Assignment 5
Installing the GT4 Core
and
Creating, deploying, and testing a simple GT4 grid service
(Windows version)

B. Wilkinson, Clayton Ferner, Jasper Land, and Jeremy Villalobos
March 26, 2010

Acknowledgements: This assignment is based on the book: *Globus Toolkit 4 Programming Java Services* by Borja Sotomayor and Lisa Childers [1]. All the code comes from this book. The book is based upon the on-line tutorial available at [http://gdp.globus.org/gt4-tutorial/](http://gdp.globus.org/gt4-tutorial/). Also certain parts have been developed directly from [2].

1. Overview

The goal of this assignment is to create a Globus 4.0 (GT4) stateful Web service and deploy it in GT4 container, and test it with a simple client. All the code to achieve this goal is provided but you will need to understand the code. The last part of the assignment requires you to modify the service to perform an additional operation.

Systems Problem and Approach Taken: Web services are deployed within a service “container” as described in the lecture notes. To be able to do the assignment, it is necessary to have access to GT4 “core” software and included GT4 container. The full suite of GT4 software, including the core, is installed on four Grid computing servers, coit-grid01.uncc.edu – coit-grid04.uncc.edu and it is possible for users to start a separate container for themselves on one of these servers. However, each container has a rather large footprint and it is impractical to have a large number of separate containers running simultaneously (Operating system thrashing occurs). Also when a container is started, it will deploy all available services. Hence one would see the deployable services of all students within your container. Each service must have a unique name. In the Fall 2005 Grid computing course, a script was written to automatically rename students’ services but still problems can occur that cause all work in the whole class to be held up. Hence a different approach is now used, that of installing GT4 core on your own computer. You will actually learn more with this approach, and the software development becomes easier. (You do not need an Internet connection after the software is installed and you can use any installed editor.) It also enables us to do other more advanced assignments on services easily.

*Very important points are given in red. If you are still having problems, also consult FAQ’s on the course home page.*
2. Installing the GT 4 Core and Associated Software

The Java version of the core will be used, which can be installed on Windows, Linux or a mac. The following instructions are for a Windows system. For other systems, see the home page of the software for installation.

(i) Supporting required software: Globus 4.0 requires:

- JDK 1.4.2+ Download from http://java.sun.com/j2se and install if you do not have JDK.
- Ant 1.6.1+ Download from http://jakarta.apache.org/ant and install if you do not have ant.

This Windows version of this assignment also requires for building:

- Python 2.4+ Download from http://www.python.org/ and install if you do not have Python.

The versions that were used for testing this assignment by Saurav Bhattarai in February 2010 were JDK 1.6.0_18, ANT 1.6.5, and Python 2.6.4.¹ We strongly recommend that you use these software versions. There may be compatibility issues with other versions. Known incompatibilities will be posted on FAQ page.

(ii) Globus Core

Next install the Globus 4.0 core from http://www.globus.org/toolkit/downloads. Choose Globus 4.0.8 and the Java WS Core binary installer (NOT Globus 4.2²). Unzip (“extract all”) into the directory ws-core-4.0.8, which holds /bin etc:

---

¹ It was also tested in 2008 by BW JDK 1.5.0_11, ANT 1.6.5, and Python 2.4.
² The most recent version of Globus 4.0 as of Sept 2008 is version 4.0.8. Globus 4.2.0 has been released but we have not tested the assignment with Globus 4.2.0.
Move **ws-core-4.0.8** to main drive say C.

(iii) **Environment variables:** Create systems environment variables³:

- **GLOBUS_LOCATION** Set to the location of the Globus binary, e.g. C:\**ws-core-4.0.8**
- **ANT_HOME** Set to C:\**apache-ant-1.6.5** (or where ever you installed ant)
- **JAVA_HOME** Set to the location of the Java installation (something like C:\Program Files\Java\jdk1.6.0_18)

It is very important that the above are system variables, not user variables. If JAVA_HOME exists as a user variable (e.g. for the JRE installer), delete it and re-create it as a system variable.

