towards services operations, using the service browser Graphic User Interface (GUI). By using this service browser tool, the developer is able to see how services work, how they are related to each other, and to observe other components of the Grid services.

4.1 Start hosting environment container

The container listens to requests made by users and invokes another program or service as requests are made. For more information about the container, refer to 2.3.1, "Hosting environment, Web Service Engine, and Grid Service Container" on page 17.

A sample is included with the toolkit that can be used to verify that everything is working properly. To run the sample, start a container and request the sample service.

Start the container as user globus by running the **globus-start-container** command.

As the globus user, issue the following commands for this purpose:

```
$ cd $GLOBUS_LOCATION
$ globus-start-container
```

You should get an output similar to Example 4-1.

Example 4-1   Start container output

```
...(author omits lines)
http://192.168.0.241:8080/ogsa/services/base/index/IndexService
http://192.168.0.241:8080/ogsa/services/base/servicegroup/ServiceGroupFactory
http://192.168.0.241:8080/ogsa/services/base/servicegroup/ServiceGroupService
http://192.168.0.241:8080/ogsa/services/base/streaming/FileStreamFactoryFactoryService
http://192.168.0.241:8080/ogsa/services/base/multirft/MultiFileRFTFactoryService
http://192.168.0.241:8080/ogsa/services/gsi/AuthenticationService
http://192.168.0.241:8080/ogsa/services/gsi/SecureNotificationSubscriptionFactoryService
...(author omits lines)
```

Notes:

1. The container runs in the foreground, and you will not be returned back to a command prompt. To issue other commands, open another terminal window.

2. The list of URLs shown in Example 4-1 are the list of Grid services that are started along with the container.

3. If the machine that you installed GT3 on is slow, you can get a socket read time-out error. This is because the machine was not able to start the container within 60 seconds (the default timeout of a Grid service invocation). In order to change this behavior, refer to the Globus Toolkit 3.0 FAQ Web site at:

   http://www-unix.globus.org/toolkit/faq.html

4.2 Command line interface

This section shows how to run a base service provided by GT3 using the CLI. The base services are not part of the GT3 core, so they need to be installed separately:

► Job management
► Index services
► RFT services (multiRF)

4.2.1 Job management

This section presents a brief description on how to submit a job in the Grid. Prior to doing this task, it is necessary to log into the Grid by requesting a Grid proxy certificate to connect to the globus environment. This is accomplished through the **grid-proxy-init** command, as shown

in Example 4-2 and Example 4-3. The (Master) Managed Job Factory Service is the service that needs to be invoked by the `managed-job-globusrun` command to submit a job. The command is equivalent to the GT2 `globusrun` command. For more information about the `managed-job-globusrun` command, refer to the Globus Web site at http://www-unix.globus.org/developer/resource-management.html.

The `managed-job-globusrun` command shown in Example 4-4 submits a job in the x2 machine, as illustrated in Figure 4-1. The job is described in a Resource Specification Language (RSL) script embedded in a XML document.
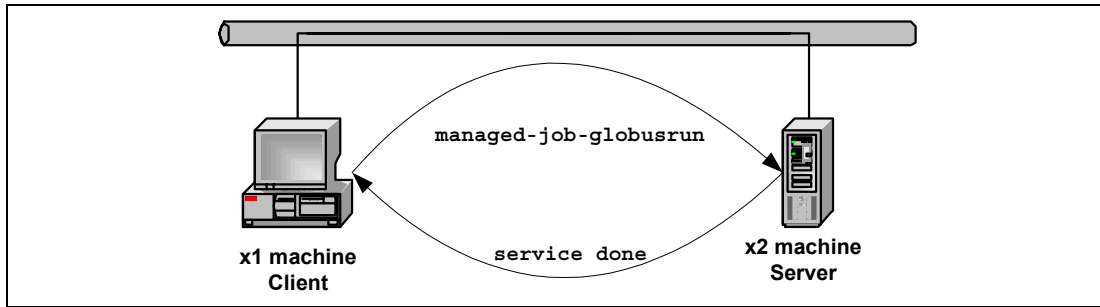
Figure 4-1   managed-job-globusrun

On x1, as the itso user, issue the commands in Example 4-2 to request a proxy and call the service:

Example 4-2   Request a proxy and call the service

```
$ grid-proxy-init
$ cd $GLOBUS_LOCATION
$ managed-job-globusrun -factory \
http://x2:8080/ogsa/services/base/gram/MasterForkManagedJobFactoryService -file \
$GLOBUS_LOCATION/etc/test.xml
```

You should get outputs similar to Example 4-3 and Example 4-4.

Example 4-3   Request a proxy output

```
[itso@x1 itso]$ grid-proxy-init
Your identity: /O=Grid/O=Globus/OU=itso-xingu.com/CN=itso
Enter GRID pass phrase for this identity: <passphrase>
Creating proxy ............................................ Done
Your proxy is valid until: Sat Jun 21 05:38:08 2003
```

Example 4-4   Call the service output

```
[itso@x1 itso] $ managed-job-globusrun -factory \
http://x2:8080/ogsa/services/base/gram/MasterForkManagedJobFactoryService -file
$GLOBUS_LOCATION/etc/test.xml
WAITING FOR JOB TO FINISH
========== Status Notification ==========
Job Status: Done
=========================================
DESTROYING SERVICE
SERVICE DESTROYED
[itso@x1 itso]
```

This service executes a test.xml (XML document) that embeds a RSL script. This test XML document is provided by GT3 and can be found in the $GLOBUS_LOCATION/etc directory.

Some key parts of the XML document can be pinpointed in order to better understand the behavior of the job. The job prints out string values into /tmp/sdtout and /tmp/stderr files. This operation can take a minute to return with the results shown above. This long turnaround time is because of the quantity of internal services and layers executed. For more information about these layers, refer to the Globus Web site at:

http://www-unix.globus.org/developer/gram-architecture.html

4.2.2 Index services

Index services provide a client-side command called `ogsi-find-service-data`. The command allows querying any service data element from any Grid service. The example illustrated in Figure 4-2 queries a SDE in a specific service running in the Grid. For a detailed description of the `ogsi-find-service-data` command, refer to the following Globus Web pages at:

http://www.globus.org/ogsa/releases/final/docs/infosvcs/indexsvc_overview.html
http://www.globus.org/ogsa/releases/final/docs/infosvcs/indexsvc_ug.html
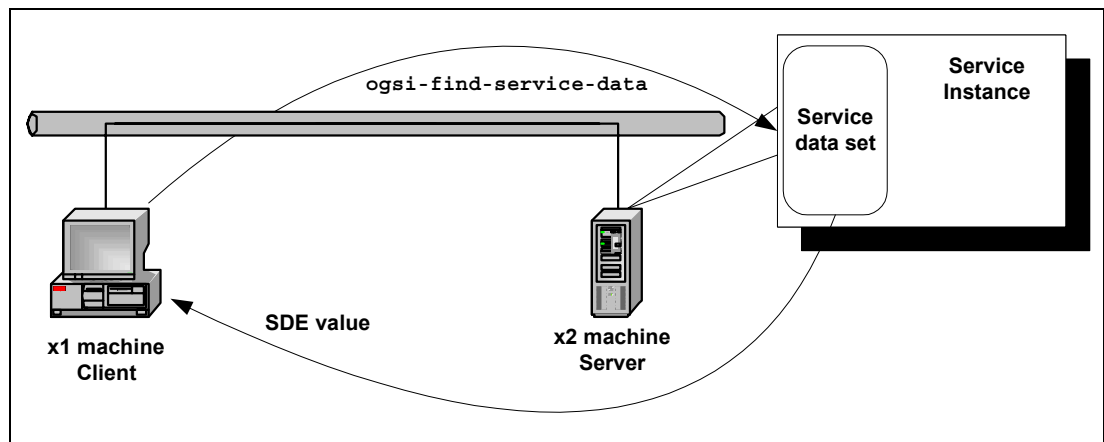
Figure 4-2   ogsi-find-service-data

4.2.3 RFT services

RFT provides the interface for reliable file transfers on Grid servers. A RFT client side Java based program is provided (RFTClient). For the server side, Data Management utilizes the same GridFTP daemon provided in GT2. For more information on Data Management and RFT, refer to the following Globus Web pages at:

http://www-unix.globus.org/developer/data-management.html
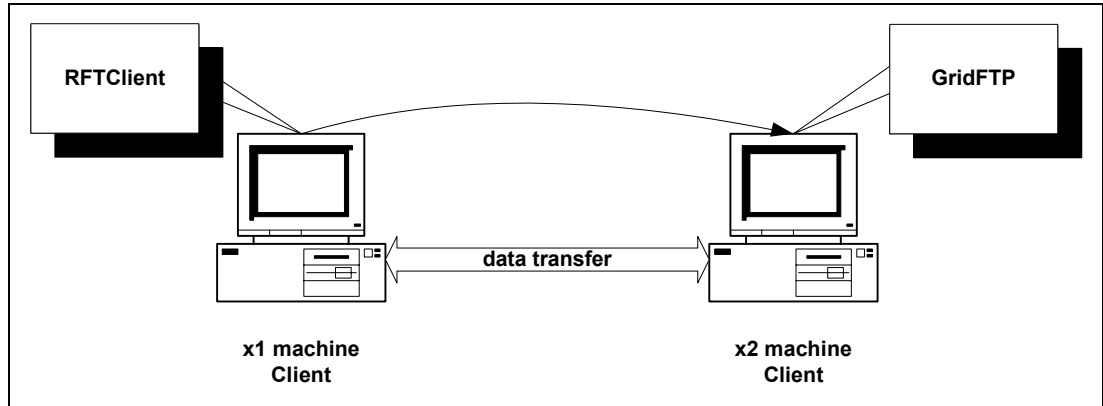http://www-unix.globus.org/toolkit/reliable_transfer.html

Figure 4-3   An example of RFTClient

4.3 Running the service browser

This section is oriented towards services operations using the service browser tool. Developers are able to see how the services and how they are relate to other components of the Grid services.

Programming tutorials and APIs descriptions are covered in the Java Programmer's Guide at http://www.globus.org.

Also, a tutorial hosted at http://www.casa-sotomayor.net/gt3-tutorial/ contains a good description of the programmatic aspects of GT3.

The examples walk through two samples distributed by GT3: Basic Counter Sample and Guide Samples. We will see how you can invoke the samples through the service browser.

► Basic Counter Sample

  This sample is a simple service that allows the user to perform add and subtract operations and get the results. There are two ways to run this sample: as a command line or using the service browser.

► Guide Samples

  This sample provides more elaborated versions of the basic counter service, such as adding service data and notification mechanism.

4.3.1 Running the Basic Counter Sample locally on x1

Having installed GT3 and MMJFS, we will now have a look on how you can use the service browser and the command line clients to run the Basic Counter Sample.

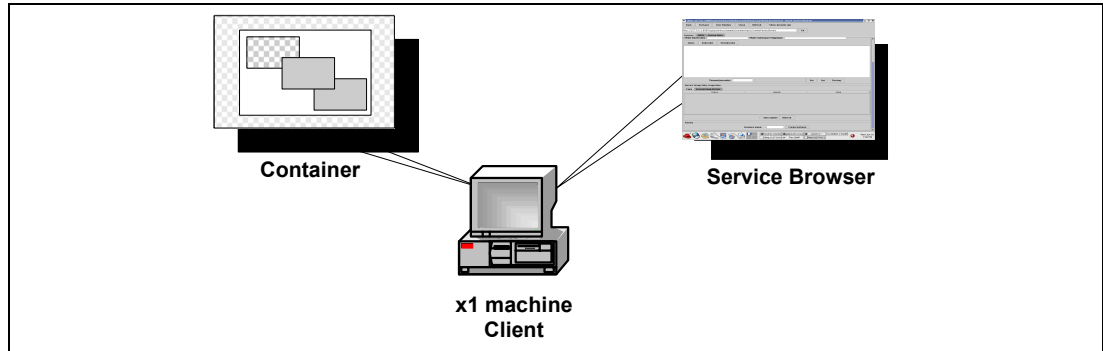As illustrated in Figure 4-4 on page 54, the examples run on the x1 machine.

Figure 4-4   Container and service browser

This is the list of activities that will be performed in this section:

- ► Start default container
- ► Start the service browser
- ► Create CounterServiceFactoy using the Create counter service instance with the service browser
- ► Use the counter service using the service browser
- ► Create a new counter service instance with the command line
- ► Use the second counter service instance with the command line
- ► Verify existence of both instances using the service browser
- ► Destroy both instances with the command line

Starting the default container

You may open a new window as globus (the user who installed the toolkit) and run `globus-start-container` from the globus-install directory ($GLOBUS_LOCATION).

On x1, as the globus user, issue the following commands to start the container:

```
$ cd $GLOBUS_LOCATION
$ globus-start-container | tee /tmp/servelog &
```

> **Note:** We are redirecting the output to a log file. This file would be of immense use in finding out what is going on and in getting stack traces (in case of errors).
>
> The `tee` command is not necessary, but it is recommended to preserve the output of the command.

Starting the service browser

Because of the quantity of data included in the service browser, make sure the resolution is set to display the necessary information in the window. In our lab a display resolution of 1024 by 768 was used. Also, you may need to run `xhost` to allow the use of the GUI, as described below.

As root, run `xhost`:

```
# xhost +
```

In a separate window, log in as the itso user and start the service browser.

As the itso user, issue the following commands for this purpose:

```
$ cd $GLOBUS_LOCATION
$ globus-service-browser | tee /tmp/clientlog &
```

At the end of this step, a Java-based GUI would have appeared on screen. That is the service browser tool.

