Instruction Sheet **Mar 30, 2005**

Creating, building and executing your own grid service

You should have got familiar with building and executing the provided grid services (`MathService` and `AmazonSearchService`). This document walks you through creating your own gridservice by reusing code from the `MathService`.

It also gives some background information about a number of things related to building your own service. Make sure you read those comments.

Throughout the document, wherever you see `<username>` replace it with your actual username (mhvora, etc.)

Unless otherwise mentioned, all paths are relative to the $TUTORIAL_DIR directory.

Basic concepts

A simple grid service, requires the following files from the developer:

- Service implementation class (`MathImpl.java`): This class actually implements the methods that your grid service is supposed to provide.

- Schema definition file (`.gwsdl file`): This XML file specifies the methods that your service will implement in an extremely structured way. Its syntax is very similar to a `.wsdl` file.

- Deployment descriptor (`server-deploy.wsdd`): This file specifies service name (……MathService), service implementation class (`org.globus……impl.MathImpl`) name, PortType class name (`org.globus……MathPortType`) and also schema definition file (`Math_service.wsdl`)

Directory structure and Naming convention

The files that you need are located in two different directory hierarchies.

- The implementation class and deployment descriptor will be present in:
  `org/globus/<username>_progtutorial/services/<service directory>/`

  ```
                                      |
                                      \---> impl/
                                      \---> server-deploy.wsdd
  ```

  This directory will contain a sub-directory impl, that must contain the Implementation class and it also contains the deployment descriptor. If you service is named XyzService, the implementation class is usually named `XyzImpl`.

- The schema definition file (`.gwsdl`) is present inside:
  `schema/<username>_progtutorial/<service directory>`

  There is no strict convention to follow when naming the .gwsdl file.

<service directory> mentioned above could be multiple directories too (especially in the first case). For the simple MathService, the service directory is "core/first". Note that the service directories in the above two cases does not have to be the same.

The build script
build.sh and ANT (using build.xml) do a lot of tasks when you actually build the service. This includes creating build directories, generating and compiling stub classes, compiling the implentation class and packaging files together.

Start creating your own service
We will create a simple service, which will calculate the commission of a salesman, given the employee's sales amount and grade. There will just be a single method:
`public double calcComm(double sales, int grade)`

1. Create the following three directories:

We will place the implementation class and deployment descriptor in:
`org/globus/<username>_progtutorial/services/commission`

We will place the schema definition file in:
`schema/<username>_progtutorial/CommissionService`

The client is present in:
`org/globus/<username>_progtutorial/clients/commission`

2. Copy the MathService files into these directories and rename them. We will then modify these files.

Copy the contents of:
`org/globus/<username>_progtutorial/services/core/first`
to:
`org/globus/<username>_progtutorial/services/commission`
Rename the implementation class from `MathImpl.java` to `CommissionImpl.java`

Copy the contents of:
`schema/<username>_progtutorial/MathService`
to:
`schema/<username>_progtutorial/CommissionService`
Rename the schema definition file from `Math.gwsdl` to `Commission.gwsdl`

Copy the contents of:
`org/globus/<username>_progtutorial/clients/MathService`
to:
`org/globus/<username>_progtutorial/clients/commission`

Modify Implementation class (`CommissionImpl.java`)
As mentioned before, the implemtation class implements all the methods that the service is supposed to provide. What the service "provides" is defined by the `.gwsdl` file.

This class must implement the automatically generated interface `CommissionPortType`. This interface is generated by the build script.

This class may extend `PersistentGridServiceImpl` unless you use Operation Providers (i.e. the Delagation Model)

3. Make modifications in the `CommisionImpl.java`:

a) The name of the package that the implementation class will now change to:
```
package org.globus.<username>_progtutorial.services.commission.impl;
```

b) The class will now import a different PortType:
```
import
    org.globus.<username>_progtutorial.stubs.CommissionService.CommissionPortType;
```

c) Change the name of the class and the interface it implements:
```
public class CommissionImpl extends PersistentGridServiceImpl implements
    CommissionPortType
```

d) Change the name and content of the constructor:
```
public CommissionImpl() {
    super("Simple CommissionService");
}
```

e) Keep only one implementation method as follows:
```
public double calcComm(double sales, int grade) throws RemoteException {
    int commissionPercent;
    switch(grade) {
    case 1:
            commissionPercent = 20;
            break;
    case 2:
            commissionPercent = 15;
            break;
    case 3:
            commissionPercent = 10;
            break;
    case 4:
            commissionPercent = 5;
            break;
    default:
            commissionPercent = 2;
    }
    return (sales * commissionPercent / 100);
}
```

