

Grid Market Directory: A Web and Web Services based Grid Service Publication Directory

Jia Yu and Rajkumar Buyya

Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
Email: {jiayu, raj}@cs.mu.oz.au

1. Introduction

Grids enable the sharing, exchange, discovery, selection, and aggregation of geographically/Internet-wide distributed heterogeneous resources--such as computers, data sources, visualization devices, and scientific instruments [1]. Accordingly, they have been proposed as the next-generation computing platform and global cyber-infrastructure for solving large-scale problems in science, engineering, and business. As the Grid comprises of a wide variety of resources owned by different organization with different goals, the management of resource is a challenging task. Accordingly, a distributed Grid economy has been proposed as an efficient solution for managing the supply-and-demand for resources [3][2]. To realize the full potential of Grid economy, the Gridbus project [4] has been developing technologies that provide end-to-end support for allocation of resources based on resource providers and consumers quality of service (QoS) requirements. One of the key components of Gridbus system is the Grid Market Directory (GMD). The GMD enables service providers to publish their services and related costs to the public, so that consumers can browse or query by using Web services and find services which meet their budget.

2. GMD Architecture

Grid Market Directory (GMD)

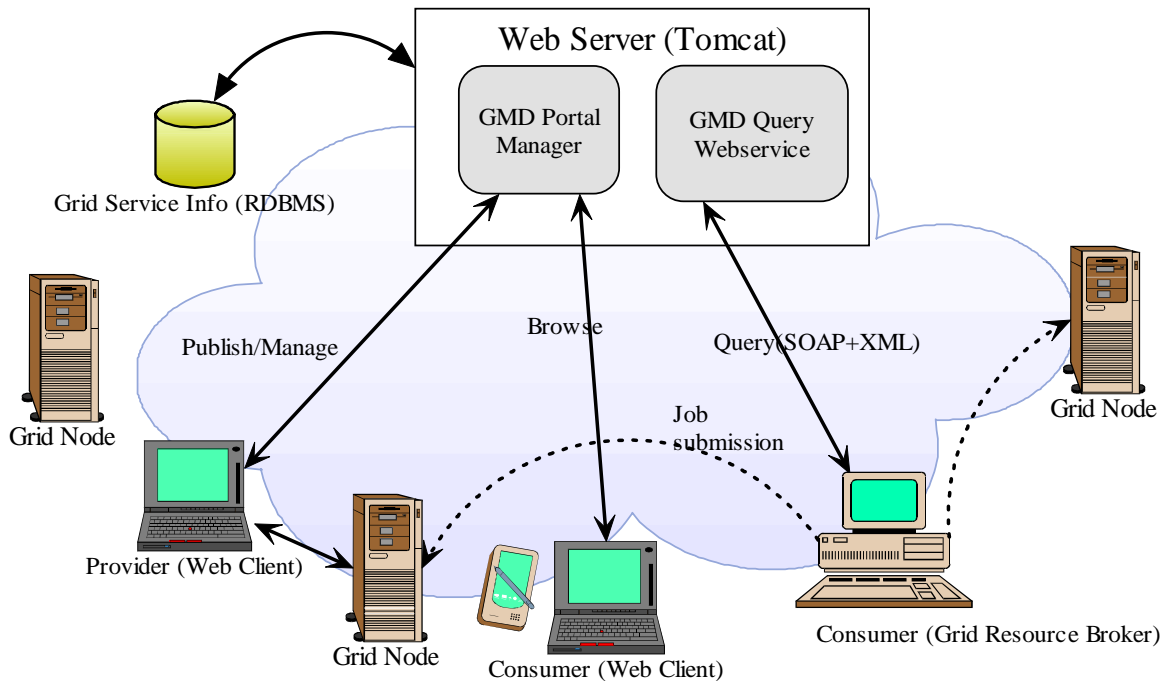


Figure 1: Grid Market Directory Architecture.

Grid Market Directory (GMD) contains two key components:

- GMD Portal Manager allows the grid service providers to publish and manage their services through Web browsers, so that consumers can be informed via accessing the Website supported by the GMD portal manager.
- GMD Query Web-service facilitates program querying, which intends to search in the GMD and find a suitable service to meet the job execution requirement, such as budget, or to get the price of a service for cost calculation and then to make a payment. Consumer program talks to the GMD in XML. The structure of the query and response messages is described in section 6. In order to allow programs written in different computer languages to converse to the GMD, SOAP is used as the transport mechanism for carrying these query and response messages.

Two components are held by a Web server like Tomcat provided by the Apache Jakarta project (<http://jakarta.apache.org/tomcat/>). The information of the grid services and service providers are deposited in the database where the GMD portal manager and query Web-service communicate.