Using the service browser to run the basic counter service

The idea of the service browser, presented in Figure 4-5, is similar to a Web browser, where you can specify the URL for the required Web location and allow navigating on it as well. For a Grid service, the URL is replaced by a URI (Uniform Resource Identifier).
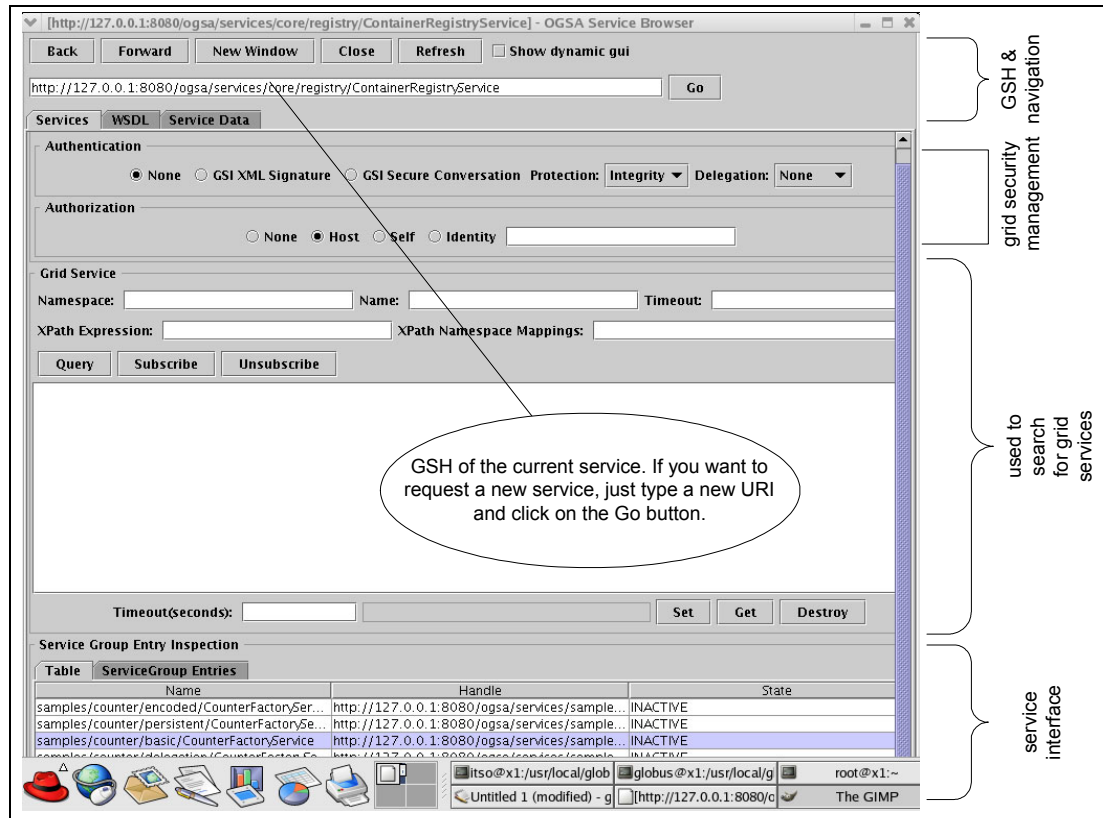
Figure 4-5   The service browser

Note: For the specific example presented Figure 4-5, the GSH points to the ContainerRegistryService, and therefore the service interface reflects the list of the container services.

Also, note that if you select another GSH, the service interface part of the window will change to reflect the proper interface for the selected service.

Step 1: Invoking the counter factory service

You may see the Service Group Entry Inspection panel in the bottom portion of the service browser. All services that are deployed in the container are listed there. You need to locate the CounterFactoryService in the Table tab, as shown in Figure 4-6 on page 56.
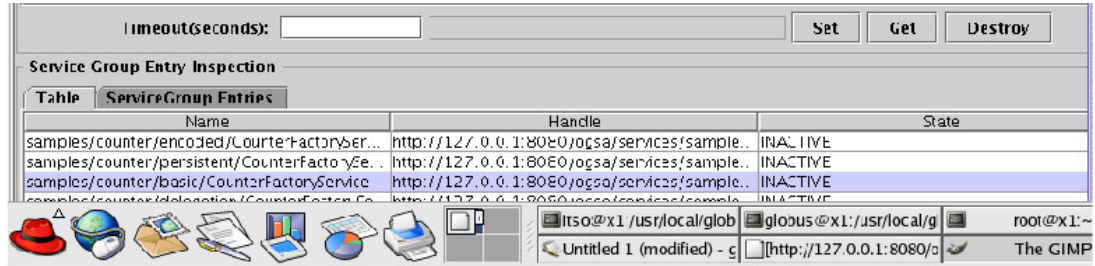
Figure 4-6   Locating the counter factory service

Now, double-click on the highlighted CounterFactoryService to invoke it. The service browser can take a long time to come up, so please be patient. The window will show CounterFactoryService in the URI field near the top of the screen, as shown in the Figure 4-7 on page 57.

Note: If you get the error message `Failed to obtain WSDL: null`, verify the URI field and click on the **Go** button located near the top of the screen.

Important: The idea of Factory is widely used throughout the Grid services. For more information about the concept of Grid service factories, refer to "Factory" on page 15.

Step 2: Creating a service instance

Near the bottom of the screen, you find a text box for entering an instance name and a Create Instance button. You may enter an instance name (for example, C1) and click this button to create a service instance.

Tip: You may leave the Instance name text box blank and click on the **Create Instance** button. In this case, the letter's hash (followed by a number) will be generated as your instance name. The instance name will form the last part of the URI, which is called the Grid Service Handle (GSH)).

After you press the **Create Instance** button, a new window will show the client interface for the newly C1 created counter factory service instance (c1), as shown in Figure 4-7 on page 57.
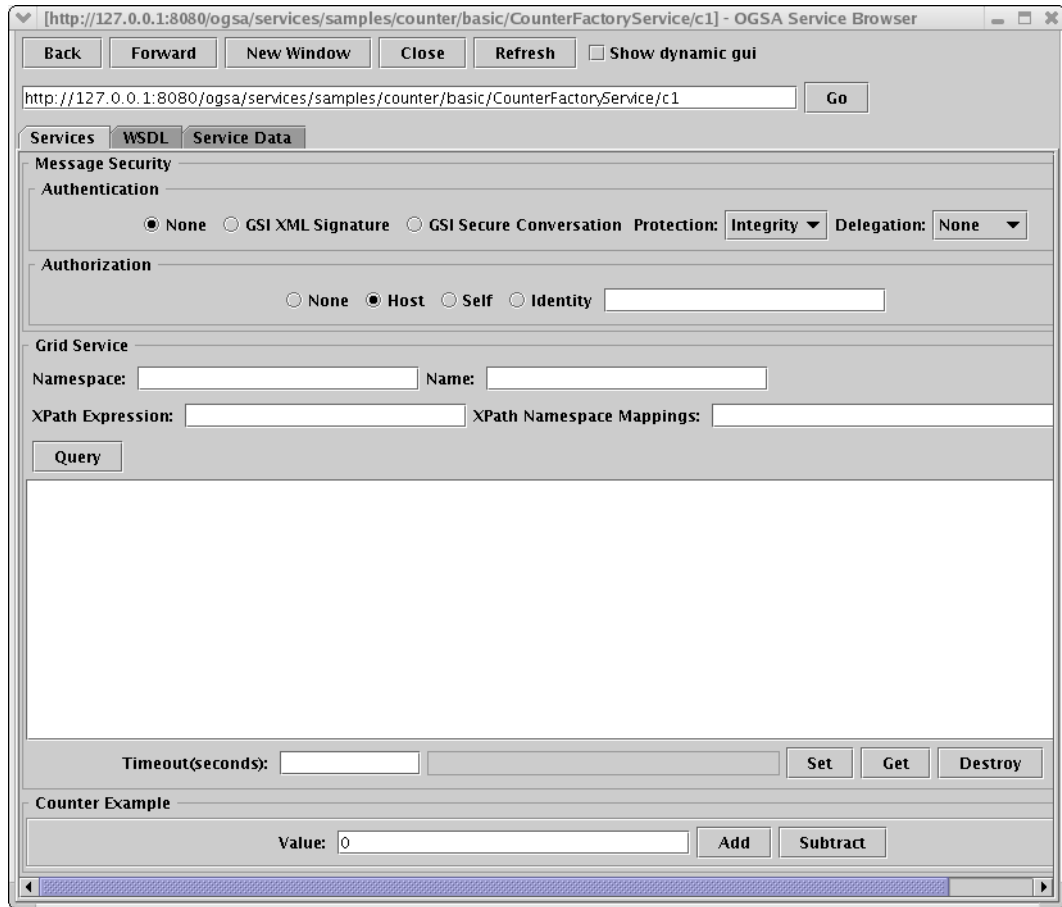
Figure 4-7   Counter factory service instance

You may notice than an URI appears in the top portion of the screen in Figure 4-7. This is the GSH of the counter factory service instance. Note that the instance name c1 has been added at the end of GSH.

Step 3: Using the counter service

Now we will perform operations on the counter factory service client. Near the bottom portion of the screen, under the Counter Example panel, you will find a text box for entering values and two buttons:

1. Add
2. Subtract

The buttons represents the available operations you can perform on the Counter Value. Note that the initial the value is 0. Let us increase the value of the counter by entering 10 in the text box provided for the value and pressing the **Add** button. Note that the value is 10 now. Now we can subtract 5 from the counter by entering 5 in the text box and clicking on the **Subtract** button.

You may observe that the final value at the end of the operation is 5. In the next section, let us see how you can perform these steps using command line clients.

Using the command line to run the basic counter service

GT3 is bundled with the ogsa-samples.jar file, which also contains Java programs that are basically command line clients. Java classes are available for creating and destroying

counter service instances. Let us now see how to create a second counter service instance and perform the operations using the command line.

Step 1: Creating a new service instance

In order to create a new instance, you need to use a Java program named CreateService, and pass the GSH of the factory service and the instance name as arguments.

On x1, as the itso user, issue the following commands to create the service instance called c2:

```
$ cd $GLOBUS_LOCATION
$ . setenv.sh
$ java org.globus.ogsa.client.CreateService \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService c2
```

> **Note:** The java org.globus.ogsa.client.CreateService call can be substituted with the `ogsi-create-service` command.

> **Tip:** In general, **". setenv.sh"** is needed when executing java programs similar to the one shown in Example 4-4 on page 51, but it is not needed for the command line tools **globus-start-container** and **globus-stop-container**.

> **Note:** In Example 4-5, the Termination Time by default is infinity. This can be changed by passing it as an option. The XML lines after `Create Service with Reference:` are the Grid Service Reference (GSR). This is nothing but the WSDL document associated with the Sample Counter. The last portion of the GSR shows an URI value for the soap:address location. This is the Grid Service Handle (GSH) through which you can locate the service instance named c1. For a more detailed explanation about GSR and GSH, refer to 2.2.1, "Naming" on page 13.

Example 4-5   Creating the service instance called c2 output

```
[itso@x1 globus]$ java org.globus.ogsa.client.CreateService
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService c2
Termination Time: infinity
Created service with reference:
<definitions name="Counter"
targetNamespace="http://ogsa.globus.org/samples/counter/service"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:counterbinding="http://ogsa.globus.org/samples/counter/bindings"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><import
location="http://192.168.0.241:8080/schema/samples/counter/counter_bindings.wsdl"
namespace="http://ogsa.globus.org/samples/counter/bindings"/><service name="CounterService"
xmlns:gsdl="http://ogsa.globus.org/"><gsdl:instanceOf
handle="http://192.168.0.241:8080/ogsa/services/samples/counter/basic/CounterFactoryService
/c2" xmlns=""/><gsdl:instanceOf handle="http://192.168.0.241:8080/ogsa/services/instance"
xmlns=""/><port binding="counterbinding:CounterSOAPBinding"
name="CounterPort"><soap:address
location="http://192.168.0.241:8080/ogsa/services/samples/counter/basic/CounterFactoryServi
ce/c2"/></port></service></definitions>
[itso@x1 globus]$
```

Step 2: Using the service instance

Now we will perform operations on the counter service instance, as presented in "Step 3: Using the counter service" on page 57, but now in command line mode. The permitted operations are still the same:

1. Add
2. Subtract

On x1, as the itso user, issue the following commands to add 5, subtract 2, and get the results:

```
$ java org.globus.ogsa.impl.samples.counter.client.Add 5 \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
$ java org.globus.ogsa.impl.samples.counter.client.Subtract 2 \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
$ java org.globus.ogsa.impl.samples.counter.client.GetValue \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
```

You should get an output similar to Example 4-6.

Example 4-6   Using the counter service with the command line output

```
[itso@x1 globus]$ java org.globus.ogsa.impl.samples.counter.client.Add 5
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
Counter Value:5
[itso@x1 globus]$ java org.globus.ogsa.impl.samples.counter.client.Subtract 2
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
Counter Value:3
[itso@x1 globus]$ java org.globus.ogsa.impl.samples.counter.client.GetValue
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
Counter Value:3
```

There also are several other options that you might supply for security and authentication. However those options are not covered in this Redpaper. You may invoke the programs with the -help flag to get the usage and the summary of those options, as presented in Example 4-7.