Also set system path:

**PATH**

To include locations of Globus, and Ant, Python, and Java, e.g.: %GLOBUS_LOCATION%\bin;%ANT_HOME%\bin;C:\Python24;%JAVA_HOME%\bin

Paths are separated by semicolons. Environment variable names are surrounded with %’s

**BE VERY CAREFUL NOT TO ALTER EXISTING PATHS ON YOUR COMPUTER. WE WILL NOT RESPONSIBLE IF YOU DO NOT THIS STEP PROPERLY AND CAUSE PROBLEMS TO YOUR COMPUTER OPERATION. (If it worries you, set a Windows checkpoint first.)**

³ To set/edit an environment variable in Windows, goto **Start**, right-click **My Computer**, Go down menu and click on **Properties**. Click on **Advanced** tab. Click on **Environment variables** tab. Click **new**, **edit**, or **delete** as appropriate.
3. Command Prompt Windows

After setting the environmental variables, you will need to close all Command (DOS prompt) windows and reopen them. (You should not need to reboot.) In the following, you will need have at least two Command windows open:

1) To start/stop the container as well as deploy/undeploy services, and
2) To compile/run your client program.

We will call the first window the “Container Window” and the second the “Client Window”. Open up the two command prompt windows. Remember that these are DOS command windows. Use DOS command dir to list the contents of a directory (not the Linux ls command). Changing directories is the same (cd).

4. Testing your Software Installation

From the Container Window, start the container with the command:

```
globus-start-container -nosec
```

The -nosec flag indicate that a non-secure container will be started. The server will listen on port 8080. Starting a container usually takes under a minute. Once started, you should get a listing of the deployed services, for example:

```
C:\Documents and Settings\Barry Wilkinson.BARRY>globus-start-container -nosec
Starting SOAP server at: http://166.82.138.77:8080/warf/services/
With the following services:
[1]: http://166.82.138.77:8080/warf/services/AdminService
[2]: http://166.82.138.77:8080/warf/services/AuthCalloutTextService
[3]: http://166.82.138.77:8080/warf/services/ContainerRegistryEntryService
[4]: http://166.82.138.77:8080/warf/services/ContainerRegistryService
[5]: http://166.82.138.77:8080/warf/services/CounterService
[6]: http://166.82.138.77:8080/warf/services/ManagementService
[7]: http://166.82.138.77:8080/warf/services/NotificationConsumerFactoryService
[8]: http://166.82.138.77:8080/warf/services/NotificationConsumerService
[9]: http://166.82.138.77:8080/warf/services/NotificationTextService
[10]: http://166.82.138.77:8080/warf/services/PersistenceTestSubscriptionManager
[11]: http://166.82.138.77:8080/warf/services/SampleAuthzService
[12]: http://166.82.138.77:8080/warf/services/SecureCounterService
[13]: http://166.82.138.77:8080/warf/services/ShutdownService
[14]: http://166.82.138.77:8080/warf/services/SubscriptionManagerService
[15]: http://166.82.138.77:8080/warf/services/TestRFCService
[16]: http://166.82.138.77:8080/warf/services/TestService
[17]: http://166.82.138.77:8080/warf/services/TestServiceRequest
[18]: http://166.82.138.77:8080/warf/services/TestServiceWrongWSDL
[19]: http://166.82.138.77:8080/warf/services/Version
[20]: http://166.82.138.77:8080/warf/services/WidgetNotificationService
[21]: http://166.82.138.77:8080/warf/services/WidowsClientService
```

4 To open a command prompt, goto start -> All Programs -> Accessories and select Command Prompt. Alternatively, select start -> run and enter cmd.
In this example, the host computer has the Internet address 166.82.130.77. Without an external Internet connection, you will get the local host address (e.g. 192.168.123.100…) or a local network address (e.g. 127.0.0.1). Notice that while the container is running, the command prompt window is not usable for other tasks. One can run the container in the background, but it is more convenient to open a second command prompt window for starting client programs.