3. Provider Registration

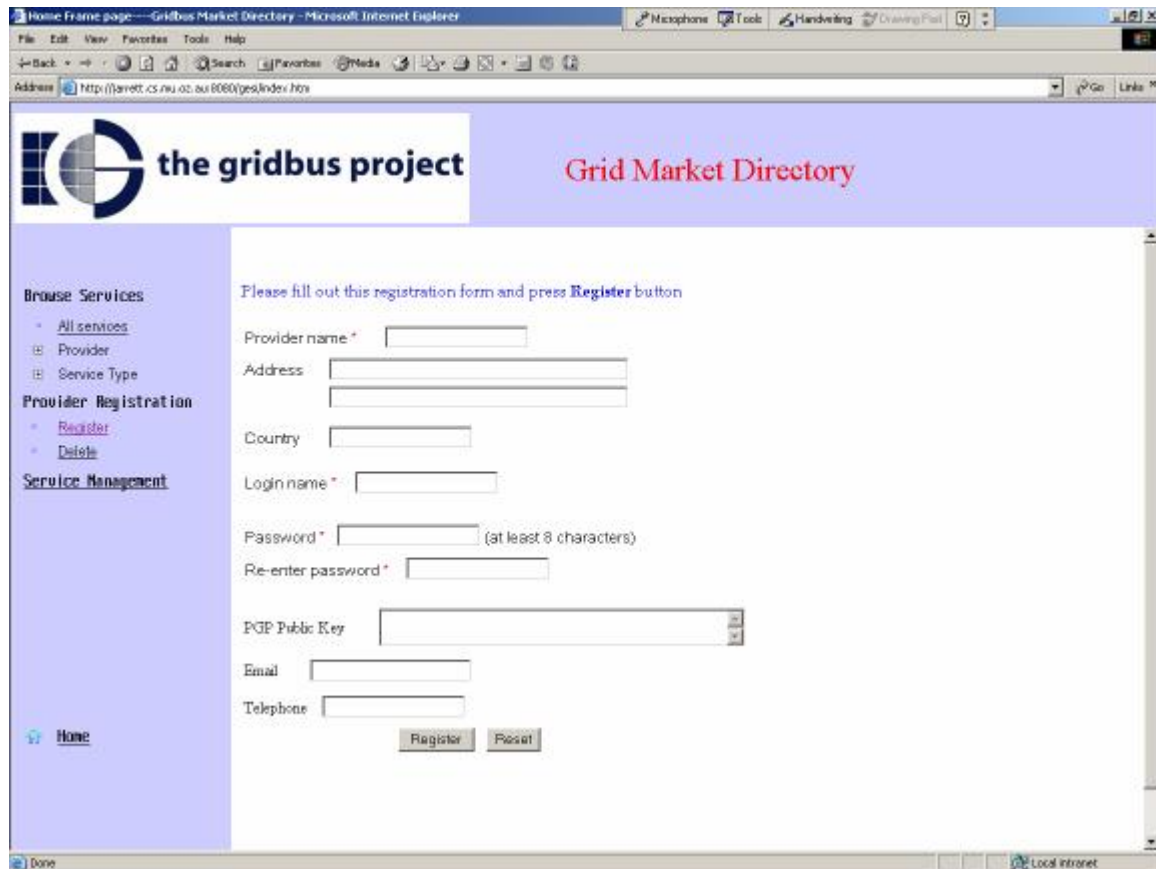


Figure 2: Provider registration form for the Grid Market Directory

Before publishing services, providers need to register to the GMD via Web. The registration form is described in figure 2. In the registration process, provider's login name and password are required to provide; when providers want to publish and modify their services, the GMD authenticates providers by their login name and password.

Delete option in the Provider Registration section is used to delete a provider. After a provider clicks Delete and login, warning page appears. After confirming, the provider's information and its corresponding services will be removed from the GMD.