Finally this is how CommissionImpl.java will look like:
```
package org.globus.mhvora_progtutorial.services.commission.impl;

import org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl;
import org.globus.mhvora_progtutorial.stubs.CommissionService.CommissionPortType;
import java.rmi.RemoteException;

public class CommissionImpl extends PersistentGridServiceImpl implements
CommissionPortType
{
        public CommissionImpl()
        {
                super("Simple CommissionService");
        }
```

```
        public double calculateCommission(double sales, int grade) throws RemoteException
        {
                int commissionPercent;

                switch(grade) {
                case 1:
                        commissionPercent = 20;
                        break;
                case 2:
                        commissionPercent = 15;
                        break;
                case 3:
                        commissionPercent = 10;
                        break;
                case 4:
                        commissionPercent = 5;
                        break;
                default:
                        commissionPercent = 2;
                }
                return (sales * commissionPercent / 100);
        }
}
```

Modify schema definition file (`Commission.gwsdl`)
The `gwsdl` file is very similar in syntax to the Web Services `wsdl` file. In fact, for portability, the build script creates an equivalent `wsdl` file based on the `gwsdl` file.

Basically, it is used to define methods that will be implemented by the service. The build script uses this file to generate the PortType interface file.

The `gwsdl` is highly structured. To understand how it is specified, visualize methods as objects and input and output parameters as messages that are passed to and from the method. The message itself carries an object of the "type" of the relevant parameters.

Hence, to define a method, you must first declare the "type" of object that is transported in the message. Then, we must define the two input and output "messages" separately. After that we declare the "operation" (method).

4. Make modifications in `Commission.gwsdl`:

a) The name of the root element, `definitions` must be changed. Also, namespaces, targetnamespace and xmlns:tns needs to be changed. Below is the complete `definitions` tag:

```
<definitions name="CommissionService"
    targetNamespace="http://www.globus.org/namespaces/<username>/CommissionService"
    xmlns:tns="http://www.globus.org/namespaces/<username>/CommissionService"
    xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
    xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
```

b) Also change the targetnamespace in `xsd:schema` tag:

```
<xsd:schema
    targetNamespace="http://www.globus.org/namespaces/mhvora/CommissionService"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns="http://www.w3.org/2001/XMLSchema">
```

c) Remove all `element` tags in the `types` section and insert the following two elements:

```
<xsd:element name="calcComm">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="sales" type="xsd:double"/>
                        <xsd:element name="grade" type="xsd:int"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
<xsd:element name="calcCommResponse">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="commission" type="xsd:double"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

d) Remove all `message` tags and insert the following two messages:

```
<message name="CalcCommInputMessage">
        <part name="parameters" element="tns:calcComm"/>
</message>
<message name="CalcCommOutputMessage">
        <part name="parameters" element="tns:calcCommResponse"/>
</message>
```

e) Remove all `operation` tags in the `portType` section and insert the following `operation`:

```
<operation name="calcComm">
        <input message="tns:CalcCommInputMessage"/>
        <output message="tns:CalcCommOutputMessage"/>
        <fault name="Fault" message="ogsi:FaultMessage"/>
</operation>
```

Finally, after modification, the file will look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="CommissionService"
        targetNamespace="http://www.globus.org/namespaces/mhvora/CommissionService"
        xmlns:tns="http://www.globus.org/namespaces/mhvora/CommissionService"
        xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
        xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../ogsi/ogsi.gwsdl"
        namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>

<types>
<xsd:schema targetNamespace="http://www.globus.org/namespaces/mhvora/CommissionService"
        attributeFormDefault="qualified"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="calcComm">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="sales" type="xsd:double"/>
                                <xsd:element name="grade" type="xsd:int"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="calcCommResponse">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="commission" type="xsd:double"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
</xsd:schema>
</types>

<message name="CalcCommInputMessage">
        <part name="parameters" element="tns:calcComm"/>
</message>
<message name="CalcCommOutputMessage">
        <part name="parameters" element="tns:calcCommResponse"/>
</message>

<gwsdl:portType name="CommissionPortType" extends="ogsi:GridService">
        <operation name="calcComm">
                <input message="tns:CalcCommInputMessage"/>
                <output message="tns:CalcCommOutputMessage"/>
                <fault name="Fault" message="ogsi:FaultMessage"/>
        </operation>
</gwsdl:portType>

</definitions>
```