You can stop the container with control-C:

You will need to restart the container later.

5. Creating, Deploying, and Testing a simple GT4 Grid Service

The next task to deploy a service into the container. The service for this task will be a simple Math service that can perform basic arithmetic for a client.

The Math service will access a resource with two properties:

(a) An integer value that can be operated upon by the service
(b) A string values that holds string describing the last operation

The service itself will have three remotely accessible operations that operate upon value:

(a) add, that adds a to the resource property value.
(b) subtract that subtracts a from the resource property value.
(c) getValueRP that returns the current value of value.

Usually, the best way for any programming task is to begin with an overall description of what you want the code to do, which in this case is the service interface. The service interface describes how what the service provides in terms of names of operations, their arguments and return values. A Java interface for our service is:

```java
public interface Math {
    public void add(int a);
    public void subtract(int a);
    public int getValueRP();
}
```
It is possible to start with this interface and create the necessary WSDL file using the standard Web service tool called Java2WSDL. However, the WSDL file for GT 4 has to include details of resource properties that are not given explicitly in the interface above. Hence, we will provide the WSDL file.

**Step 1 Getting the Files**

All the required files are provided and comes directly from [1]. The MathService source code files can be found from [http://www.gt4book.com](http://www.gt4book.com) (http://www.gt4book.com/downloads/gt4book-examples.tar.gz) A Windows zip compressed version can be found at [http://www.cs.uncc.edu/~abw/ITCS4146S07/gt4book-examples.zip](http://www.cs.uncc.edu/~abw/ITCS4146S07/gt4book-examples.zip). Download and uncompress the file into a directory called **GT4services**. Everything is included (the java source WSDL and deployment files, etc.):

![GT4services](image)

**WSDL service interface description file** -- The WSDL service interface description file is provided within the GT4services folder at:

**GT4Services\schema\examples\MathService_instance\Math.wsdl**

This file, and discussion of its contents, can be found in Appendix A. Later on we will need to modify this file, but first we will use the existing contents that describe the Math service above.

**Service code in Java** -- For this assignment, both the code for service operations and for the resource properties are put in the same class for convenience. More complex services and resources would be defined in separate classes. The Java code for the service and its resource properties is located within the **GT4services** folder at:

**GT4services\org\globus\examples\services\core\first\impl\MathService.java**

**Deployment Descriptor** -- The deployment descriptor gives several different important sets of information about the service once it is deployed. It is located within the **GT4services** folder at:

6
GT4services\org\globus\examples\services\core\first\deploy-server.wsdd.

**Step 2 – Building the Math Service**

It is now necessary to package all the required files into a GAR (Grid Archive) file. The build tool ant from the Apache Software Foundation is used to achieve this as shown overleaf:

![Diagram of build process](http://gdp.globus.org/gt4-tutorial/multiplehtml/ch03s04.html)

Generating a GAR file with Ant (from [http://gdp.globus.org/gt4-tutorial/multiplehtml/ch03s04.html](http://gdp.globus.org/gt4-tutorial/multiplehtml/ch03s04.html))

Ant is similar in concept to the Unix make tool but a java tool and XML based. Build scripts are provided by Globus 4 to use the ant build file. The windows version of the build script for MathService is the Python file called `globus-build-service.py`, which held in the **GT4services** directory. The build script takes one argument, the name of your service that you want to deploy. To keep with the naming convention in [1], this service will be called **first**.

In the **Client Window**, run the build script from the **GT4services** directory with:

`globus-build-service.py first`

The output should look similar to the following:

**Buildfile: build.xml**

```xml
.
.
.
```
BUILD SUCCESSFUL
Total time: 8 seconds

During the build process, a new directory is created in your GT4Services directory that is named build. All of your stubs and class files that were generated will be in that directory and its subdirectories. More importantly, there is a GAR (Grid Archive) file called org.globus.examples.services.core_first.gar. The GAR file is the package that contains every file that is needed to successfully deploy your Math Service into the Globus container. The files contained in the GAR file are the Java class files, WSDL, compiled stubs, and the deployment descriptor.