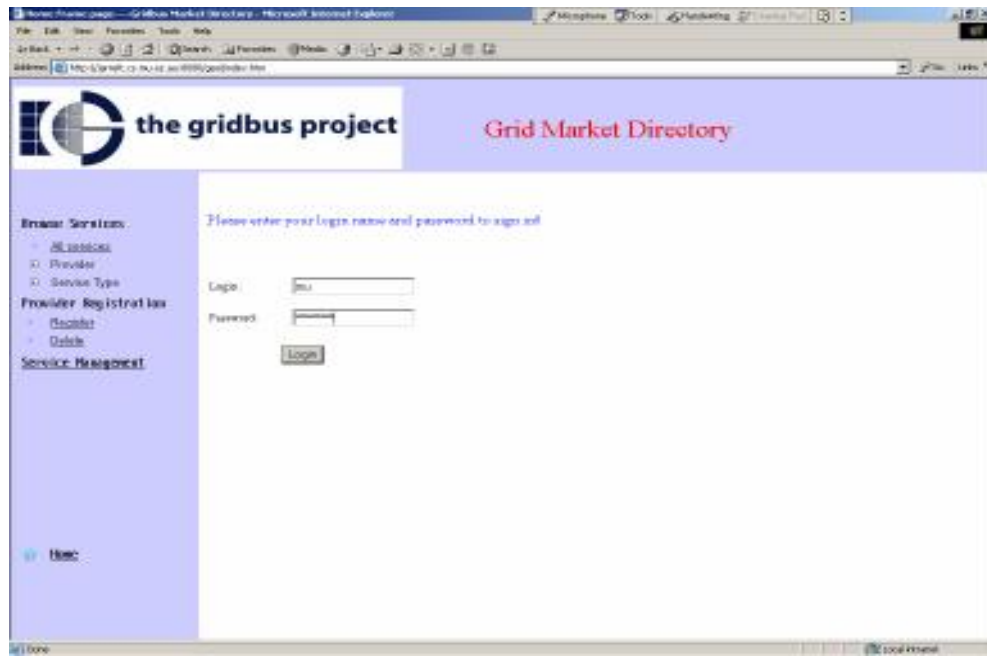


Figure 3: Provider login page for provider quitting

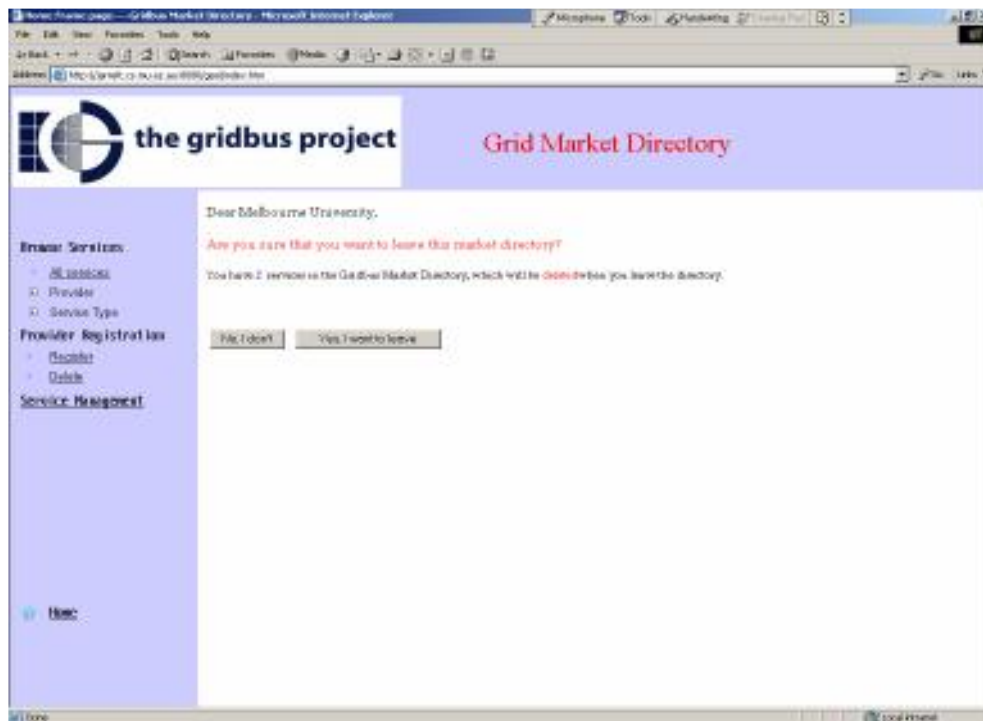


Figure 4: Provider quitting confirm page

4. Service Management

The principle of the GMD service management is to allow service providers to maintain their services by themselves rather than a GMD administrator. Through service management function on the GMD website provided by the GMD portal manager, service providers can add and delete a service, and update their service information as well. Service management option can be found on the GMD homepage. After successful login, service providers can be able to modify their services. As shown in figure 6 and figure 7, there are two major options for the service management, new service addition and service modification.



The image shows a login form titled "Service management area" in red text. Below the title, there is a blue instruction: "Please enter your login name and password to sign in!!". The form contains two input fields: "Login:" followed by a text box, and "Password:" followed by a text box. Below these fields is a "Login" button.

Figure 5: Provider login page for the service management

5. Service browses

There are three ways for consumers to browse grid services in the GMD. Consumers can not only browse all services available in the GMD by the **All services** option, but also browse services offered by a specified provider like IBM. In some circumstance, consumers only need to browse services in a particular area. For example, high energy physics people only want to know how many services providing high energy simulation and how much they cost. For this purpose, services in the GMD are organized by service type, such as Earthquake Engineering and Molecular Docking, and consumers are allowed to browse services by the service type on Web.