Example 4-7   Help option for counter output

```
[itso@x1 globus]$ java org.globus.ogsa.impl.samples.counter.client.Add -help
Usage: Add [options] <value> <service handle>
Where options are:
 -gsiSecConv <type>
         Enables GSI Secure Conversation. <type> one of:
          'sig' - for XML Signature
          'enc' - for XML Encryption
 -gsiSecConvActor <actor>
         Sets actor name for GSI Secure Conversation
 -gsiXmlSig
         Enables GSI XML Signature
         (can be used together with -gssxml)
 -gsiXmlSigActor <actor>
         Sets actor name for GSI XML Signature
 -deleg <mode>
         Performs delegation. <mode> one of:
          'limited' - performs limited delegation
          'full'    - performs full delegation
 -auth <type>
         Performs authorization type. If <type> is:
          'host' - performs host authorization
          'self' - performs self authorization
```

```
        'none' - disables authorization
        Otherwise, identity authorization is performed
        with <type> identity.
-grimPolicyHandler <policyClass>
        Sets GRIM policy handler
```

Step 3: Destroying the service instance

You can destroy the previously created c1 and c2 counter service instances by using the DestroyService Java program. But before you do, you may want to see which service instances you have created. For that task, you can use the service browser, as illustrated in Figure 4-8.

Go back to the window with the service browser, and press the back button until you return to the ContainerRegistryService GSH. Click on the **Service Group Entry Inspection** tab and take a look at the Name and Handle columns. You may need to expand the width of those columns. You will find a BasicCounterFactory followed by two Basic Counter instances that have GSH distinguished by the instance names (c1 and c2), as shown in Figure 4-8.
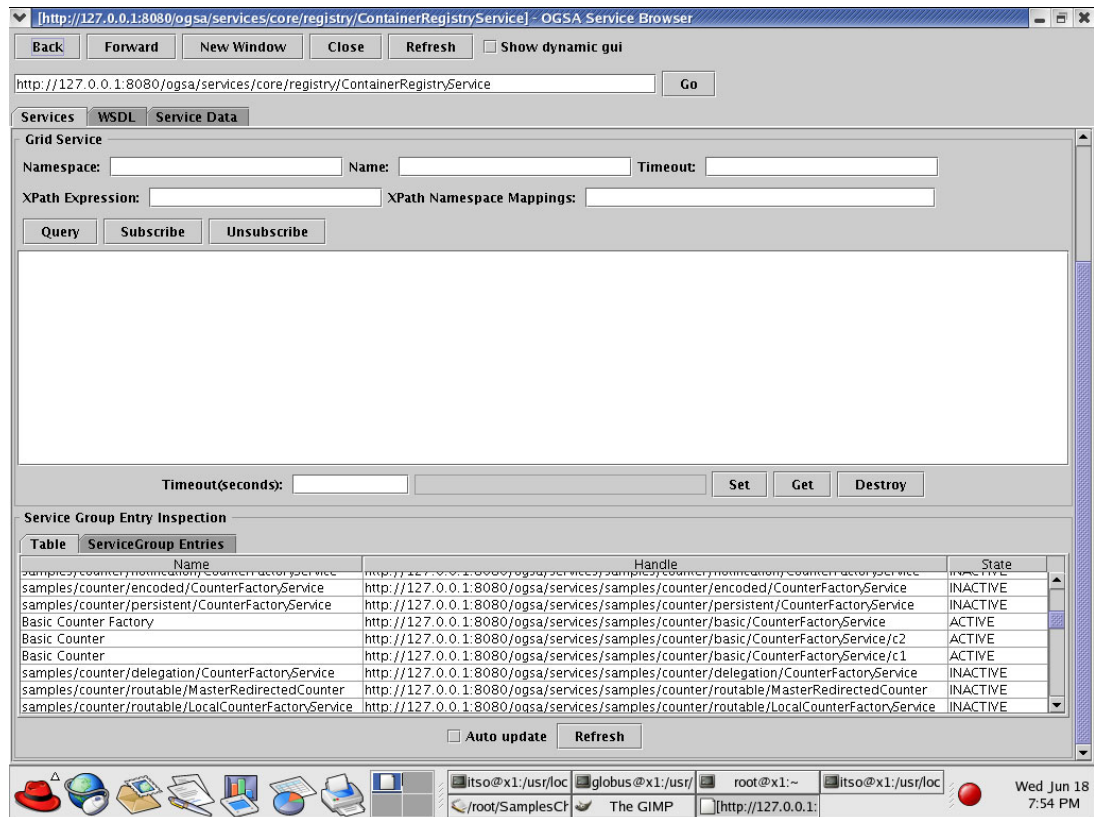
Figure 4-8   Verifying the counter service instances with the service browser

Now on x1, as the itso user, issue the following commands to destroy both services instances:

```
$ java org.globus.ogsa.client.DestroyService \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c1
$ java org.globus.ogsa.client.DestroyService \
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
```

You should get an output similar to Example 4-8 on page 61.

Example 4-8   Destroying c2 service instance output

```
[itso@x1 globus]$ java org.globus.ogsa.client.DestroyService
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
Destroyed service:
http://localhost:8080/ogsa/services/samples/counter/basic/CounterFactoryService/c2
```

> **Note:** The java org.globus.ogsa.client.DestroyService call can be substituted with the `ogsi-remove-service` command.

4.3.2 Running Basic Counter Sample remotely on the x2 machine

Having seen the execution of the Counter Sample where both the service and the client are on the same machine, we are now going to see how to execute the same counter service on a remote machine. The machine x1 will be the client, where the GUI runs, and the machine x2 will be the server, where the container runs, as illustrated in Figure 4-9.

.
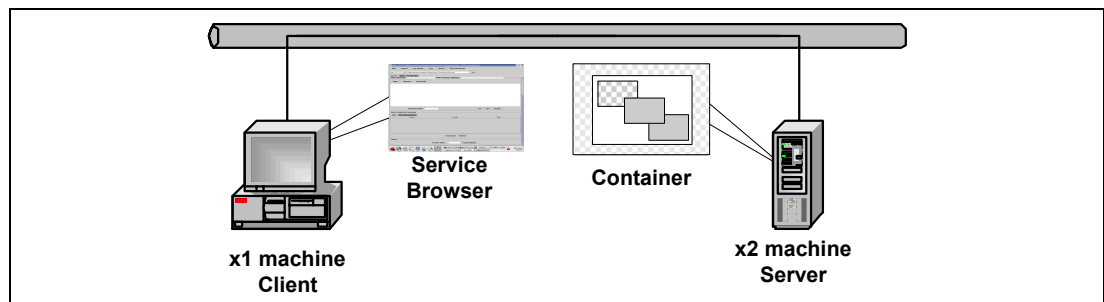
Figure 4-9   Service browser and container

This is the list of activities that will be performed in this section:

- ► Stop container in the x1 and x2 machines
- ► Start container on x2 only
- ► Start the service browser on x1
- ► Create the counter service instance on x2 from x1 using the command line
- ► Perform the add operation on x2 using the service browser on x1
- ► Perform the add operation on x2 with a command in x1

Stopping the container

Before executing the remote test sample, ensure that you stop the service browser and container on both x1 and x2.

You now need to stop the containers running on both x1 and x2. So, from the $GLOBUS_LOCATION directory, run the `globus-stop-container` command. Pressing Ctrl+C or running the `kill` command to stop the container may not gracefully shut down the container.

> **Note:** At the time of the writing of this Redpaper, we had some problems trying to stop the container. The following message was received:
>
> ```
> Error: Defective credential detected [Root error message: Proxy file (/tmp/x509up_u500)
> not found.]
> ```
>
> In order to work around with this problem, we set the X509_USER_PROXY environment variable to the proper certificate file name before stopping the container, because the correct credential in not located in the default location.
>
> You can find the certificate file name in the server-config.wsdd configuration file located in the $GLOBUS_LOCATION directory:
>
> ```
> $ grep -i containerProxy server-config.wsdd
> ```

On x1 and x2, as the globus user, issue the following command to stop the container:

```
$ cd $GLOBUS_LOCATION
$ grep -i containerProxy server-config.wsdd
$ export X509_USER_PROXY=/tmp/<name of the certificate name>
$ globus-stop-container -secure soft
```

Start the service browser on x1 and the container on x2

Start the container on the remote machine (x2) and the service browser on the local machine (x1). You can create the counter service instance and perform operations.

As the globus user, issue the following commands to start the container on x2:

```
$ cd $GLOBUS_LOCATION
$ globus-start-container | tee /tmp/serverlog
```

Now, start the service browser on x1 to show the services in x2. This is done by passing the service.port and service.host as parameters.

> **Note:** The container will not run on x1, since x1 for this present test is playing only the client role. It is used only as a client.

On x1, as the globus user, issue the following commands to start the service browser:

```
$ globus-service-browser http://x2 | tee /tmp/clientlog &
```

Now the service browser will appear, showing the list of available services on x2.

Creating a counter service instance through the command line

In order to create a counter service instance on x2, you need to run the CreateService Java program on the x1 machine.

On x1, as the itso user, issue the following command to create the counter service instance on x2, called from x1:

```
$ cd $GLOBUS_LOCATION
$ . setenv.sh
$ java org.globus.ogsa.client.CreateService \
http://x2:8080/ogsa/services/samples/counter/basic/CounterFactoryService fromx1
```

> **Note:** The java org.globus.ogsa.client.CreateService call can be substituted with the **ogsi-create-service** command line.

You should get an output similar to Example 4-9.

Example 4-9   Creating an instance on x2 from the x1 machine output

```
[itso@x1 globus]$ java org.globus.ogsa.client.CreateService
http://x2.itso-xingu.com:8080/ogsa/services/samples/counter/basic/CounterFactoryService
fromx1
Termination Time: infinity
Created service with reference:
<definitions name="Counter"
targetNamespace="http://ogsa.globus.org/samples/counter/service"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:counterbinding="http
://ogsa.globus.org/samples/counter/bindings"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><import
location="http://192.168.0.242:8080/schema/samples/count
er/counter_bindings.wsdl"
namespace="http://ogsa.globus.org/samples/counter/bindings"/><service name="CounterService"
xmlns:gsdl="http://ogsa.globus.org/"><gsdl
:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/samples/counter/basic/CounterFactoryService
/fromx1" xmlns=""/><gsdl:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/instance" xmlns=""/><port
binding="counterbinding:CounterSOAPBinding"
name="CounterPort"><soap:addresslocation="http://192.168.0.242
:8080/ogsa/services/samples/counter/basic/CounterFactoryService/fromx1"/></port>
</service></definitions>
```

Performing an add operation

Let us now perform an add operation on the counter instance called from x1.

On x1, as the itso user, issue the following command to perform add operation:

```
# java org.globus.ogsa.impl.samples.counter.client.Add 5 \
http://x2:8080/ogsa/services/samples/counter/basic/CounterFactoryService/fromx1
```

Note: The counter service instance will run on the x2 machine, but it was started from the command line on the x1 machine.

You should get an output similar to Example 4-10.

Example 4-10   Using the counter remotely output

```
[itso@x1 globus]$ java org.globus.ogsa.impl.samples.counter.client.Add 5
http://x2:8080/ogsa/services/samples/counter/basic/CounterFactoryService/fromx1
Counter Value:5
```

You may want to double check the counter using the service browser.

4.3.3 Guide Samples

In order to run the Guide Samples, you need to deploy them on the default container. They provide a more elaborate version of the basic counter service, such as adding service data and a notification mechanism. For more details about these two topics, refer to 2.2.2, "Service data" on page 13 and 2.2.3, "Notifications" on page 14.

The counter service samples used until now are the simple ones and are provided ready-to-use by default. But the Guide Samples need to be deployed in the container before you can use them.

Deploying the Guide Samples

This is the list of activities that will be performed in this section:

► Deploy the Guide Sample on the container

To deploy the Guide Samples in the default container on x2, the **ant** command must be used with the target deployGuide from $GLOBUS_LOCATION.

► On x2, as the globus user, issue the following command to deploy the guides:

```
$ cd $GLOBUS_LOCATION
$ ant deployGuide
```

You should get an output similar to Example 4-11.

Example 4-11   Deploying the Guide Samples output

```
[root@x1 root]# su - globus
[globus@x1 globus]$ cd $GLOBUS_LOCATION
[globus@x1 globus]$ ant deployGuide
Buildfile: build.xml

guide:

rebuildGarTest:

setenv:
    [mkdir] Created dir: /usr/local/globus/guide/build/classes
    [mkdir] Created dir: /usr/local/globus/guide/build/lib
    [mkdir] Created dir: /usr/local/globus/guide/build/schema
    [mkdir] Created dir: /usr/local/globus/guide/build/stubs
     [copy] Copying 198 files to /usr/local/globus/guide/build/schema

mappingAvailable:

mergePackageMapping:
     [echo] merging NStoPkg.properties.guide
     [copy] Copying 1 file to /usr/local/globus/guide/build/schema

generateWSDL:


...(author omits lines)


deployGarFiles:
    [mkdir] Created dir: /usr/local/globus/build/gar/schema
    [mkdir] Created dir: /usr/local/globus/build/gar/etc
    [mkdir] Created dir: /usr/local/globus/build/gar/bin
    [mkdir] Created dir: /usr/local/globus/build/gar/docs
    [unjar] Expanding: /usr/local/globus/guide/build/lib/guide.gar into
/usr/local/globus/build/gar
     [copy] Copying 17 files to /usr/local/globus/schema
    [chmod] Skipping fileset for directory /usr/local/globus/bin. It is empty.
     [copy] Copying 3 files to /usr/local/globus/lib

testServerDeployAvailable:

deployServer:
     [echo] deploying server config...

testClientDeployAvailable:
```

```
deployClient:

generateUndeploy:

setGarID:

testPostDeployAvailable:

setAbsoluteGlobusLocation:

postDeploy:
    [delete] Deleting directory /usr/local/globus/build/gar

BUILD SUCCESSFUL
Total time: 1 minute 59 seconds
```

Deploying new services in the container means installing new Java classes in the container. So, you now need to restart the container using these new services. So, as the itso user, from the $GLOBUS_LOCATION directory, run the **globus-stop-container** command. Pressing Ctrl+C or running the **kill** command to stop the container may not gracefully shut down the container.