**Step 3 – Deploying the Math Service**

If the container is still running in the Container Window, then stop it using Control-C. To deploy the Math Service, you will use a tool provided by the Globus Toolkit called globus-deploy-gar. In the Container Window, issue the command:

```
globus-deploy-gar org.globus.examples.services.core_first.gar
```

Successful output of the command is:

![Command Output]

The service has now been deployed.

Check service is deployed by starting container from the Container Window:
You should see the service called **MathService**.

**Step 4 – Compiling the Client**

A client has already been provided to test the Math Service and is located in the **GT4Services** directory at:

**GT4Services\org\globus\examples\clients\MathService_instance\Client.java**

and contains the following code:

```java
package org.globus.examples.clients.MathService_instance;
import org.apache.axis.message.addressing.Address;
import org.apache.axis.message.addressing.EndpointReferenceType;
import org.globus.examples.stubs.MathService_instance.MathPortType;
import org.globus.examples.stubs.MathService_instance.service.MathServiceAddressingLocator;
public class Client {
    public static void main(String[] args) {
        MathServiceAddressingLocator locator = new MathServiceAddressingLocator();
        try {
            String serviceURI = args[0];
            // Create endpoint reference to service
            EndpointReferenceType endpoint = new
```
EndpointReferenceType();
endpoint.setAddress(new Address(serviceURI));

MathPortType math;

// Get PortType
math = locator.getMathPortTypePort(endpoint);

// Perform an addition
math.add(10);

// Perform another addition
math.add(5);

// Access value
System.out.println("Current value: "
   + math.getValueRP(new GetValueRP()));

// Perform a subtraction
math.subtract(5);

// Access value
System.out.println("Current value: "
   + math.getValueRP(new GetValueRP()));
}
} catch (Exception e) {
   e.printStackTrace();
}

When the client is run from the command line, you pass it one argument. The argument is the URL that specifies where the service resides. The client will create the end point reference and incorporate this URL as the address. The end point reference is then used with the getMathPortTypePort method of a MathServiceAdressingLocator object to obtain a reference to the Math interface (portType). Then, we can apply the methods available in the service as though they were local methods. Notice that the call to the service (add and subtract method calls) must be in a “try {} catch {}” block because a “RemoteException” may be thrown. The code for the “MathServiceAddressingLocator” is created during the build process. (Thus you don’t have to write it!)

(a) Setting the Classpath

To compile the new client, you will need the JAR files from the Globus toolkit in your CLASSPATH. Do this by executing the following command in the Client Window:

%GLOBUS_LOCATION%\etc\globus-devel-env.bat

You can verify that this sets your CLASSPATH, by executing the command:

echo %CLASSPATH%

You should see a long list of JAR files.
Running `\gt4\etc\globus-devel-env.bat` only needs to be done once for each Client Window that you open. It does not need to be done each time you compile.

**(b) Compiling Client**

Once your CLASSPATH has been set, then you can compile the Client code by typing in the following command:

```
javac -classpath build\classes\org\globus\examples\services\core\first\impl\:%CLASSPATH% org\globus\examples\clients\MathService_instance\Client.java
```

**Step 5 – Start the Container for your Service**

Restart the Globus container from the Container Window with:

```
globus-start-container -nosec
```

if the container is not running.