Figure 8: service browses menu



The screenshot shows the Grid Market Directory website. The header includes the GRID5 Laboratory logo and the text "Grid Market Directory". The left sidebar contains a "Browse Services" menu with options for "All services", "Provider", and "Service Type". The "Provider" option is selected, showing a list of providers including Branson Global Service, Doshisha University, Gridbus, Melbourne University, Osaka University, and Sydney University. The "Service Type" option is also visible. Below the sidebar, the main content area displays "Melbourne University" and a table of services. The table has columns for Name, Type, Price (Hardware and Software), Host, and Path. Two services are listed: gnet01 and lem. Below the table, there is a note: "*AO-Application Operation(e.g. Docking & molecule)". At the bottom, the website is powered by the gridbus project, with its logo and name.

Name	Type	Price		Host	Path
		Hardware (\$perCPU-hr)	Software (\$perCPU-hr)		
gnet01	globus 4.0	0.4	0.4	gnet01.hpc.unimelb.edu.au	\globus\hps
lem	globus 1.0	0.1	0.1	lem.hpc.unimelb.edu.au	\globus\hps

Figure 9: Service browse by a provider

GRID5 Laboratory

 **Grid Market Directory**

Browse Services

- All services
- Provider
- Service Type**
 - AdHoc Network Simulat
 - Astronomy
 - Crash Simulation
 - Data Mining
 - Earthquake Engineering
 - Financial Modeling
 - globus**
 - High Energy Simulation
 - Image Rendering
 - Manufacturing Planning
 - Molecular Docking
 - NeuroScience
 - Supply-Chain Managemen
 - Weather Forecasting
 - WireFrame Modeling

Provider Registration

- Register
- Delete

globus

Name	Provider	Price		Host	Path
		Hardware (GB/Disk/Sec)	Software (GB/Min)		
galley	Doshisha University	2.0	0.2	galley.doshisha.ac.jp	\globus\hps
cabibbo	Sydney University	3.0	0.3	cabibbo.physics.usyd.edu.au	\globus\hps
date1	Osaka University	1.0	0.1	date1.ces.osaka-u.ac.jp	\globus\hps
gnet01	Melbourne University	4.0	0.4	gnet01.hpc.unimelb.edu.au	\globus\hps
lem	Melbourne University	1.0	0.1	lemhp.unimelb.edu.au	\globus\hps

*AC-Application Operators (e.g. Docking a molecule)

Powered by:

 **the gridbus project**

Figure 10: Service browse by a service type

6. Security Mechanism

Service information modification supported by the GMD website is protected by the login name and password. Java servlet HttpSession API is used for the GMD security implementation. As shown in figure 11, after logging in a name attribute is set to the HttpSession object of corresponding request session. When the user logs out, the name attribute is deleted from the object. Therefore, every modification page knows whether the user of the request has logged in through checking the name attribute. If the request is illegal, the GMD home page will be redirected to the browser.

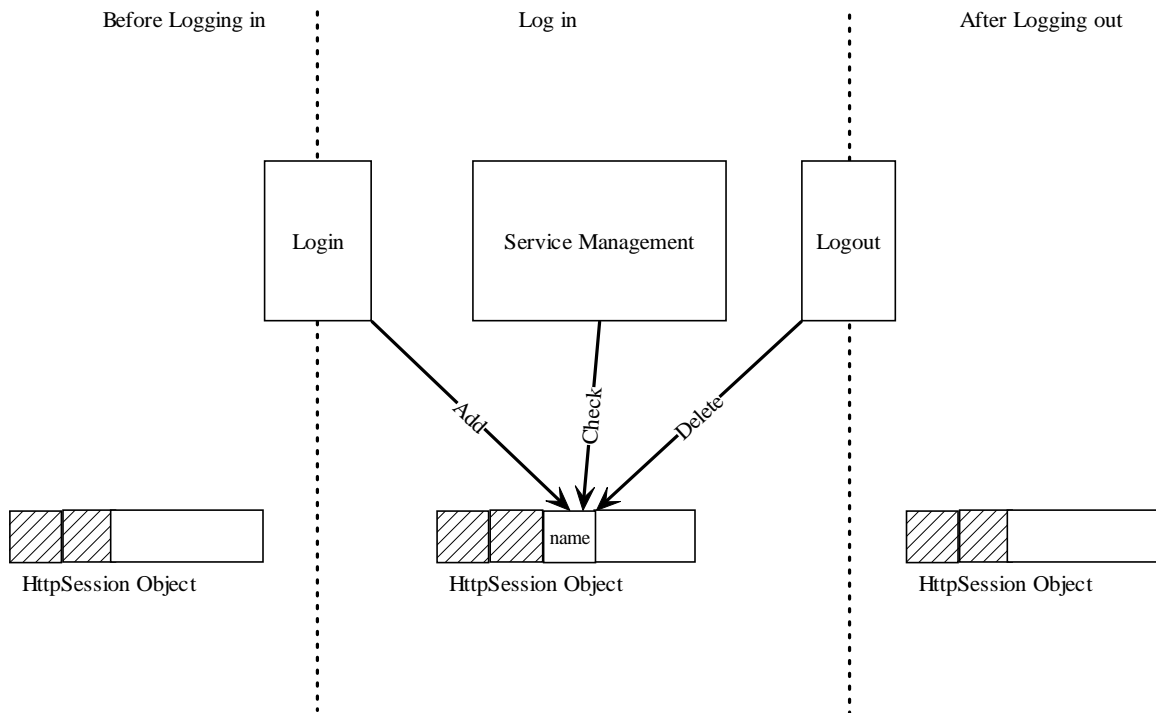


Figure 11: Security Mechanism of GMD Portal Manager

7. Web pages layout

Most pages on the GMD website are generated dynamically by using the Java server pages technology, which is able to combine HTML and java code. The layout of the web pages is illustrated in figure 12.