On x1 and x2, as the globus user, issue the following command to stop the container:

```
$ cd $GLOBUS_LOCATION
$ export X509_USER_PROXY=/tmp/x509cp_globus_grim
$ globus-stop-container -secure soft
```

Now you may restart the container using the **globus-start-container** command. It will load the Guide Sample Services while starting, as shown in Example 4-12.

On x2, as the itso user, issue the following command to start the container:

```
$ cd $GLOBUS_LOCATION
$ globus-start-container | tee /tmp/serverlog &
```

You should get an output similar to Example 4-12.

Example 4-12   A few lines from the /tmp/serverlog output

```
...(author omits lines)
http://192.168.0.242:8080/ogsa/services/base/index/IndexService
http://192.168.0.242:8080/ogsa/services/base/servicegroup/ServiceGroupFactory
http://192.168.0.242:8080/ogsa/services/base/servicegroup/ServiceGroupService
http://192.168.0.242:8080/ogsa/services/base/streaming/FileStreamFactoryFactoryService
http://192.168.0.242:8080/ogsa/services/base/multirft/MultiFileRFTFactoryService
http://192.168.0.242:8080/ogsa/services/gsi/AuthenticationService
http://192.168.0.242:8080/ogsa/services/gsi/SecureNotificationSubscriptionFactoryService
...(author omits lines)
```

Note that in Example 4-12 that the new deployed guide services GSH is presented in the list of available services in the container.

You can be sure that the Guide Samples have been deployed by issuing the service browser in x2.

On x2, as the globus user, issue the following commands to start the service browser:

```
$ cd $GLOBUS_LOCATION
$ globus-service-browser | tee /tmp/clientlog
```

> **Note:** Do not forget to source .setenv.sh to add the newly created guide.jar and guide-stubs.jar to the CLASSPATH.

You will observe in the service browser under the Service Group Entry Inspection panel's Table tab, several guide/counter/<name>Service(s) are available, as shown in Figure 4-10.
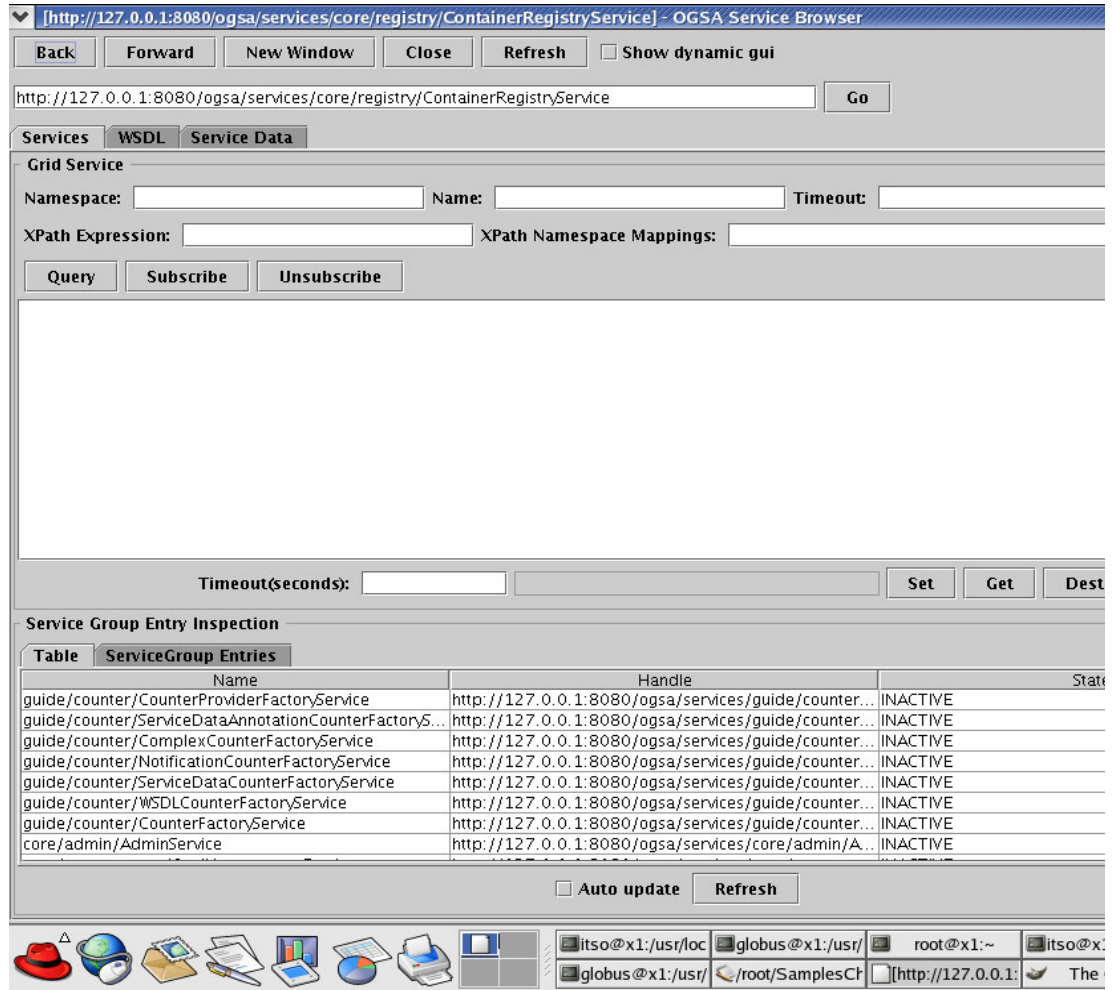

Figure 4-10   The service browser in x2 showing the Guide Samples services

In the next section, we will show you how to run the Guide's Counter Service.

Run the new ServiceDataCounter Sample

Unlike the Basic Counter we saw earlier, the Guide's Counter Sample has a single new client Java program known as CounterClient for performing all operations. In addition to the service browser and command lines, this Java program can also be used to call the Counter Samples services.

The Guide's Counter Sample is very similar to the Basic Counter Sample. The Guide Samples are intended to explain the programming concepts like service data and notifications, which you can implement in your services. In the following sections, we will look at the ServiceData and Notification services.

This is the list of activities that will be performed in this section:

- ► Create a service data counter instance on x2 from x1 with commands
- ► Query x2 service data from x1 with commands
- ► Perform an add operation on x2 from x1 with commands
- ► Query x2 service data from x1 with commands
- ► Check a service data value using the service browser from x1

Step 1: Create the service instance

In order to create the service instance, we can use the Java program CreateService. For this current example we need to create an instance of the ServiceDataCounterFactoryService service on the x2 remote machine.

On x1, as the itso user, issue the following commands to create the service instance on x2:

```
$ cd $GLOBUS_LOCATION
$ . setenv.sh
$ java org.globus.ogsa.client.CreateService \
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService c2
```

Note: The java org.globus.ogsa.client.CreateService call can be substituted with the **ogsi-create-service** command.

You should get an output similar to Example 4-13.

Example 4-13   Creating the ServiceDataCounter instance from the x2 output

```
[itso@x1 globus]$ java org.globus.ogsa.client.CreateService
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService c2
Termination Time: infinity
Created service with reference:
<definitions name="ComplexCounter"
targetNamespace="http://www.globus.org/namespaces/2003/04/guide/counter/service"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:complexcounterbinding="http://www.globus.org/namespaces/2003/04/guide/counter/binding
s" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><import
location="http://192.168.0.242:8080/schema/guide/Counter/counter_bindings.wsdl"
namespace="http://www.globus.org/namespaces/2003/04/guide/counter/bindings"/><service
name="ComplexCounterService" xmlns:gsdl="http://ogsa.globus.org/"><gsdl:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryServ
ice/c2" xmlns=""/><gsdl:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/instance" xmlns=""/><port
binding="complexcounterbinding:ComplexCounterSOAPBinding"
name="ComplexCounterPort"><soap:address
location="http://192.168.0.242:8080/ogsa/services/guide/counter/ServiceDataCounterFactorySe
rvice/c2"/></port></service></definitions>
```

Note: All subsequent steps can be done in the same window as the itso user.

Step 2: Querying the ServiceData

The original default counter service sample includes three operations: add, sub, and getvalue. Now, the deployed guides implement a new operation called state. This new operation returns the value of the service data associated to the service.

Now we are going to call, from x1, the state operation of ServiceDataCounter service deployed on the x2 machine.

On x1, as the itso user, issue the following commands to query x2:

```
$ java org.globus.ogsa.guide.impl.CounterClient \
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/c2 state
```

You should get an output similar to Example 4-14.

Example 4-14   Querying the service data output

```
[itso@x1 globus]$ java org.globus.ogsa.guide.impl.CounterClient
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/c2 state
Counter state:
    status: initialized
    value: 0
    timestamp: Fri Jun 20 12:14:51 CDT 2003
```

The CounterClient service was called to perform the state operation. Here the service data is called Counter State and comprises the current status, value and timestamp. Because the service has never been used before, the status is just set to initialized.

Step 3: Performing operations with the client program

On x1, as the itso user, use the Java program CounterClient to perform the add operation on the ServiceDataCounter instance c2 running on x2's container.

On x1, as the itso user, issue the following commands to perform the add operation on x2:

```
$ java org.globus.ogsa.guide.impl.CounterClient \
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/c2 add 4
```

You should get an output similar to Example 4-15.

Example 4-15   Using a client to perform operations output

```
[itso@x1 globus]$  java org.globus.ogsa.guide.impl.CounterClient
http://x2.itso-xingu.com:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/
c2 add 4
Counter add: 4
```

Step 4: Querying the ServiceData

As in "Step 2: Querying the ServiceData" on page 67, run the CounterClient to perform a state operation on the ServiceDataCounter instance c2 running on x2's container.

On x1, as the itso user, issue the following commands to query x2:

```
$ java org.globus.ogsa.guide.impl.CounterClient \
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/c2 state
```

You should get an output similar to Example 4-16.

Example 4-16   Querying the service data of instance c2 running on x2's container output

```
[itso@x1 globus]$ java org.globus.ogsa.guide.impl.CounterClient
http://x2:8080/ogsa/services/guide/counter/ServiceDataCounterFactoryService/c2 state
Counter state:
    status: add done
    value: 4
    timestamp: Fri Jun 20 12:17:13 CDT 2003
```

Note that the new value of the elements of service data has changed, that is, the value is 4 and the status is add done.

Important: The idea of the service data queries is widely used throughout the Grid services. The above examples demonstrate that when a Grid service with service data is running, any user, at any moment, can query the ServiceData service to obtain the status or current value of a data associated to a service. For more information about service data, refer to 2.2.2, "Service data" on page 13.

Step 5: Querying the ServiceData with the service browser

Now we are going to use the service browser to query the service data. Go to your service browser in the x2 machine and select the guide service data counter c2 instance and double-click on it. Now the service browser displays the data associated to the c2 instance of the service data counter instance. Click on the **Service Data** tab, just under the GSH area, as illustrated in Figure 4-11.

Tip: On the service browser frame, you may use the Back and Forward buttons to navigate between the different Factory services and service instances you had opened earlier.

Figure 4-11   CounterState service data

In the last line of the Figure 4-11, you can see the CounterState. By extending the XML elements, you can retrieve the status, the value, and time-stamp information.

NotificationCounter Sample

Notifications allow clients to be notified of changes that occur in a Grid services. They deliver interesting messages from a notification source to a notification sink. For an explanation of notification, refer to 2.2.3, "Notifications" on page 14.

This is the list of activities that will be performed in this section:

- ► Start the notification counter instance on x2 from x1 with commands
- ► Start a listener from x1 for the x2 notification counter service with commands
- ► Perform add operation on x2 from x1 with commands
- ► Check the events received on the x1 listener

Let us now take a look at the notifications by using the NotificationCounter Sample.

Step 1: Create a service instance

Just like the previous example, we use CreateService Java program to instantiate the NotificationCounterFactoryService on the remote x2 machine. This version of the Counter Sample sends a notification to the subscriber every time an add or subtract operation is done.

On x1, as the itso user, issue the following commands to create the service instance on x2:

```
$ cd $GLOBUS_LOCATION
$ . setenv.sh
$ java org.globus.ogsa.client.CreateService \
http://x2:8080/ogsa/services/guide/counter/NotificationCounterFactoryService n1
```

> **Note:** The java org.globus.ogsa.client.CreateService call can be substituted with the
> **ogsi-create-service** command.

You should get an output similar to Example 4-17.

Example 4-17   Creating the NotificationCounter instance on x2 output

```
[itso@x1 globus]$ java org.globus.ogsa.client.CreateService
http://x2:8080/ogsa/services/guide/counter/NotificationCounterFactoryService  n1
Termination Time: infinity
Created service with reference:
<definitions name="NotificationCounter"
targetNamespace="http://www.globus.org/namespaces/2003/04/guide/counter/notification/servic
e" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:notificationcounterbinding="http://www.globus.org/namespaces/2003/04/guide/counter/no
tification/bindings" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><import
location="http://192.168.0.242:8080/schema/guide/Counter/notification_counter_bindings.wsdl
"
namespace="http://www.globus.org/namespaces/2003/04/guide/counter/notification/bindings"/><
service name="NotificationCounterService"
xmlns:gsdl="http://ogsa.globus.org/"><gsdl:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/guide/counter/NotificationCounterFactorySer
vice/n1" xmlns=""/><gsdl:instanceOf
handle="http://192.168.0.242:8080/ogsa/services/instance" xmlns=""/><port
binding="notificationcounterbinding:NotificationCounterSOAPBinding"
name="NotificationCounterPort"><soap:address
location="http://192.168.0.242:8080/ogsa/services/guide/counter/NotificationCounterFactoryS
ervice/n1"/></port></service></definitions>
```

Step 2: Starting the StatusListener

Now we need a service listening to the notification that will be sent. This service is called a subscriber, and is executed on local x1 machine by using the StatusListener Java program.