**Step 6 – Run the Client**

To start the client from your GT4Services directory, do the following in the Client Window, which passes the GSH of the service as an argument:

```
java -classpath build\classes\org\globus\examples\services\core\first\impl\:%CLASSPATH% org.globus.examples.clients.MathService_instance.Client http://localhost:8080/wsrf/services/examples/core/first/MathService
```

which should give the output:

```
Current value: 15
Current value: 10
```

**Step 7 – Undeploy the Math Service and Kill a Container**

Before we can add functionality to the Math Service (Section 5), we must undeploy the service. In the Container Window, kill the container with a Control-C. Then to undeploy the service, type in the following command:

```
globus-undeploy-gar org_globus_examples_services_core_first
```

which should result with the following output:

Undeploying gar...

Deleting /.
6. Adding Functionality to the Math Service

In this final task, you are asked to modify the Math service and associated files so the service supports the multiplication operation. To do this task, you will need to modify:

- Service code (**MathService.java**)
- WSDL file (**Math.wsdl**)

The exact changes that are necessary are not given. You are to work them out yourself. You will need to fully understand the contents of service code and WSDL files and then modify them accordingly. Appendix A gives an explanation of the important parts of these files.

Keep all file names the same and simply redeploy the service afterwards. You will also need to add a code to the client code (**Client.java**) to test the modified service to include multiplication.

7. Challenge Step for Graduate Students
   (Extra credit for undergraduate students)

Rename the Math service and re-deploy the service with the new name and test it.

8. Assignment Submission

How to submit your assignment is described in a separate document

Submit a PDF document by the due date posted on the course home page. Include:

- Code where applicable, with comments
- Brief explanation of code.
- Show that you successfully followed the instructions and performs all tasks.
- Take screen shots and include these screen shots in the document. Number of screen shots is up to you but it must demonstrate your programs worked.
- Conclusions
- For the record, state what programming skills and knowledge you used but already knew before doing this assignment. This information may be helpful in subsequent course developments. No information you provide will reduce your grade, but extra credit might be given for very useful comments.
References


Footnote about “Globus” user

Ideally, operational tasks on the container such as starting the container, deploying services, and stopping the container, should only be done by a separate non-root user, typically named “globus” user, which is different to the regular user. The regular user would create the services, and create and execute the clients. Borja Sotomayor and Lisa Childers [1] point out on page 433 that this is what would happen in a real situation when the container is on one machine and the users on other machines. Also flaws in code may not present themselves without the Globus user. We are not asking for you to create and use globus user for simplicity.
APPENDIX A
Explanations for WSDL file, service code, and WSDD file

1. WSDL file
GT4Services\schema\examples\MathService_instance\Math.wsdl

Contents:

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MathService"

targetNamespace="http://www.globus.org/namespaces/examples/core/MathService_instance"
    xmlns="http://schemas.xmlsoap.org/wsdl/
    xmlns:tns="http://www.globus.org/namespaces/examples/core/MathService_instance"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/
    xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-
    01.xsd"
    xmlns:wsrpw="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-
    01.wsdl"
    xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreprocessor"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!--============================================================
T Y P E S
============================================================-->

<T Y P E S

<!-- REQUESTS AND RESPONSES -->

<xsd:element name="add" type="xsd:int"/>
<xsd:element name="addResponse">
    <xsd:complexType/>
</xsd:element>

<xsd:element name="subtract" type="xsd:int"/>
<xsd:element name="subtractResponse">
    <xsd:complexType/>
</xsd:element>

<xsd:element name="getValueRP">
    <xsd:complexType>
<xsd:element>
<xsd:element name="getValueRPResponse" type="xsd:int"/>

<!-- RESOURCE PROPERTIES -->
<xsd:element name="Value" type="xsd:int"/>
<xsd:element name="LastOp" type="xsd:string"/>

<xsd:element name="MathResourceProperties">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="tns:Value" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="tns:LastOp" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

</xsd:schema>
</types>

<!--============================================================
M E S S A G E S
============================================================-->
<message name="AddInputMessage">
    <part name="parameters" element="tns:add"/>
</message>
<message name="AddOutputMessage">
    <part name="parameters" element="tns:addResponse"/>
</message>