Modify deployment descriptor (`server-deploy.wsdd`):
The deployment descriptor gets copied into the grid archive (`.gar`) as is. It is used when you deploy your service into Globus to figure out locations of component files. At this point of time, you need to know that the deployment descriptor has the following options that need to be changed:

- Service Name: This is mentioned in the higher-level service element. This is the name that gets reflected in the container that you start up. **Remember that you must have your username in this service name.** The complete service name, includes the server name, port number appended by "ogsa/services". Complete service name for the Commission service would be

  ```
  http://<machine IP>:<port number>/ogsa/services/<username>/CommissionService
  ```

5. Make modifications to `server-deploy.wsdd`

a) Change service name as follows:

```
<service name="<username>/CommissionService" provider="Handler" style="wrapped">
```

- Name: This is a name of the service.

b) Change name as follows:

```
<parameter name="name" value="CommissionService"/>
```

- Base Class Name: This needs to point to the Implementation Class.

c) Change baseClassName as follows:

```
<parameter name="baseClassName"
value="org.globus.<username>_progtutorial.services.commission.impl.CommissionImpl"/>
```

- Class Name: This needs to point to the PortType Interface.

d) Change className as follows:

```
<parameter name="className"
value="org.globus.<username>_progtutorial.stubs.CommissionService.CommissionPortType"
/>
```

- Schema Path: This needs to point to the schema definition file (.wsdl) format. This file is generated by the build script for you.

e) Change className as follows:

```
<parameter name="schemaPath"
value="schema/<username>_progtutorial/CommissionService/Commission_service.wsdl"/>
```

Finally, after modification, `server-deploy.wsdd` will look as follows:

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

    <service name="mhvora_progtutorial/core/first/MathService" provider="Handler"
        style="wrapped">
    <parameter name="name" value="CommissionService"/>
    <parameter name="baseClassName"
        value="org.globus.mhvora_progtutorial.services.commission.impl.CommissionImpl"/>
    <parameter name="className"
      value="org.globus.mhvora_progtutorial.stubs.CommissionService.CommissionPortType"/>
    <parameter name="schemaPath"
      value="schema/mhvora_progtutorial/CommissionService/Commission_service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*"/>
    <parameter name="persistent" value="true"/>
    <parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIProvider"/>
</service>
```

Modify namespace2package.mappings

This file is used to map from XML namespaces mentioned in the schema definition file to corresponding Java packages

6. Add the following lines of code to the namespace2package.mappings file:

```
http\://www.globus.org/namespaces/<username>/CommissionService=org.globus.<usern
ame>_progtutorial.stubs.CommissionService
http\://www.globus.org/namespaces/<username>/CommissionService/bindings=org.glob
us.<username>_progtutorial.stubs.CommissionService.bindings
http\://www.globus.org/namespaces/<username>/CommissionService/service=org.globu
s.<username>_progtutorial.stubs.CommissionService.service
```

Building the service

The service is now ready to be built and deployed into the container. To build the service, you need to run the build.sh script. This script in turn starts up ANT. ANT refers to the build.xml file to carry out all the tasks required to build and package the service.

The build script takes two parameters:

- The base directory of the service implementation
  `org/globus/<username>_progtutorial/services/commission`
- The schema definition file (.gwsdl)
  `schema/<username>_progtutorial/CommissionService/Commission.gwsdl`

7. Execute build.sh

Execute “ant clean” and then execute the following:

```
./build.sh org/globus/<username>_progtutorial/services/commission
      schema/<username>_progtutorial/CommissionService/Commission.gwsdl
```

Exploring the `build` directory

The build script causes a number of things to be generated. All these are present inside the `build` directory inside `$TUTORIAL_DIR`.