> **Note:** Since the following operation does not return the prompt immediately, open a new
> xterm window for that. Also, this will help you to easily see all notifications.

On x1, as the itso user, issue the following commands to start the listener:

```
$ cd $GLOBUS_LOCATION
$ . setenv.sh
$ java org.globus.ogsa.guide.impl.StatusListener \
http://x2:8080/ogsa/services/guide/counter/NotificationCounterFactoryService/n1
```

This StatusListener Java program does not return the prompt immediately, but instead waits for the occurrence of the upcoming notification. As mentioned before, it is just a subscriber.

The
`http://x2:8080/ogsa/services/guide/counter/NotificationCounterFactoryService/n1`
argument is the GSH of the service from which we will receive the notification.

Step 3: Performing operations with the command line client

Now, in another window, execute an operation on the remote x2 machine.

On x1, as the itso user, issue the following commands to perform an add operation on the remote x2 machine:

```
$ java org.globus.ogsa.guide.impl.CounterClient \
http://x2:8080/ogsa/services/guide/counter/NotificationCounterFactoryService/n1 add 5
```

You should get an output similar to Example 4-18.

Example 4-18   Using CounterClient Java program output

```
[globus@x1 globus]$ java org.globus.ogsa.guide.impl.CounterClient
http://x2.itso-xingu.com:8080/ogsa/services/guide/counter/NotificationCounterFactoryService
/n1 add 5
```

Step 4: Observing the StatusListener

Toggle to the xterm window running the StatusListener initiated in "Step 2: Starting the StatusListener" on page 70. You will see the add done notification, as shown in Example 4-19.

Example 4-19   StatusListener output

```
[itso@x1 itso]$ java org.globus.ogsa.guide.impl.StatusListener
http://x2.itso-xingu.com:8080/ogsa/services/guide/counter/NotificationCounterFactoryService
/n1
Counter status: add done
```

A
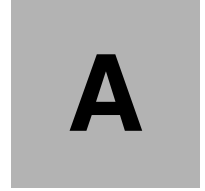
Infrastructure server setup

This appendix presents the steps necessary to build the infrastructure server that will serve as a NFS install image repository, a Network Time Protocol (NTP) server, and a simple certificate authority (CA).

73

m0 environment

In order to establish a Grid environment, we recommend that you first set up an installation server, which is a repository where you can keep the latest versions of the necessary software to install, configure, and maintain your Grid environment as well as a Network Time Protocol (NTP) server. In addition, we suggest that you create your own certificate authority (CA).

> **Attention:** In a production environment, a pre-established and reputable CA should be used. The reason we created our own simple CA is to build an easy and simple certificate signing process and to introduce the certificate authority topic to the readers, in case they are not familiar with it.

The host name we set for this machine is m0.itso-maya.com, also known as m0, and contains the following software items, as described in Figure A-1.

► Red Hat Version 9.0
► NFS installation repository
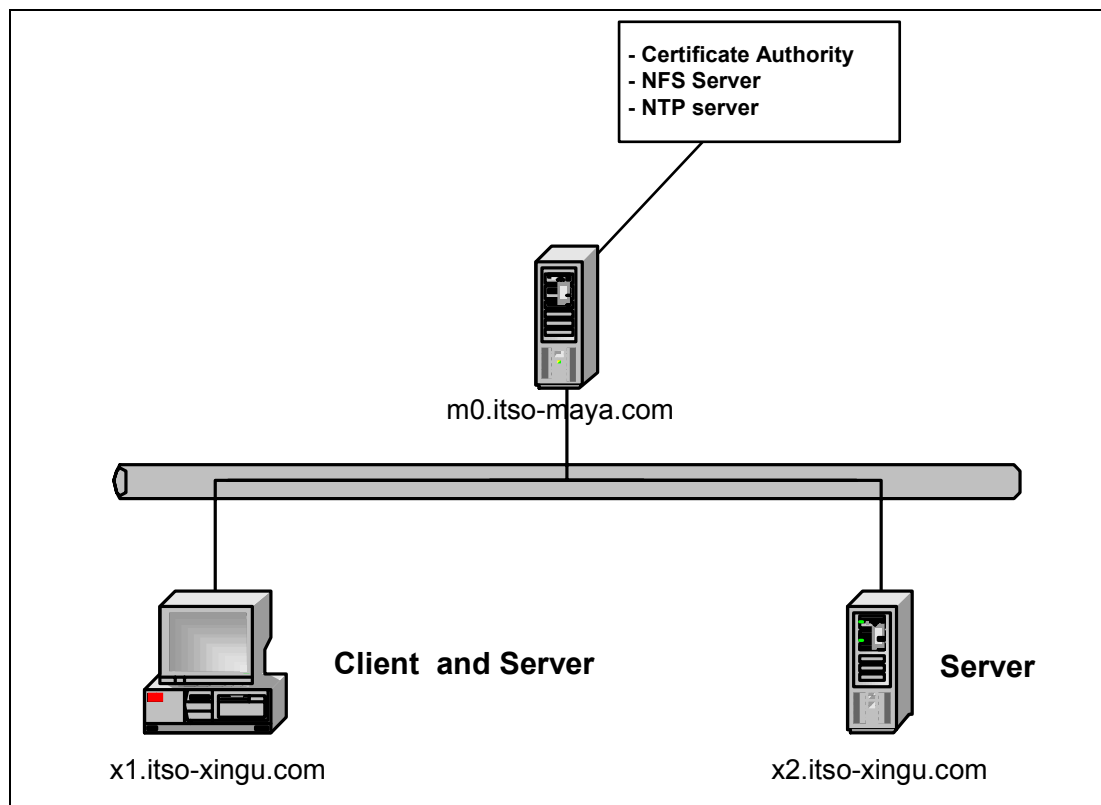► Network Time Protocol (NTP)
► Certificate authority (CA)

Figure A-1   m0 machine

Hardware requirements

The hardware needed to prepare the necessary environment to build the m0 machine is:

► A Pentium 1 GHz processor or compatible
► 512 MB of memory

► A 20 GB hard disk

Software installed

The software needed to build the m0 machine is Red Hat Linux (workstation package).

Naming and addressing planning

Table A-1 through Table A-6 on page 76 describe the naming convention and addressing planning used in our lab.

Table A-1   ID and passwords

user ID	group ID	password
root	root	<password>

Table A-2   Host names and IP addressing

Host name	IP	Description
x1.itso-xingu.com	192.168.0.241	Globus server and Client machine
x2.itso-xingu.com	192.168.0.242	Globus server
m0.itso-maya.com	192.168.0.10	Infrastructure server

Table A-3   CA Distinguished Name

Tag	Description	Example
C	Country Name (2 letter code)	C=US
ST	State or Province Name (full name)	ST=Texas
L	Locality Name (for example, city)	L=Austin
O	Organization Name (for example, company)	O=ITSO
OU	Organizational Unit Name (for example, section)	OU=XINGU
CN	Common Name (for example, your name or your server's host name)	CN=m0
Email	E-mail address of the CA administrator	ca@m0

Table A-4   Certificate naming convention

Generic name	Role	Name in the CA
usercert_request.pem	non-signed user certificate	<user>usercert_request.pem
usercert_cert.pem	signed user certificate	<user>usercert_cert.pem
hostcert_request.pem	non-signed host certificate	<machine>hostcert_request.pem
hostcert_cert.pem	signed host certificate	<machine>hostcert_cert.pem

Table A-5   CA directory structure

Directory name	Role	Permission
/CA	Top level directory of the CA. This directory must be exported via NFS. Also, it contains the OpenSSL demoCA directory and the openssl.cnf configuration file.	drwxr-xr-x
/CA/IN	Used to store incoming certificate requests. The non-signed certificate must be copied to this directory before signing it.	drwxrw-rw-
/CA/OUT	Used to store signed certificates. The signed certificate is placed in this directory by the CA after signing it.	drwxr--r--
/CA/PROCESSED	Contains processed certificate requests.	drwxr-xr-x

Table A-6   Repository directory structure

Directory name	Role	Permission
/expgt3	Top level directory of the repository. This directory must be exported via NFS.	drwxr-xr-x
/expgt3/globus	globus repository.	drwxr-xr-x
/expgt3/sun-java	sun-java repository.	drwxr-xr-x
/expgt3/apache	apache repository.	drwxr-xr-x
/expgt3/junit	junit repository.	drwxr-xr-x
/expgt3/postgresql	postgresql repository.	drwxr-xr-x
/expgt3/camgr	camgr repository.	drwxr-xr-x

m0 installation

The following steps should be performed in the order presented in this chapter. There are dependencies among them that require performing them in this order. The major steps to set up the Grid environment include:

1. Install Red Hat Linux.
2. Configure Network Time Protocol.
3. Configure IDs and groups.
4. Configure /etc/hosts.
5. Populate the installation image repository.
6. Create Java SDK RPM and install it.
7. Install and configure the certificate authority.

Install Red Hat Linux

Install Red Hat Linux on m0, selecting **Workstation** and **No firewall**. Do not use DHCP. Instead, use a fixed network IP address with the corresponding host name, as presented in Table A-2 on page 75.

Configure Network Time Protocol (NTP)

NTP needs to be configured as the clocks on the systems throughout the Grid need to be synchronized. The security process creates proxy certificates that are valid for specific times. If the systems do not have their clocks synchronized, the users may not be able to use the Grid, as the proxy certificates may appear to be expired or not yet valid.

As root, issue the following command to check if NTP is installed:

```
# rpm -aq |grep ntp
```

You should get an output similar to Example A-1.

Example: A-1   Checking for NTP output

```
root@m0 root]# rpm -aq |grep ntp
ntp-4.1.2-0.rc1.2
```

Next, modify the /etc/ntp.conf file to properly configure the NTP server.

As root, use your preferable editor to modify the /etc/ntp.conf file, as follows:

```
server 127.127.1.0
driftfile /etc/ntp/drift
keys            /etc/ntp/keys
```

Attention: In a production environment, a pre-established and reputable NTP server should be used.

As root, issue the following commands to start the NTP daemon and change the configuration of the run-time level to start the ntpd daemon automatically:

```
# service ntpd restart
# chkconfig --level 345 ntpd on
```

Configure /etc/hosts

/etc/hosts needs to be configured to map the host names to the IP addresses of the machines in your environment. Example A-2 shows the /etc/hosts file used in our lab.

As root, use your preferable editor to modify the /etc/hosts, as shown in Example A-2.

Example: A-2   New /etc/hosts

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain    localhost
# infrastructure machine
192.168.0.10    m0.itso-maya.com         m0
# grid machines
192.168.0.241   x1.itso-xingu.com        x1
192.168.0.242   x2.itso-xingu.com        x2
```

Populate installation image repository

A Network File System (NFS) will be used to maintain an installation image repository consisting of all of the software necessary to install GT3. To begin, several directories will be created, as defined in Table A-6 on page 76, where all of the software will be downloaded.

As root, issue the following commands to create the directory structure:

```
# mkdir /expgt3
# cd /expgt3
# mkdir sun-java
# mkdir apache
# mkdir junit
# mkdir postgresql
# mkdir globus
# mkdir camgr
```

Next, download the necessary software to be used to the appropriate directory. For details, see Appendix C, "Software installed" on page 91.

As root, use your preferred Web browser or FTP client to download the software packages shown in Table A-7.

Table A-7   Software repository

Software	Repository directory	Software for
GPT Source Installation Package	/expgt3/globus	x1 and x2 (Grid machines)
GPT Linux Installation Package	/expgt3/globus	x1 and x2 (Grid machines)
Java SDK	/expgt3/sun-java	x1 and x2 (Grid machines) and m0 (infrastructure machine)
Apache Ant	/expgt3/apache	x1 and x2 (Grid machines)
Junit	/expgt3/junit	x1 and x2 (Grid machines)
PostgreSQL	/expgt3/postgresql	x1 and x2 (Grid machines)
CAMgr tool	/expgt3/camgr	m0 (infrastructure machine)

Configure Network File System (NFS)

The NFS server needs to be configured and started to provide (export) the directories for installation and management of the Grid machines, which are the /CA and /expgt3 directories.

As root, use your preferred editor to include the directories in the access control list file:

```
/CA     *(rw,no_root_squash)
/expgt3 *(ro,no_root_squash)
```

As root, issue the following commands to start the NFS daemon and change the configuration of the run-time level to start the nfsd automatically:

```
# service nfs start
# chkconfig --level 345 nfs on
```

Create Java SDK RPM

The Java Software Developer Kit (SDK) is an executable that needs to be run to create an RPM. The read/write/execute privileges first need to be changed to allow execution using the **chmod** command. After executing the binary, you will be presented with the Sun Microsystems License Agreement. Press the Spacebar to progress through the screens and type yes on the final screen to accept the agreement.

As root, issue the following commands to create the RPM:

```
# cd /expgt3/sun-java
# chmod +x j2sdk-1_4_1_03-linux-i586-rpm.bin
# ./j2sdk-1_4_1_03-linux-i586-rpm.bin
```

Note: Make sure you read and comply with the Binary Code License Agreement provided by Sun Microsystems, Inc.

You should get an output similar to Example A-3.

Example: A-3   Create Java SDK RPM output

```
root@m0 root]# cd /extgp3/sun-java/
[root@m0 sun-java]# chmod +x j2sdk-1_4_1_03-linux-i586-rpm.bin
[root@m0 sun-java]# ./j2sdk-1_4_1_03-linux-i586-rpm.bin
                Sun Microsystems, Inc.
            Binary Code License Agreement

READ THE TERMS OF THIS AGREEMENT AND ANY PROVIDED
SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT")
CAREFULLY BEFORE OPENING THE SOFTWARE MEDIA PACKAGE.  BY
OPENING THE SOFTWARE MEDIA PACKAGE, YOU AGREE TO THE TERMS
OF THIS AGREEMENT.  IF YOU ARE ACCESSING THE SOFTWARE
ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY
SELECTING THE "ACCEPT" BUTTON AT THE END OF THIS
AGREEMENT.  IF YOU DO NOT AGREE TO ALL THESE TERMS,
PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF
PURCHASE FOR A REFUND OR, IF THE SOFTWARE IS ACCESSED
ELECTRONICALLY, SELECT THE "DECLINE" BUTTON AT THE END OF
THIS AGREEMENT.