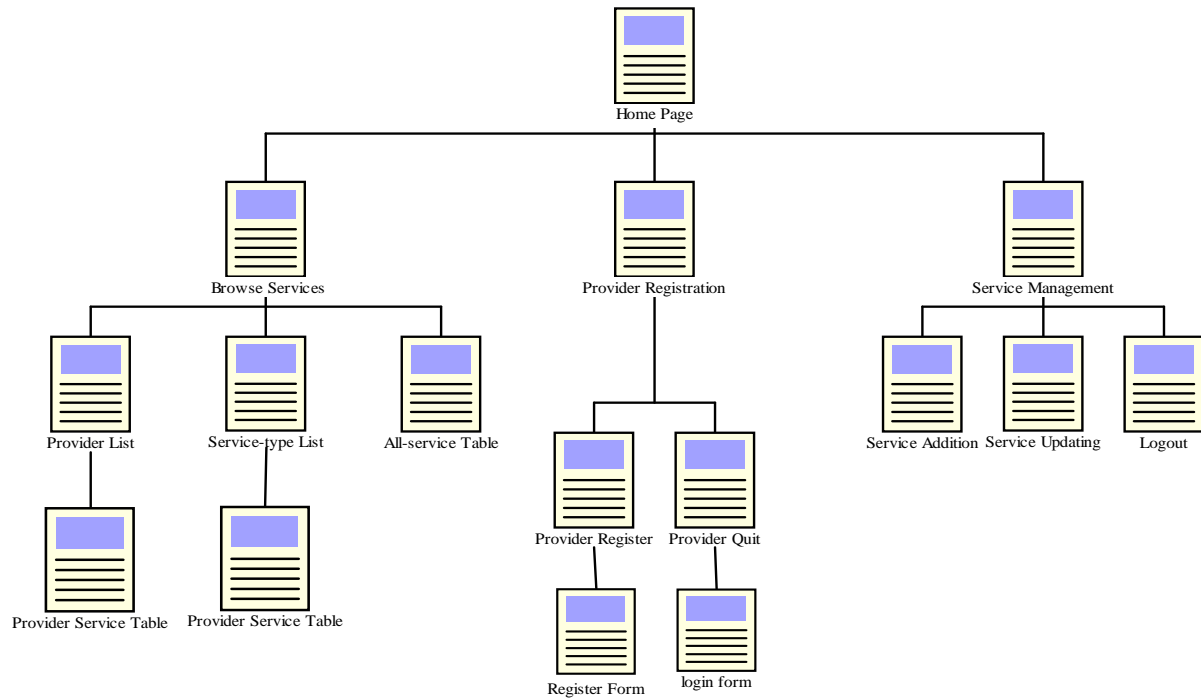


Figure 12: GMD Web page Layout

8. Service Query via the GMD Query Web-service

8.1 SOAP Query Methods

As mentioned earlier, the GMD also accepts SOAP query from any SOAP client, such as a grid resource broker or consumer payment agent. Methods provided by the GMD Query Web-service for remotely SOAP invoking are listed below:

- `queryService()`- get the list with entire service info of all services in the GMD
- `queryServiceByType(String)` –get the list with entire service info of services offered by a certain service type
- `queryServiceByHost(String)` – get the service information associated with a certain host name
- `queryServiceByProvider(String)`-get the list with entire service info of services provided by a certain provider
- `queryServiceContact(String)` – get the list with service name and address of services provided by a specified service type.
- `queryPrice(String)`- get the service price for a specified service.

8.2 Query and Response Message

The GMD Query Web-service converses with its client in XML and these XML messages are encapsulated in SOAP messages. The SOAP engine does not deal with these XML messages but treats them as String. The client invokes a SOAP method with a necessary parameter called query message in this paper. The GMD Query Web-service handles XML message parsing and makes corresponding actions according to the content of the query message. The XML response message will then be sent back to the client by SOAP. The XML building and parsing are required for client programs as well.

8.2.1 Basic elements of response messages

Basic parameters described by response messages are listed below:

- NAME- Service name
- TYPE-Service type
- PROVIDER- Service provider's name
- PRICE-Cost of the service
- HOST(ADDRESS)-The host name of a service
- PATH-The whole path of the program offering a grid service
- DESCRIPTION- The description for a service
- REASON-The reason of a query failure

There are two children elements of *PRICE*.