<message name="SubtractInputMessage">
    <part name="parameters" element="tns:subtract"/>
</message>
<message name="SubtractOutputMessage">
    <part name="parameters" element="tns:subtractResponse"/>
</message>

<message name="GetValueRPInputMessage">
    <part name="parameters" element="tns:getValueRP"/>
</message>
<message name="GetValueRPOutputMessage">
    <part name="parameters" element="tns:getValueRPResponse"/>
</message>

<!--============================================================
P O R T T Y P E
============================================================-->
<portType name="MathPortType">
    wsdlpp:extends="wsrpw:GetResourceProperty"
    wsrp:ResourceProperties="tns:MathResourceProperties"
<operation name="add">
  <input message="tns:AddInputMessage"/>
  <output message="tns:AddOutputMessage"/>
</operation>

<operation name="subtract">
  <input message="tns:SubtractInputMessage"/>
  <output message="tns:SubtractOutputMessage"/>
</operation>

<operation name="getValueRP">
  <input message="tns:GetValueRPInputMessage"/>
  <output message="tns:GetValueRPOutputMessage"/>
</operation>

</portType>
</definitions>

EXPLANATIONS (from reference [2])

The first bolded section of importance looks like the following:

<definitions name="MathService"

targetNamespace="http://www.globus.org/namespaces/examples/core/MathService_instance"

The first line sets the name of the service that we are trying to implement, in this case, MathService. The second name specifies the namespace for this definition. Note that the URL given, http://www.globus.org/namespaces/examples/core/MathService_instance, does not exist on the Globus website.

The next bolded section of importance looks like the following:

<xsd:element name="add" type="xsd:int"/>
  <xsd:complexType/>
</xsd:element>

The first line says that there will be a method called ‘add’ that takes a parameter of type int. The next line sets the return type of the add method, which is a complex type, not a primitive type.

The next bolded section, shown below, sets the message types that will be passed.

<message name="AddInputMessage">
  <part name="parameters" element="tns:add"/>
</message>
<message name="AddOutputMessage">
  <part name="parameters" element="tns:addResponse"/>
Notice that there are input and output messages for the add method.

The next section:

```xml
<portType name="MathPortType"
    wsdlpp:extends="wsrcw:GetResourceProperty"
    wsrp:ResourceProperties="tns:MathResourceProperties">

    <operation name="add">
        <input message="tns:AddInputMessage"/>
        <output message="tns:AddOutputMessage"/>
    </operation>

    <operation name="subtract">
        <input message="tns:SubtractInputMessage"/>
        <output message="tns:SubtractOutputMessage"/>
    </operation>

    <operation name="getValueRP">
        <input message="tns:GetValueRPInputMessage"/>
        <output message="tns:GetValueRPOutputMessage"/>
    </operation>

</portType>
```

ties everything together by providing the messages that should occur on each operation. Each method must have all of the similar entries as shown for the ‘add method’.

For more general information on the structure of WSDL files, visit the W3C website at http://www.w3.org/TR/wsdl. The WSDL file here is tailored to GT4 requirements. The WSDL file for a GT4 service does not have any binding entries such as usually present in the WSDL files because these bindings are rather special and will be created automatically by a GT4 tool used when the service is built. Also note that the WSDL file contains reference to a MathQName class. This class is described in [1] and is present to simplify referring to the services’s qualified names.

2. **Service code in Java**

`GT4services\org\globus\examples\services\core\first\impl\MathService.java`

The contents of the file is:

```java
package org.globus.examples.services.core.first.impl;
import java.rmi.RemoteException;
import org.globus.wsrf.Resource;
import org.globus.wsrf.ResourceProperties;
import org.globus.wsrf.ResourceProperty;
import org.globus.wsrf.impl.ReflectionResourceProperty;
```
import org.globus.wsrf.impl.SimpleResourcePropertySet;
import org.globus.examples.stubs.MathService_instance.AddResponse;
import org.globus.examples.stubs.MathService_instance.SubtractResponse;
import org.globus.examples.stubs.MathService_instance.GetValueRP;

public class MathService implements Resource, ResourceProperties {

    /* Resource Property set */
    private ResourcePropertySet propSet;