...(author omits lines)


Do you agree to the above license terms? [yes or no]
yes
Unpacking...
Checksumming...
0
0
Extracting...
UnZipSFX 5.40 of 28 November 1998, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  inflating: j2sdk-1_4_1_03-fcs-linux-i586.rpm
Done.
```

Install Java SDK

The CAMgr code is written in Java and is necessary to install the Java platform on the m0 machine. For our installation, we chose to install Sun Microsystem's Java Software Developer Kit (SDK). This package includes the Java APIs and classes necessary for the toolkit to function properly.

More information about Java SDK can be found at the following Web site:

http://java.sun.com

Java SDK is installed using the `rpm` command. Make sure the RPM package of SDK is present on m0.

As root, issue the following commands for this purpose:

```
# cd /expgt3/sun-java/
# rpm -ivh j2sdk-1_4_1_03-fcs-linux-i586.rpm
```

Set environment variables

Next, the environment variables need to be set as shown in Table A-8 in order for the location of Java to be known throughout the system.

Table A-8   Variable

Variable	Description	Value
JAVA_HOME	Base directory of Sun Java	/usr/java/j2sdk1.4.1_03

Make sure you place the variable declarations in the correct position, before the existing export PATH statement.

As root, use your preferred editor to modify the /etc/profile file, as follows:

```
if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

JAVA_HOME=/usr/java/j2sdk1.4.1_03
PATH=$JAVA_HOME/bin:$PATH
export JAVA_HOME

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

...(author omits lines)
```

Create a certificate authority

The Globus Toolkit page at http://www.globus.org/ includes the Globus Simple CA Package. This package provides a simple and useful method for setting up a certificate authority (CA) that can be used for test Grid environments. However, in order to install the CA provided in the Globus Toolkit page, at least one bundle of GT2 or GT3 needs to be installed on the machine due to a dependency. So, we decided not to install this CA, but instead choose OpenSSL, which was provided by Red Hat Linux during the installation. More information on OpenSSL can be found at http://www.openssl.org/.

These are the necessary macro activities needed to create the CA:

► Create the CA directory structure.
► Copy the CA configuration file.
► Set up the CA.
► Copy the certificate/public key to the /CA directory.

CA directory structure

Create the directory structured defined in Table A-5 on page 76.

As root, issue the following commands to create the CA directory structure:

```
# mkdir /CA
# mkdir /CA/IN
# mkdir /CA/OUT
# mkdir /CA/PROCESSED
# chmod 766 /CA/IN
# chmod 744 /CA/OUT
```

CA configuration file

Copy the openSSL /usr/share/ssl/opensl.cnf configuration file to our CA directory. This configuration file contains options for openSSL, in case you want to customized it.

As root, issue the following command to copy the file:

```
# cp /usr/share/ssl/openssl.cnf /CA
```

Be sure to include the definition of the SSLEAY_CONFIG environment variable in /etc/profile, as presented in Table A-9. This variable is used by the internal scripts executed by the **openssl** command.

Table A-9   SSLEAY_CONFIG variable

Variable	Description	Value
SSLEAY_CONFIG	OpenSSL configuration file	-config /CA/openssl.cnf

As root, use your preferred editor to add the line into /etc/profile, as shown next:

```
...(author omits lines)


HOSTNAME=`/bin/hostname`
HISTSIZE=1000

if [ -z "$INPUTRC" -a ! -f "$HOME/.inputrc" ]; then
    INPUTRC=/etc/inputrc
fi

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUTRC

export SSLEAY_CONFIG="-config /CA/openssl.cnf"

...(author omits lines)
```

Once this line is added to /etc/profile file, you must source /etc/profile so that the necessary variables are set.

As root, issue the following command for this purpose, or re-login as root:

```
# . /etc/profile
```

CA setup

The CA script command is used to set up a new CA. This script is provided by openSSL and is located in the /usr/share/ssl/misc directory, and it will display a set-up screen where you should enter the information presented in Table A-10 on page 82 and Table A-11 on page 82.

Table A-10   CA Setup screen prompt

Setup screen prompt	To be entered
CA certificate file name (or Enter to create)	Press the Enter key.
Enter PEM pass phrase	<passphrase>.
Verifying - Enter PEM pass phrase	<passphrase>.
Distinguished Name (DN)	Information that will be part of the CA DN, Distinguished Name. Table A-11 presents the DN used in our lab.

Important: Be sure to write the PEM passphrase (password) down where you will not forget it. You will have to uninstall/reinstall the CA if you forget the password.

Table A-11   CA Distinguished Name used in our lab

DN component	Value entered
Country Name (two letter code) [GB]:	US
State or Province Name (full name) [Berkshire]:	Texas
Locality Name (for example, city) [Newbury]:	Austin
Organization Name (for example, company) [My Company Ltd]:	ITSO
Organizational Unit Name (for example, section) []:	XINGU
Common Name (for example, your name or your server's host name) []:	m0
Email Address []:	cd@m0

As root, issue the following command to set up a new CA:

```
# cd /CA
# /usr/share/ssl/misc/CA -newca
```

You should get an output similar to Example A-4.

Example: A-4   CA setup output

```
[root@m0 CA]# /usr/share/ssl/misc/CA -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
...................................................++++++
........++++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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.
-----
```

```
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Texas
Locality Name (eg, city) [Newbury]:Austin
Organization Name (eg, company) [My Company Ltd]:ITSO
Organizational Unit Name (eg, section) []:XINGU
Common Name (eg, your name or your server's hostname) []:m0
Email Address []:ca@m0
```

Public key

The certificate/public key (cacert.pem) is created during the CA setup and posted in the /CA/demoCA directory. This certificate/public key must be distributed later to all Grid machines, but with a different file name that is formed by the hash number of the certificate plus ".0". The `c_hash` command can be used to get the hash of your CA.

As root, issue the following command to get the hash number of the CA certificate and copy it to the /CA directory.

```
# cd /CA/demoCA
# HASH='/usr/share/ssl/misc/c_hash cacert.pem | cut -c 1-10'
# echo $HASH
# cp cacert.pem /CA/$HASH
```

> **Attention:** In the previous commands, the symbol ' represents the back quote, that is, backtick, which is used to expand shell commands to stdout.

You should get an output similar to Example A-5.

Example: A-5   Get the hash number of the CA output

```
[root@m0 CA]#  cd /CA/demoCA
[root@m0 demoCA]# HASH=`/usr/share/ssl/misc/c_hash cacert.pem | cut -c 1-10`
[root@m0 demoCA]# echo $HASH
5243259c.0
[root@m0 demoCA]#  cp cacert.pem /CA/$HASH
[root@m0 CA]# ls /CA
5243259c.0  demoCA  IN  openssl.cnf  OUT  PROCESSED
```

Deploy the CAMgr tool

CAMgr is a Java program and set of related classes for managing a simple certificate authority based on OpenSSL. As mentioned in "Software for M0 machine" on page 92, CAMgr can be obtained at the ITSO FTP site at:

ftp://www.redbooks.ibm.com/redbooks/REDP3697

Installation

This program is intended to be executed by root. Copy the CAMgr.jar and camgr files into the /usr/local/bin directory. Make sure this directory is in your PATH. The example expects CAMgr.jar to be located in the current directory. You may want to specify a specific directory.

As root, issue the following command to install the CAMgr tool.

```
# cd /usr/local/bin
# cp /expgt3/camgr/CAMgr.jar .
# cp /expgt3/camgr/camgr .
# chmod +x camgr
```

You should get an output similar to Example A-6.

Example: A-6   camgr script file

```
#! /bin/sh
#   Modify the classpath parameter as needed to
#    point to the CAMgr.jar file.
java -classpath ./CAMgr.jar itso.austin.camgr.CAMgr
```

Testing

Make sure camgr is set properly before testing it. Source /etc/profile and then run the `camgr` command. You should receive a screen similar to Figure A-2.

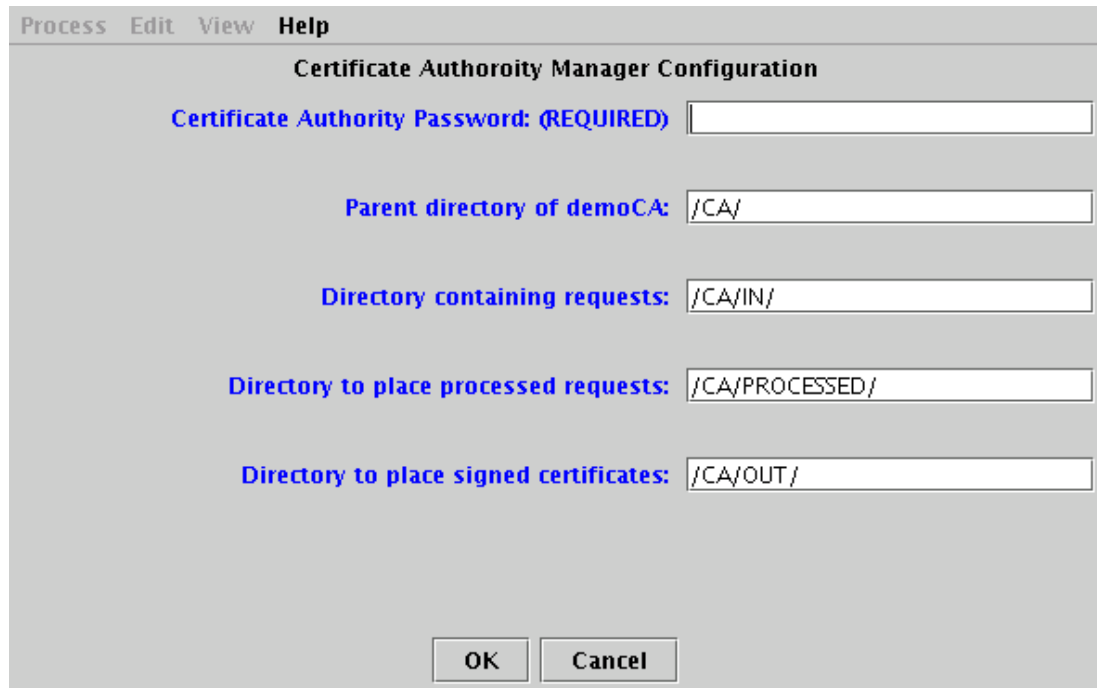As root, issue the following commands to source /etc/profile and run CAMgr.