- HARDWARE-The cost for using service hardware.
- SOFTWARE- The cost for using service software.

In addition, there are several attributes for the root element of response messages.

- STATUS-Status of a query. It has two values "OK" and "ERROR".
- HOST-The name of a host providing a grid service
- TYPE-Service type
- PROVIDER-Service provider's name

8.2.2 Query and Response Messages

- Query-services Message

For getting the list of all services in the GMD, no parameter is required. Clients only invoke method *queryService*. The following is an example response for *queryService*.

```
<?xml version="1.0" encoding="UTF-8"?>
<SERVICES-DETAILS STATUS="OK">
  <SERVICE>
    <NAME>galley</NAME>
    <TYPE>globus</TYPE>
    <PROVIDER>Doshisha University</PROVIDER>
    <PRICE>
      <HARDWARE>2.0</HARDWARE>
```

```

    <SOFTWARE>0.2</SOFTWARE>
  </PRICE>
  <ADDRESS>galley.doshisha.ac.jp</ADDRESS>
  <DESCRIPTION>gridbus testing</DESCRIPTION>
</SERVICE>
<SERVICE>
  <NAME>cabibbo</NAME>
  <TYPE>globus</TYPE>
  <PROVIDER>Sydney University</PROVIDER>
  <PRICE>
    <HARDWARE>3.0</HARDWARE>
    <SOFTWARE>0.3</SOFTWARE>
  </PRICE>
  <ADDRESS>cabibbo.physics.usyd.edu.au</ADDRESS>
  <DESCRIPTION>gridbus testing</DESCRIPTION>
</SERVICE>
<SERVICE>
  <NAME>lem</NAME>
  <TYPE>globus</TYPE>
  <PROVIDER>Melbourne University</PROVIDER>
  <PRICE>
    <HARDWARE>1.0</HARDWARE>
    <SOFTWARE>0.1</SOFTWARE>
  </PRICE>
  <ADDRESS>lem.hp.unimelb.edu.au</ADDRESS>
  <DESCRIPTION>gridbus testing</DESCRIPTION>
</SERVICE>
<SERVICE>
  <NAME>GridbusEF</NAME>
  <TYPE>Earthquake Engineering</TYPE>
  <PROVIDER>Gridbus</PROVIDER>
  <PRICE>
    <HARDWARE>200.0</HARDWARE>
    <SOFTWARE>0.4</SOFTWARE>
  </PRICE>
  <ADDRESS>previn.gridbus.com.au</ADDRESS>
  <DESCRIPTION>earthquake data analyzing</DESCRIPTION>
</SERVICE>
</SERVICES-DETAILS>

```

The attribute *STATUS* of root element *SERVICE-DETAILS* indicates that the query is successful.

- Query-service-by-type Message

In order to get the list of services for a certain service type, the service type should be presented as invoking the SOAP method *queryServiceByType*. The type is indicated in the query message. The following is the format of query-services-by-type message:

```

<?xml version="1.0" encoding="UTF-8"?>
<QUERY-SERVICE>
  <SERVICE_TYPE>...</SERVICE_TYPE>
</QUERY-SERVICE>

```

When the query is successful, the format of the response is shown below:

```

<?xml version="1.0" encoding="UTF-8"?>
<SERVICES-DETAILS TYPE="..." STATUS="OK">
  <SERVICE>
    <NAME>...</NAME>
    <PROVIDER>...</PROVIDER>
    <PRICE>...</PRICE>
    <HARDWARE>...</HARDWARE>
    <SOFTWARE>...</SOFTWARE>
  </SERVICE>

```

```

<ADDRESS>...</ADDRESS>
<DESCRIPTION>...</DESCRIPTION>
</SERVICE>
<SERVICE>
  <NAME>...</NAME>
  <PROVIDER>...</PROVIDER>
  <PRICE>...</PRICE>
    <HARDWARE>...</HARDWARE>
    <SOFTWARE>...</SOFTWARE>
  <ADDRESS>...</ADDRESS>
  <DESCRIPTION>...</DESCRIPTION>
</SERVICE>
</SERVICE-DETAILS>

```

Or when the query is failure, the response message looks like below:

```

<?xml version="1.0" encoding="UTF-8"?>
<SERVICES-DETAILS TYPE="..." STATUS="ERROR">
  <REASON>...</REASON>
</SERVICES-DETAILS>

```

- **Query-service-by-host Message**

After invoking *queryServiceByHost* with query message indicating host name, corresponding service information is sent back.