    /* Resource properties */
    private int value;
    private String lastOp;

    /* Constructor. Initializes RPs */
    public MathService() throws RemoteException {
        /* Create RP set */
        this.propSet = new SimpleResourcePropertySet(
            MathQNames.RESOURCE_PROPERTIES);

        /* Initialize the RP's */
        try {
            ResourceProperty valueRP = new
                ReflectionResourceProperty(
                    MathQNames.RP_VALUE, "Value", this);
            this.propSet.add(valueRP);
            setValue(0);

            ResourceProperty lastOpRP = new
                ReflectionResourceProperty(
                    MathQNames.RP_LASTOP, "LastOp", this);
            this.propSet.add(lastOpRP);
            setLastOp("NONE");
        } catch (Exception e) {
            throw new RuntimeException(e.getMessage());
        }
    }

    /* Get/Setters for the RPs */
    public int getValue() {
        return value;
    }

    public void setValue(int value) {
        this.value = value;
    }

    public String getLastOp() {
        return lastOp;
    }

    public void setLastOp(String lastOp) {
        this.lastOp = lastOp;
    }

    /* Remotely-accessible operations */

public AddResponse add(int a) throws RemoteException {
    value += a;
    lastOp = "ADDITION";
    return new AddResponse();
}

public SubtractResponse subtract(int a) throws RemoteException {
    value -= a;
    lastOp = "SUBTRACTION";
    return new SubtractResponse();
}

public int getValueRP(GetValueRP params) throws RemoteException {
    return value;
}

/* Required by interface ResourceProperties */
public ResourcePropertySet getResourcePropertySet() {
    return this.propSet;
}
}

**EXPLANATION (from reference [2])**

As you can see, MathService implements Resource and ResourceProperties. Both of these classes are part of the org.globus.wsrf package that Globus Toolkit 4.0 provides. The primary method we are interested is the add method, which will be used by the clients. The constructor and the methods must throw a RemoteException because the services use Java Remote Method Invocation (RMI) behind the scenes.

3. Deployment Descriptor

**GT4services\org\globus\examples\services\core\first\deploy-server.wsdd.**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<deployment name="defaultServerConfig"
    xmlns="http://xml.apache.org/axis/wsdd/"
    xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <service name="examples/core/first/MathService" provider="Handler" use="literal" style="document">
        <parameter name="className" value="org.globus.examples.services.core.first.impl.MathService"/>
        <wsdlFile>share/schema/examples/MathService_instance/Math_service.wsdl</wsdlFile>
        <parameter name="allowedMethods" value="*"/>
        <parameter name="handlerClass" value="org.globus.axis.providers.RPCProvider"/>
        <parameter name="scope" value="Application"/>
        <parameter name="providers" value="GetRPProvider"/>
        <parameter name="loadOnStartup" value="true"/>
    </service>

```
EXPLANATION (from reference [2])

The line that looks like the following specifies where the service will be located.

<service name="examples/core/first/MathService" provider="Handler" use="literal" style="document">

If you combine the ‘name’ entry from above to the base directory for all services, you get the following:

http://<IP>:8080/wsrf/services/examples/core/first/MathService

This URL will be the URL of the service once it is deployed in the container, where <IP> is your machines IP address or “localhost”. The lines that look like the following are important for the follow-up to the first building.

<parameter name="className" value="org.globus.examples.services.core.first.impl.MathService"/>

<wsdlFile>share/schema/examples/MathService_instance/Math_service.wsdl</wsdlFile>

These lines specify where the class name for the service may be found and where the WSDL file for that service may be found.

__________________________

5 http://gdp.globus.org/gt3-tutorial/multiplehtml/ch03s03.html