```
# . /etc/profile
# camgr
```


Figure A-2   CAMgr

Summary

The m0 machine is now ready. It has all the necessary software for Grid server installations and NTP, and the certificate authority is prepared. In order to proceed with Grid server installation and configuration, go to Chapter 3, "Installing on Linux" on page 23.

Managing a certificate

This appendix presents a tool and procedures for managing certificate requests and certificates. The most important management function is signing a certificate, but the tools can also be used to check the contents of the certificates generated by the CA and delete them as well.

The primary tool for managing certificates is the `openssl ca` command. This command is provided by the OpenSSL package. See the OpenSSL documentation for more details on the various options related to signing certificate requests.

In our lab environment, we have put together a procedure and a directory structure for handling certificate requests. We also developed a small Java application that provides a graphical interface for managing the various files associated with the certificate authority. This appendix, in conjunction with the information already presented on setting up a Grid environment in "Create a certificate authority" on page 80, describes how you might manage your certificate authority using our CAMgr tool.

85

CAMgr tool

CAMgr is a Java program and set of related classes for managing a simple certificate authority based on OpenSSL. CAMgr can be obtained from the ITSO FTP site at ftp://www.redbooks.ibm.com/redbooks/REDP3697/. Information about the CAMgr files are available at that site; refer to "Software for M0 machine" on page 92 for more details.

Assumptions

By default. CAMgr assumes the following:

► The simple CA directory (demoCA) is installed in /CA/.

► Certificate requests from users are copied into /CA/IN.

► After processing requests, the request files will be copied to /CA/PROCESSED.

► The signed certificates will be placed in /CA/OUT.

► All of the above directories already exist when CAMgr is launched.

► The simple CA was installed as root and therefore requires the root password to sign certificates.

► CAMgr will be run as root.

Usage

Execute the camgr script to launch the CAMgr GUI. The first screen shown (Figure B-1) prompts the user for their CA password (root password), and allows the user to modify the assumed directory structure.
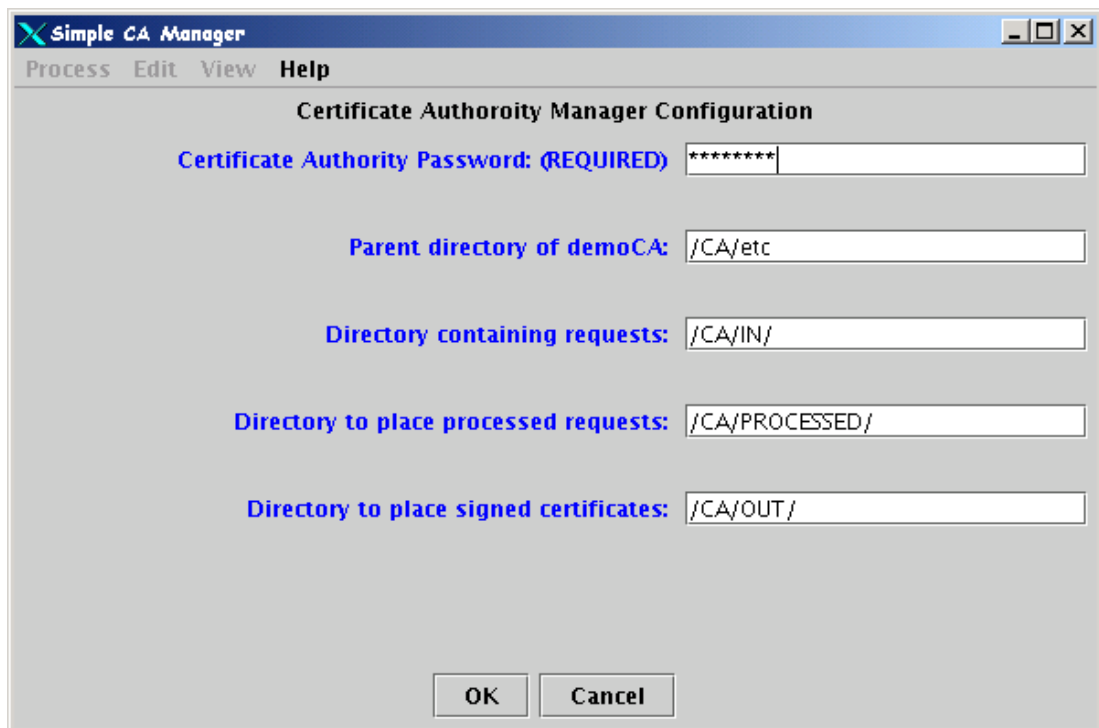
Figure B-1   Initial screen for CAMgr

Once the password is entered and any changes are made to the default directory structure, select **OK**, and you will see the default view of existing certificates, as shown in Figure B-2. The list of certificates is read from the demoCA/index.txt file, which references individual certificate files stored in the demoCA/newcerts directory. By selecting a certificate in the top panel, the details of the certificate is shown in the bottom panel. A **Delete** button is provided to delete the selected certificate or certificates if more than one is selected.
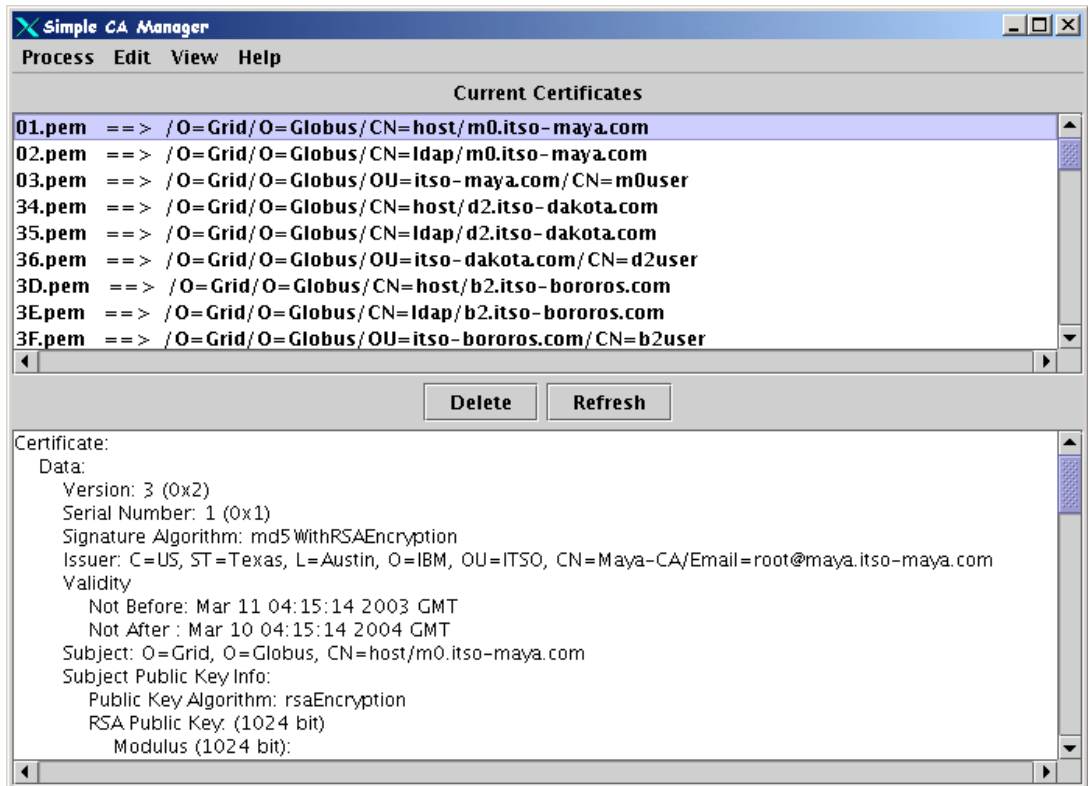
Figure B-2   List of existing certificates

By selecting the View menu, the user may choose one of several views:

► Certificates: The view described in Figure B-2.

► Requests: A similar view that shows all the requests in CA/IN. This view provides a Process button that allows the user to process one or more requests. After successfully processing the request, the request file is copied to the /CA/PROCESSED directory.

► Processed: A similar view that shows all the processed requests (from /CA/PROCESSED). A MvToReq button is provided to move a processed request back to the /CA/IN directory so that it may be processed again if necessary.

► Signed: A similar view that shows the /CA/OUT directory containing the signed certificates created by the Process option in the Requests view.

► Output: This panel shows the output from the commands issued. Currently, the only command that is implemented and shows its output in this window is the Process command of the Request window. The Process command invokes the `openssl ca` command to process the request. This command's output is shown in this view.

► Configuration: Shows the configuration panel described above.

Sign a certificate

By selecting the **View** → **Requests** menu item, all current outstanding certificate requests are shown. The upper panel will display the names of all files in the /CA/IN (by default) directory. Again, by selecting a specific item, the details are shown in the lower panel. When viewing requests, a Process button is provided. If the button is activated, the program will attempt to create a signed certificate by invoking the `openssl ca` command. Aside from invoking this command, several things occur.

The file containing the request is moved from the /CA/IN directory to the CA/PROCESSED directory.

The actual `openssl` command that is used is the following:

```
openssl ca -config CA/etc/openssl.cnf -key password -batch -policy policy_anything /
-out /CA/OUT/outfilename -infiles infilename
```

where:

► *password* is the password specified on the configuration screen.

► *outfilename* is the name of the file that will be written to /CA/OUT.

► *infilename* is the name of the file containing the request.

Note: The `openssl ca` command shown above could be used on a Linux command line or shell script instead of using the CAMgr tool.

By our convention, requests are created with a file name, such as bjacobuser_request.pem (see Figure B-3 on page 89), to identify this as a request for a user certificate for bjacob. When CAMgr signs the certificate, it writes it out as bjacob_cert.pem in the /CA/OUT directory to identify this as a signed certificate. In our lab environment, the user can then retrieve the certificate from the /CA/OUT directory.

The `openssl ca` command also writes a copy of this signed certificate into the demoCA/newcerts directory as xx.pem, where xx is a sequentially allocated hex pair.

If the processing of the `openssl ca` command fails for some reason, the file will still get written to the /CA/OUT directory. You may want to delete this file (using the Signed view) before attempting to sign the certificate again.
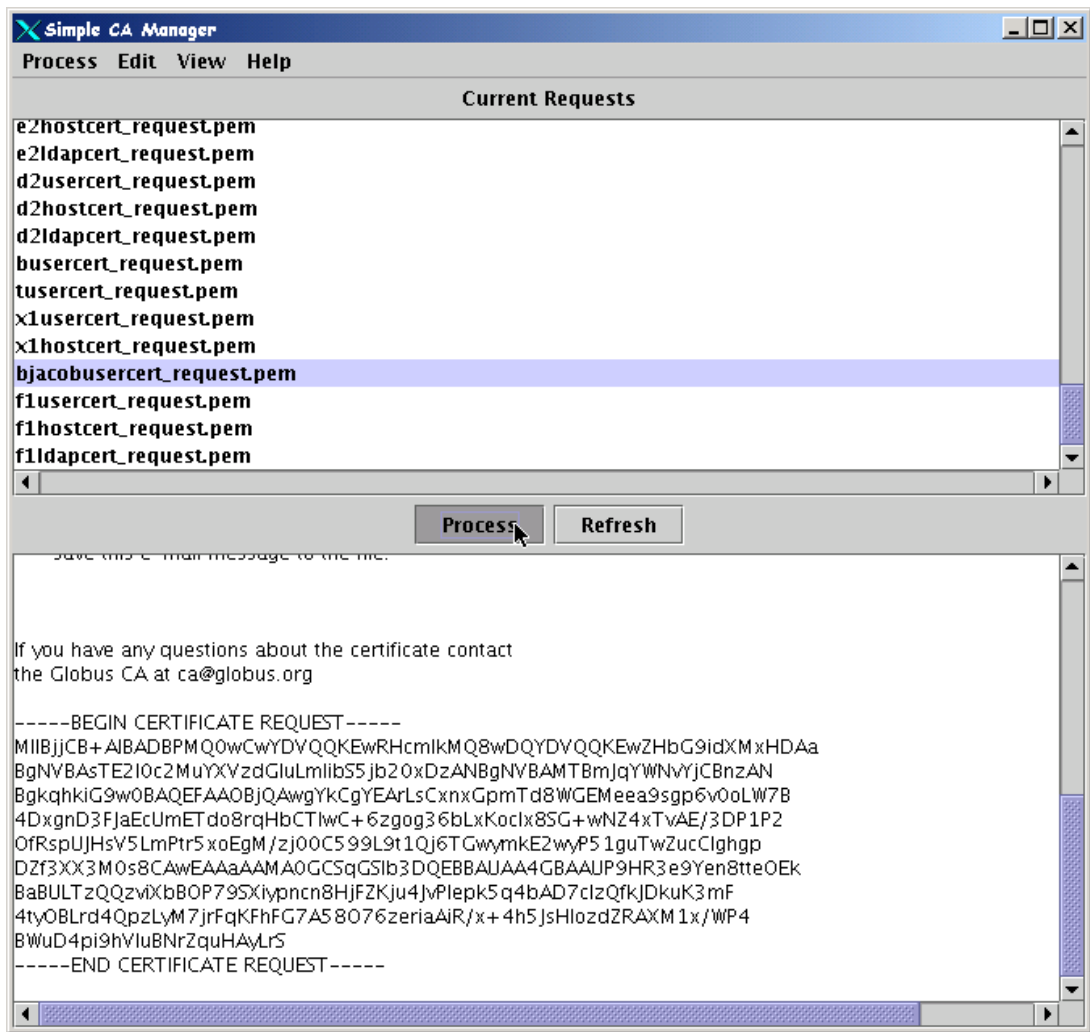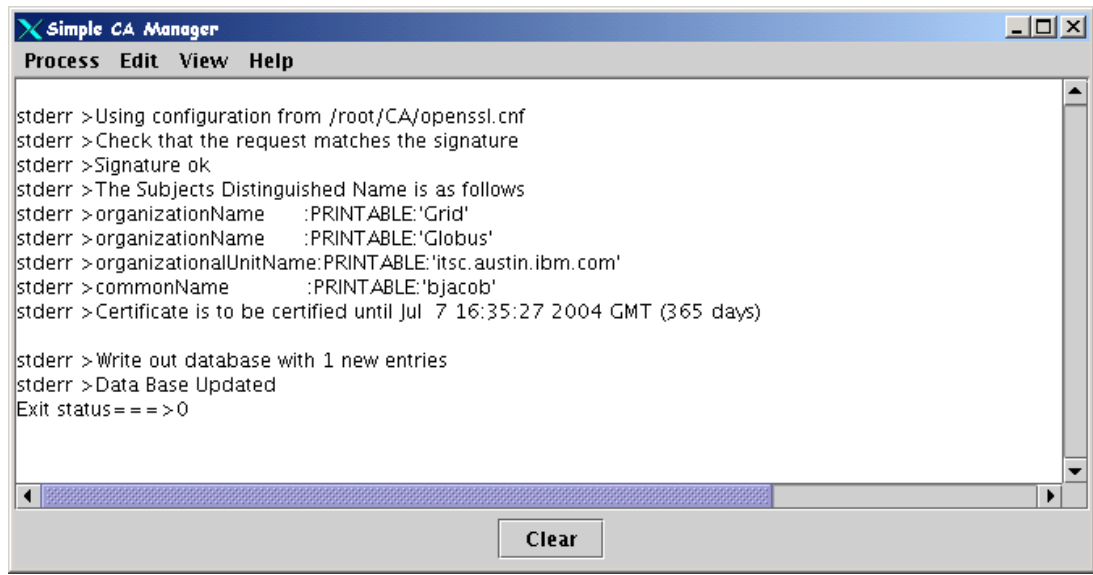
Figure B-3   Processing a request from the Current Requests view

After processing a request, the view changes to the Output view, as shown in Figure B-4 on page 90, which shows the results of the `openssl ca` processing.

Figure B-4   Results of the request processing

Note that the `openssl ca` command writes its output to stderr even if processing is successful. The CAMgr indicates within its output window that data was written to stdout or stderr.

Known limitations

The following are known limitations of this simple tool:

► Configuration information is not persistent between invocations of CAMgr.

► There is little or no cross processing, that is, a deletion of a signed certificate does not delete the certificate from the CA's database of certificates (/CA/etc/demoCA/newcerts and index.txt), or vice versa.

► There is little or no logic to handle failure when processing a request. The error is shown in the output view, but the user must clean up and correct all related files before processing again.

► There is no online help yet.

► Some error messages are still written to the text window used to launch the application.

► There are no keyboard shortcuts. Most interaction with the GUI must be done via the mouse.

Summary

The CAMgr tool is a Java-based application that is used to help manage certificate requests and signed certificates. Other than displaying and managing the related files, its main function is to call the `openssl ca command` create and sign certificates. CAMgr is intended to provide a GUI to simplify the process of managing the CA, but be aware that it has not undergone extensive testing and should not be used in production environments.

C
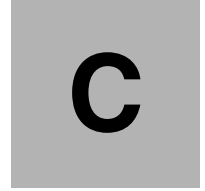
Software installed

This appendix presents the tools, files, and software packages used to build the lab environment. Also, it includes the md5sum checksum of all the files we used to build the environment.

Software for M0 machine

This section summarizes the software used on the infrastructure machine known as m0.

Linux

Table C-1 shows the Red Hat Linux we used in the lab environment.

Table C-1   Linux for m0

Package	Source	Notes
Red Hat Version 9.0 Workstation package	http://www.redhat.com	To be used as the infrastructure server

Tools

Table C-2 shows the list of tools used in the lab environment.

Table C-2   Tools used in m0

Tool	Version	Source	File name
Java SDK	1.4.1	http://java.sun.com	j2sdk-1_4_1_03-linux-i586-rpm.bin
CAMgr	N/A	http://www.redbooks.ibm.com/redbooks/REDP3697	README CAMgr.jar camgr CAMGR-docs.tar

Software for X1 and X2 machines

This section summarizes the software used on Grid machines, also known as the x machines (x1, x2, and so on).

Linux

Table C-3 shows the Red Hat Linux we used in the lab environment.

Table C-3   Linux for x1 and x2

Package	Source	Notes
Red Hat Version 9.0 Workstation package	http://www.redhat.com	For installing GPT Source Installation Package
Red Hat Version 8.0 Workstation package	http://www.redhat.com	For installing GPT Linux Installation Package

Tools

Table C-4 shows the list of tools used in the lab environment.

Table C-4   Tools used in the lab environment

Tool	Version	Source	File name
Java SDK[a]	1.4.1	http://java.sun.com	j2sdk-1_4_1_03-linux-i586-rpm.bin

Tool	Version	Source	File name
Apache Ant	1.5.3	http://ant.apache.org	apache-ant-1.5.3-1-bin.tar.gz
Junit [b]	3.8.1	http://www.junit.org	junit3.8.1.zip
PostgreSQL [c]	7.3.3	http://www.postgresql.org	postgresql-libs-7.3.3-1PGDG.i386.rpm postgresql-7.3.3-1PGDG.i386.rpm postgresql-server-7.3.3-1PGDG.i386.rpm

a. We noticed that when installing GT3 from source, the installation sometimes hung while deploying mds-db. When we updated to JDK 1.4.2, the problem did not occur anymore and installation was successful.

b. Optional tool, but required in our lab environment to run some tests.

c. The RPM packages for Red Hat Version 9.0 and Red Hat Version 8.0 are not the same, so pay attention to which package you want to download.

Globus Toolkit

Table C-5 shows the GT3 packages used in the lab environment.

Table C-5   Globus packages Version 3.0.1

Package	Source	File name
GPT Source Installation Package	http://www-unix.globus.org/toolkit/download.html	gt3.0.1-source-installer.tar.gz
GPT Linux Installation Package	http://www-unix.globus.org/toolkit/download.html	gt3.0.1-linux-installer.tar.gz

MD5 checksums

The `md5sum` command can be used to check if you are using the same versions of the files we used by comparing the values against the list in Example C-1.

Example: C-1   Checksum files