The format of the query message:

```

<?xml version="1.0" encoding="UTF-8"?>
<QUERY-HOST>
  <HOST_NAME>...</HOST_NAME>
</QUERY-HOST>

```

The format of the successful response message:

```

<?xml version="1.0" encoding="UTF-8"?>
<SERVICE-DETAIL HOST="..." STATUS="OK">
  <NAME>...</NAME>
  <TYPE>...</TYPE>
  <PROVIDER>...</PROVIDER>
  <PRICE>
    <HARDWARE>...</HARDWARE>
    <SOFTWARE>...</SOFTWARE>
  </PRICE>
  <ADDRESS>...</ADDRESS>
  <DESCRIPTION>...</DESCRIPTION>
</SERVICE-DETAIL>

```

- **Query-service-by-provider Message**

Through *queryServiceByProvider*, the list of services provided by a provider whose name is indicated in the query message is sent back.

The format of the query message:

```

<?xml version="1.0" encoding="UTF-8"?>
<QUERY-PROVIDER>
  <PROVIDER_NAME>...</PROVIDER_NAME>
</QUERY-PROVIDER>

```

The format of the successful response is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<SERVICE-DETAIL PROVIDER="..." STATUS="OK">
  <SERVICE>
    <NAME>...</NAME>
    <TYPE>...</TYPE>
    <PROVIDER>...</PROVIDER>
    <PRICE>
      <HARDWARE>...</HARDWARE>
      <SOFTWARE>...</SOFTWARE>
    </PRICE>
    <ADDRESS>...</ADDRESS>
    <DESCRIPTION>...</DESCRIPTION>
  </SERVICE>
  <SERVICE>
    <NAME>...</NAME>
    <TYPE>...</TYPE>
    <PROVIDER>...</PROVIDER>
    <PRICE>
      <HARDWARE>...</HARDWARE>
      <SOFTWARE>...</SOFTWARE>
    </PRICE>
    <ADDRESS>...</ADDRESS>
    <DESCRIPTION>...</DESCRIPTION>
  </SERVICE>
</SERVICE-DETAIL>
```

- Query-service-contact Message

Through *queryServiceContact*, the list with service name and service address of services provided by a provider indicated in the query message is sent back.

The format of the query message:

```
<?xml version="1.0" encoding="UTF-8"?>
<QUERY-SERVICE-CONTACT>
  <SERVICE_TYPE>...</SERVICE_TYPE>
</QUERY-SERVICE-CONTACT>
```

The format of the successful response is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<SERVICE-DETAIL PROVIDER="..." STATUS="OK">
  <SERVICE>
    <NAME>...</NAME>
    <ADDRESS>...</ADDRESS>
  </SERVICE>
  <SERVICE>
    <NAME>...</NAME>
    <ADDRESS>...</ADDRESS>
  </SERVICE>
</SERVICE-DETAIL>
```

- Query-price Message

Through invoking *queryPrice* with query message indicating a service name, price information for the service is sent back.

The format of the query message:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<QUERY-PRICE>
<SERVICE_NAME>...</SERVICE_NAME>
</QUERY-PRICE>
```

The format of the successful response message:

```
<?xml version="1.0" encoding="UTF-8"?>
<SERVICE-DETAIL NAME="..." STATUS="OK">
  <HARDWARE>...</HARDWARE>
  <SOFTWARE>...</SOFTWARE>
</SERVICE-DETAIL>
```

9. Query java client API

Query java client API is also provided along with the GMD, so that consumers only need to invoke methods in their client program without concerning SOAP and XML technologies.

9.1 API

9.1.1 Getting the list of all service in the GMD

```
public ListIterator getServiceListBy()
throws GMDQueryException
```

```
import java.util.ListIterator;
import gridbus.gmd.soap.gmdqueryclient.*;

.
.
.
GMDQuery g=null;

try{
    g=new GMDQuery();
}catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

ListIterator l=null;
GMDService s=null;

try{
    l=g.getServiceList();
}catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(2);
}
do{
    s=(GMDService)l.next();
    System.out.println(s.getName());
    System.out.println(s.getType());
    System.out.println(s.getProvider());
    System.out.println(s.getAddress());
    System.out.println(s.getHPrice());
    System.out.println(s.getSPrice());
    System.out.println(s.getDescription());
}while(l.hasNext());
```

GMDService	
name : String	
type : String	
provider : String	
hPrice : float	
sPrice : float	
address : String	
description : String	
GMDService(String serviceName, String serviceType, String providerName, String addr, float hPrice, float sPrice, String des)	
getName() : String	
getType() : String	
getProvider() : String	
getHPrice() : float	
getSPrice() : float	
getAddress() : String	
getDescription() : String	

9.1.2 Getting a list with all service information for a service type

```
public ListIterator getServiceListByType(String serviceType)
throws GMDQueryException
```

```
import java.util.ListIterator;
import gridbus.gmd.soap.gmdqueryclient.*;