```
build
|       classes
|       \---    org
|               |       globus
|               |       \---    <username>_progtutorial
|               |               |       services
|               |               |       \---    commission
|               |               |               \---    impl
|               |               |                       \---    CommissionImpl.class
|               |               \---    stubs
|               |                       \---    CommissionService
|               |                               |       CommissionPortType.class
|               |                               |       _calcComm.class
|               |                               |       _calcCommResponse.class
|               |                               |       bindings
|               |                               |       \---    CommissionServiceSOAPBindingStub.class
|               |                               \---    service
|               |                                       |       CommissionService.class
|               |                                       |       CommissionServiceLocator.class
|               |                                       \---    CommissionServiceGridLocator.class
|               \---    gridforum
|                       \---    ogsi
|       lib
|       |       <username>_progtutorial_CommissionService-stub.jar
|       |       org.globus.<username>_progtutorial.services.commission.jar
|       \---    org_globus_<username>_progtutorial_services_commission.gar
|       schema
|       |       NStoPkg.properties
|       |       <username>_progtutorial
|       |       |       CommissionService
|       |       |       |       Commission.gwsdl
|       |       |       |       Commission.wsdl
|       |       |       |       Commission.xsd
|       |       |       |       Commission_bindings.wsdl
|       |       |       \---    Commission_service.wsdl
|       \---    ogsi
\---    stubs
        \---    org
                |       globus
                |       \---    <username>_progtutorial
                |               \---    stubs
                |                       \---    CommissionService
                |                               |       CommissionPortType.java
                |                               |       _calcComm.java
                |                               |       _calcCommResponse.java
                |                               |       bindings
                |                               |       \---    CommissionServiceSOAPBindingStub.java
                |                               \---    service
                |                                       |       CommissionService.java
                |                                       |       CommissionServiceLocator.java
                |                                       \---    CommissionServiceGridLocator.java
                \---    gridforum
                        \---    ogsi
```

- The classes subdirectory contain the compiled classes.
- The stubs subdirectory contains the source code of the stub files.
- The lib subdirectory contains the packaged jar and gar files.
- The schema subdirectory contains schema definition files

You should look at the contents of the gar and jar packages in the lib directory by using:
`jar tf <gar-or-jar-name>`

Deploy the service
Copy this generated gar file to either of the gar repositories, depending on which machine you are working on.

8. Copy gar to gar repository
```
cp build/lib/org_globus_<username>_progtutorial_services_commission.gar
      /home/csgrad/sjlobo/<machine>gars
```
Replace `<machine>` with either `cerf / mills / vixie.`

Setup CLASSPATH
After you copy the gar to the gar repository, it takes a while to deploy the service into Globus. You will get an e-mail notifying you once the deploy is complete.

As part of the process of deploying the service, Globus copies jar files (that are present inside your gar) into `$GLOBUS_LOCATION/lib`. For your client to be able to use your service, your system classpath must include these new jar files. To do this you need to source the following file:

9. Source setenv.csh
```
source $GLOBUS_LOCATION/setenv.csh
```

Start container

9. Start container (if required also do `grid-proxy-init`)
```
ant startContainer -Dservice.port=5956
```

Create, compile and execute client
Now you can go ahead and compile and execute your client.

10. Write client (CommissionClient.java): It would be as follows:
```
package org.globus.mhvora_progtutorial.clients.commission;

import org.globus.mhvora_progtutorial.stubs.CommissionService.service.CommissionServiceGridLocator;
import org.globus.mhvora_progtutorial.stubs.CommissionService.CommissionPortType;

import java.net.URL;

public class CommissionClient
{
        public static void main(String[] args)
        {
                try
                {
                        // Get command-line arguments
                        URL GSH = new java.net.URL(args[0]);
                        double sales = Double.parseDouble(args[1]);
                        int grade = Integer.parseInt(args[2]);
                        // Get a reference to the CommissionService instance
                        CommissionServiceGridLocator commissionServiceLocator = new
CommissionServiceGridLocator();
                        CommissionPortType commission =
commissionServiceLocator.getCommissionServicePort(GSH);
```

```
                        // Call remote method 'calcComm'
                        double commissionAmount = commission.calcComm(sales, grade);
                        System.out.println("Commission: " + commissionAmount);
                }catch(Exception e)
                {
                        System.out.println("ERROR!");
                        e.printStackTrace();
                }
        }
}
```

11. Compile client

```
javac org/globus/<username>_progtutorial/clients/commission/*.java
```

12. Execute client

```
java org/globus/<username>_progtutorial/clients/commission/CommissionClient
   http://localhost:<port>/ogsa/services/<username>/CommissionService 10000 2
```

The output should be 1500.