```
[root@m0 expgt3]# md5sum sun-java/j2sdk-1_4_1_03-linux-i586-rpm.bin camgr/Redp3697.zip \
apache/apache-ant-1.5.3-1-bin.tar.gz junit/junit3.8.1.zip \
postgresql/postgresql-libs-7.3.3-1PGDG.i386.rpm \
postgresql/postgresql-7.3.3-1PGDG.i386.rpm \
postgresql/postgresql-server-7.3.3-1PGDG.i386.rpm globus/gt3.0.1-*

fc5bc1e99e58d48c54aa6a9a4900aaf6  sun-java/j2sdk-1_4_1_03-linux-i586-rpm.bin
705a80f2a08fc82c421eaca816fb7a24  camgr/Redp3697.zip
6bf0cd523d176be4fc122de9b107fa11  apache/apache-ant-1.5.3-1-bin.tar.gz
5110326e4b7f7497dfa60ede4b626751  junit/junit3.8.1.zip
76b57d8730b12d753c6512f83de6636a  postgresql/postgresql-libs-7.3.3-1PGDG.i386.rpm
9e2b24efceee8dacbde967679cf88f17  postgresql/postgresql-7.3.3-1PGDG.i386.rpm
32f6f6996654ae0a2ace4d46249aefa2  postgresql/postgresql-server-7.3.3-1PGDG.i386.rpm
f17c24fd0ed8065bf940a6a2b505a934  globus/gt3.0.1-linux-installer.tar.gz
2e70ab22914ced9a63311e021ecf3914  globus/gt3.0.1-source-installer.tar.gz
```

D

Additional material

This Redpaper refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this Redpaper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/REDP3697

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the Redpaper form number, REDP3697.

Using the Web material

The additional Web material that accompanies this Redpaper includes the following files:

File name	Description
README	A text file containing the latest information about the tool.
CAMgr.jar	A JAR file containing the various class files.
camgr	A simple shell script for launching the application.
CAMGR-docs.tar	A tar file containing the javadoc documentation for this application.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	20 MB
Operating System:	Linux
Processor:	Pentium 1 GHz processor or compatible
Memory:	512 MB

How to use the Web material

Refer to the following sections:

- ► "CAMgr tool" on page 86.
- ► "Software for M0 machine" on page 92
- ► "Populate installation image repository" on page 77

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this Redpaper.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 100. Note that some of the documents referenced here may be available in softcopy only.

► *Enabling Applications for Grid Computing with Globus*, SG24-6936

Other publications

These publications are also relevant as further information sources:

► *Scorpion: Simplifying the Corporate IT Infrastructure. An IT Health Check for Business Executives*, GF22-5168-01

► Foster, et al, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1998, ISBN 1558604758

► *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, found at:

  http://www.globus.org/research/papers/anatomy.pdf

► *A Brief Introduction to Grid Technology*, found at:

  http://www.bo.infn.it/alice/introgrd/introgrd/

► *Computational Grids*, found at:

  http://www.globus.org/research/papers/chapter2.pdf

► *The Globus Project: A Status Report*, found at:

  ftp://ftp.globus.org/pub/globus/papers/globus-hcw98.pdf

► *Globus Toolkit 3 Core: A Grid Service Container Framework*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf

► *Globus Toolkit 3.0: Grid Service Development Tools Guide*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/tools_guide.html

► *Globus Toolkit 3: Java Programmer's Guide Core Framework*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/java_programmers_guide.html

► *Globus Toolkit 3: User's Guide Core Framework*, found at:

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/users_guide.html

► *GridFTP: Protocol Extensions to FTP for the Grid*, found at:

  http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

► *GridFTP Update January 2002*, found at:

  http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf

- *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, found at:

  http://www.globus.org/research/papers/ogsa.pdf
  http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf

- *A Resource Management Architecture for Metacomputing Systems*, found at:

  ftp://ftp.globus.org/pub/globus/papers/gram97.pdf

- RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

  http://www.ietf.org/rfc/rfc3280.txt

- *Web Services Conceptual Architecture (WSCA 1.0)*, found at:

  http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

- *Web Service Description Language (WSDL)*, found at:

  http://www.w3.org/TR/wsdl

Online resources

These Web sites and URLs are also relevant as further information sources:

- Apache Ant Web site

  http://ant.apache.org

- Apache Axis

  http://ws.apache.org/axis

- The Apache Jakarta Project

  http://jakarta.apache.org/tomcat

- Argonne National Laboratory

  http://www.anl.gov/

- Choosing an EJB application server

  http://www.ibm.com/developerworks/java/library/co-tipejbs.html?dwzone=java

- Commodity Grid Kits

  http://www-unix.globus.org/cog/

- Global Grid Forum

  http://www.ggf.org/
  http://www.gridforum.org/

- The Globus Project

  http://www.globus.org/

- Globus Toolkit 3.0 Configuration

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/admin/configuration.html

- Globus Toolkit 3.0 Documentation

  http://www-unix.globus.org/toolkit/3.0/ogsa/docs/

- Globus Toolkit 3.0 FAQ Web site

  http://www-unix.globus.org/toolkit/faq.html

- Globus Toolkit 3.0 GRAM Architecture

  http://www-unix.globus.org/developer/gram-architecture.html

- ► Globus Toolkit 3.0 Index Service Overview

  http://www.globus.org/ogsa/releases/final/docs/infosvcs/indexsvc_overview.html

- ► Globus Toolkit 3.0 Index Service User's Guide

  http://www.globus.org/ogsa/releases/final/docs/infosvcs/indexsvc_ug.html

- ► Globus Toolkit 3.0 and the OGSI architecture: Overview, found at:

  http://www-106.ibm.com/developerworks/library/gr-gt3/

- ► Globus Toolkit 3.0 Programmer's Tutorial

  http://www.casa-sotomayor.net/gt3-tutorial/

- ► Globus Toolkit 3.0.2 Download

  http://www-unix.globus.org/toolkit/download.html

- ► Globus Toolkit Advisories

  http://www-unix.globus.org/toolkit/advisories.html

- ► The Globus Toolkit Bugzilla

  http://bugzilla.globus.org/bugzilla/

- ► Globus Toolkit Data Management Services

  http://www-unix.globus.org/developer/data-management.html

- ► Globus Toolkit Documentation

  http://www-unix.globus.org/toolkit/documentation.html

- ► Globus Toolkit Resource Management

  http://www-unix.globus.org/developer/resource-management.html

- ► The Globus Toolkit Support

  http://www.globus.org/toolkit/support.html

- ► Grid Service Specification

  http://www.gridforum.org/

- ► GSI Configuration Information

  http://www.globus.org/security/config.html

- ► IBM Grid

  http://www.ibm.com/grid/

- ► IBM On-demand business

  http://www.ibm.com/e-business/

- ► IBM Redbooks

  http://www.redbooks.ibm.com

- ► Information Services in the Globus Toolkit 3.0 Release

  http://www.globus.org/ogsa/releases/final/docs/infosvcs

- ► JDBC Data Access API

  http://java.sun.com/products/jdbc

- ► Junit Web site

  http://www.junit.org

- ► Open Grid Services Infrastructure Working Group (OGSI-WG)

  http://www.ggf.org/ogsi-wg

- ▶ The Open Group

  http://www.opengroup.org
- ▶ OpenSSL Project

  http://www.openssl.org/
- ▶ PostgreSQL Web site

  http://www.postgresql.com
- ▶ Reliable File Transfer Service

  http://www-unix.globus.org/toolkit/reliable_transfer.html
- ▶ Sun Java Web site

  http://java.sun.com
- ▶ UDDI Web site

  http://www.uddi.org/
- ▶ W3 Schools

  http://www.w3schools.com/
- ▶ WebSphere software platform

  http://www.ibm.com/websphere
- ▶ World Wide Web Consortium (W3C)

  http://www.w3.org/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

$ANT_HOME 29, 38
$CLASSPATH 66
$GLOBUS_LOCATION 33, 38, 46, 50, 62, 64
$JAVA_HOME 28–29, 38
$X509_USER_PROXY 62
/etc/hosts 77

A

Apache Ant
   installation 28

B

Basic Counter Sample 53
   running on x1 53
   running on x2 61

C

CA
   certificates 41, 44
   configuration 80
   setup 39, 74
CAMgr
   usage 86
camgr 45, 84, 86
certificates 39, 41, 62, 74, 77
   directories 76
   naming 76
   requesting 42
   signing 42
checksum files 93
CLI 50, 57
Command line interface
   see CLI
Common Object Request Broker Architecture
   see CORBA
computation grid 3
configuration
   /etc/hosts 27, 77
   certificates 42
   environment variables 29, 37, 80
   Grid security files 48
   GSI 38
   Linux 26
   m0 directories 27
   NTP 26, 77
   PostgreSQL 31
   user and group IDs 27
container
   EJB 17
   embedded 17
   servlet 17
   stand-alone 17
   starting 50, 54
   what is 17
CORBA 10

D

data grid 3
data management 5
Data Virtualization 3
Discovery 6

E

EJB 10
Embedded Hosting Environment 17
Enterprise JavaBeans
   see EJB

F

Factory 6, 15, 20, 45, 55, 69

G

GGF 2
Global Grid Forum
   see GGF
Globus Toolkit
   configuration 42
   installation 33
   introduction 5
   previous versions 5
Globus Toolkit version 1
   see GT1
Globus Toolkit version 2
   see GT2
Globus Toolkit version 3
   see GT3
globus-start-container 50, 54, 65
globus-stop-container 61, 65
GRAM 5, 20
Grid Computing
   classes 3
   introduction 2
Grid File Transfer Protocol
   see GridFTP
Grid Resource Allocation Management
   see GRAM
Grid Security Infrastructure
   see GSI
Grid service
   description 12
   how to write 21
Grid Service Handle
   see GSH

Globus Toolkit 3.0 Quick Start

Globus Toolkit 3.0 components

Install and run examples

Open Grid Services Infrastructure compliant

This IBM® Redpaper describes the early experiences of the ITSO team with the Globus Toolkit 3.0. This major Grid implementation, known as GT3, is compliant with the new Open Grid Services Infrastructure (OGSI).

The goal of this Redpaper is to provide the critical jump-start for someone who wants to learn about GT3 but has little or no experience with prior Globus releases or grid computing in general. It whos you how to implement a GT3 demo or a proof-of-concepts scenario. Also, it illustrates the high-level concepts of the toolkit components and the overall architecture.

Redpaper

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information: ibm.com/redbooks