.
.
.
GMDQuery g=null;

try{
    g=new GMDQuery();
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

ListIterator l=null;
GMDService s=null;

try{
    l=g.getServiceList("globus");
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(2);
}
do{
    s=(GMDService)l.next();
    System.out.println(s.getName());
    System.out.println(s.getType());
    System.out.println(s.getProvider());
    System.out.println(s.getAddress());
    System.out.println(s.getHPrice());
    System.out.println(s.getSPrice());
    System.out.println(s.getDescription());
} while(l.hasNext());
```


9.1.3 Getting the list of services provided by a provider

```
public ListIterator getServiceListByProvider(String providerName)
throws GMDQueryException
```

```
import java.util.ListIterator;
import gridbus.gmd.soap.gmdqueryclient.*;

.
.
.
GMDQuery g=null;

try{
    g=new GMDQuery();
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

ListIterator l=null;
GMDService s=null;

try{
    l=g.getServiceListByProvider("Melbourne University");
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(2);
}
do{
    s=(GMDService)l.next();
    System.out.println(s.getName());
    System.out.println(s.getType());
    System.out.println(s.getProvider());
    System.out.println(s.getAddress());
    System.out.println(s.getHPrice());
    System.out.println(s.getSPrice());
    System.out.println(s.getDescription());
}while(l.hasNext());
```

9.1.4 Getting service information for a service host

```
public GMDService getServiceByHost(String hostName)
throws GMDQueryException
```
















```

import gridbus.gmd.soap.gmdqueryclient.*;

GMDQuery g=null;
try{
    g=new GMDQuery();
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

try{
    GMDService s=g.getPrice("pitcairn.mcs.gov.oz.");
    System.out.println(s.getName());
    System.out.println(s.getType());
    System.out.println(s.getProvider());
    System.out.println(s.getAddress());
    System.out.println(s.getHPrice());
    System.out.println(s.getSPrice());
    System.out.println(s.getDescription());
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

```

GMDService	
	name : String
	type : String
	provider : String
	hPrice : float
	sPrice : float
	address : String
	description : String
	GMDService(String serviceName, String serviceType, String providerName, String addr, float hPrice, float sPrice, String des)
	getName() : String
	getType() : String
	getProvider() : String
	getHPrice() : float
	getSPrice() : float
	getAddress() : String
	getDescription() : String

9.1.5 Getting a list with provider name and service host name for a service type

public ListIterator getServiceContactList(String serviceType)
throws GMDQueryException






```
import java.util.ListIterator;
import gridbus.gmd.soap.gmdqueryclient.*;

.
.
.
GMDQuery g=null;

try{
    g=new GMDQuery();
}catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

ListIterator l=null;
ServiceAddress sa=null;

try{
    l=g.getServiceList("globus");
}catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(2);
}
do{
    sa=(ServiceAddress)l.next();
    System.out.println(sa.getName());
    System.out.println(sa.getAddress());
    System.out.println();
}while(l.hasNext());
```

ServiceAddress	
	serviceName : String
	address : String
	ServiceAddress(String serviceName, String address)
	getName() : String
	getAddress() : String

9.1.6 Accessing Raw Compute Power price and Application Service price

```
public Price getPrice(String serviceName)
throws GMDQueryException
```

```
import gridbus.gmd.soap.gmdqueryclient.*;

GMDQuery g=null;
try{
    g=new GMDQuery();
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}

try{
    Price p=g.getPrice("lem");
    System.out.println(p.getHardwarePrice());
    System.out.println(p.getSoftwarePrice());
} catch(GMDQueryException e){
    System.err.println(e.getMessage());
    System.exit(1);
}
```

Price
hardwarePrice : float
softwarePrice : float
Price(float hardwarePrice, float softwarePrice)
getHardwarePrice() : float
getSoftwarePrice() : float

8.2 Jar files for support APIs

```
xerces.jar soap.jar activation.jar mail.jar jdom.jar gmdclient.jar
```

9.3 Property file GMDQuery.prop

(Please in the current directory)

```
#Grid Market Directory host address
gridbus.gmd.GMDHostAddress=jarrett.cs.mu.oz.au

#Grid Market Directory processing port(Default 8080)
gridbus.gmd.GMDPort=8080

#SOAP directory on the host(Default /soap/servlet/rpcrouter)
gridbus.gmd.SOAPDirectory=/soap/servlet/rpcrouter

#the SOAP service name for the grid market directory(Default urn:GMD)
gridbus.gmd.SOAPQueryService=urn:GMD
```

Note: the line with # will be ignored by the program

References:

- [1] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.
- [2] R. Buyya, *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002 (submitted).
- [3] R. Buyya, D. Abramson, and J. Giddy, *A Case for Economy Grid Architecture for Service-Oriented Grid Computing*, Proceedings of the International Parallel and Distributed Processing Symposium: 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), April 23, 2001, San Francisco, California, USA, IEEE CS Press, USA, 2001.
- [4] The Gridbus Project: <http://www.gridbus.org>