

Constructing Stateful Web Services using Globus Toolkit 4

Viera Sipkova
Institute of Informatics, SAS
Dúbravská cesta 9, Bratislava
Slovakia

www.eu-egee.org



- **Open Grid Service Architecture (OGSA)**
- **Web Services Resource Framework (WSRF)**
- **Globus Toolkit 4 (GT4)**
- All is based on standard Web services technologies as SOAP and WSDL.

- **OGSA** - defines a common, standard, and open architecture for grid-based applications.
- The goal of **OGSA** -
is to standardize all the services one finds in grid applications by specifying a set of standard interfaces for these services.

- **Web Services Resource Framework (WSRF)** specifies stateful Web services.
- **WSRF** is concerned primarily with the creation, addressing, inspection, and lifetime management of stateful resources.
- **WSRF** represents the infrastructure which the OGSA is built on.

- **Globus Toolkit 4 (GT4)** -
is the software toolkit for programming
grid-based applications.
- **GT4** - includes many high-level services
which are implemented on top of WSRF.
- WSRF represents the core of GT4, however,
on the long road towards true Grid Nirvana,
WSRF is a necessary, but only the first step.

- Stateful Resources can come in all different shapes and sizes (simple values, files, rows in a database, ...).
- A Resource may be composed of zero or more service data elements, called **Resource Properties**.
- **Resource Properties** are defined in the Web service's interface using the standard XML Schema global element declarations.

Writing and Deploying a WSRF Web Service

1. Define the service's interface (with WSDL).
2. Implement the service (with Java).
3. Define deployment parameters
(with WSDD and JNDI).
4. Compile everything and generate GAR file
(with Ant).
5. Deploy the service (with Ant).

Simple stateful Web Service GenericService

- Operations provided by GenericService:
`executeJob`, `transferJobResult`,
`viewJobIdent`, `viewJobStatus`, ...
`createResource`.
- The resource properties of GenericService:
`jinputSandbox`, `outputSandbox`,
`jobIdentifier`, `jobStatus`.

1. Defining the Web Service's Interface

- The interface must specify what the Web service is going to provide to the outer world.
- The service interface is called the **portType** and is specified in the **Web Service Descriptor Language (WSDL)**.

Basic parts of the WSDL Interface file

- The root element - <definitions> tag
- Interface itself - <portType> tag
- Input and output messages for each operation - <message> tag
- Declaration of request and response elements along with the resource properties
 - <types> tag

Root element

```
<definition name="GenericService"  
targetNamespace=  
"http://medigrid/globus/org/generic"  
<!-- bunch of WS and WSRF namespaces -->  
<import namespace=  
"http://.../WS-ResourceProperties.wsdl"/>  
<!-- other necessary WSRF WSDL-files -->  
</definition>
```

PortType

```
<portType name="GenericPortType"  
wsrp:ResourceProperties=  
"GenericResourceProperties">  
  
    <!-- for all operations -->  
    <operation name="executeJob">  
        <input message="tns:ExecuteJobRequest"/>  
        <output message="tns:ExecuteJobResponse"/>  
    </operation>  
  
</portType>
```

Input and Output Messages

```
<!-- all messages -->

<message name="ExecuteJobRequest">
    <part name="parameters"
        element="tns:executeJob"/>

</message>

<message name="ExecuteJobResponse">
    <part name="response"
        element="tns:executeJobResponse"/>

</message>
```

Request and Response elements

```
<types>
```

```
<schema targetNamespace=
```

```
    "http://medigrid/globus/org/generic"
```

```
    xmlns="http://www.w3.org/2001/XMLSchema"
```

```
    <import namespace="http://.../addressing"
```

```
        schemaLocation=".../WS-Addressing.xsd"/>
```

```
    <!-- other namespaces and imports -->
```

```
    <!-- resource properties -->
```

```
    <!-- operation types -->
```

```
</schema> </types>
```

Resource Properties

```
<xsd:element name="jobIdentifier"
  type="xsd:string"/>

<xsd:element name="inputSandbox"
  type="xsd:string"/>

<xsd:element name="outputSandbox"
  type="xsd:string"/>

<xsd:element name="jobStatus" type="xsd:string"/>
```

Resource Properties (cont.)

```
<element name="GenericResourceProperties">
    <complexType> <sequence>
        <element ref="tns:jobIdentifier"/>
        <element ref="tns:inputSandbox"/>
        <element ref="tns:outputSandbox"/>
        <element ref="tns:jobStatus"/>
    </sequence> </complexType>
</element>
```

Request and Response elements

```
<!-- for all operations -->

<xsd:element name="executeJob">
  <xsd:complexType> <xsd:sequence>
    <xsd:element name="inp" type="xsd:string"/>
    <xsd:element name="outp" type="xsd:string"/>
  </xsd:sequence> </xsd:complexType>
</xsd:element>

<xsd:element name="executeJobResponse">
  <xsd:complexType>
</xsd:element>
```

2. Implementing the Web Service (in Java)

- Service Resource
- Service Home
- Service itself

Service Resource

- Resource - stateful Java object including the methods for handling resource properties.
- Initialize function – creates the resource identifier, and creates and initializes all resource properties.
- Get/Set functions for each property

GenericResource (code fragment)

```
public class GenericResource implements  
    Resource, ResourceProperties, ResourceIdentifier,  
    GenericConstants  
{ // resource identifier  
    private Object key;  
    // resource properties  
    private ResourcePropertySet propSet;  
    private ResourceProperty jobIdentifier;  
    ...  
}
```

GenericResource (code fragment)

```
public void initialize() {  
    // create resource identifier  
    this.key = new Integer(hashCode());  
    // create resource properties  
    this.jobIdentifier = new SimpleResourceProperty(  
        JOBIDENTIFIER_RP)  
    this.jobIdentifier.add("none");  
    this.propSet.add(this.jobIdentifier);  
    ...  
}
```

Service Home

- Service Home java object - is in charge of managing the resources (create, find, ...)
- Create method will
 - construct the new instance of the resource,
 - invoke initialization of the resource,
 - create the ResourceKey object,
 - add the new resource to the internal list,
 - return the ResourceKey

GenericResourceHome (code fragment)

```
public ResourceKey create() {  
    // create a new GenericResource  
    GenericResource resource =  
        (GenericResource) createNewInstance();  
    resource.initialize();  
    ResourceKey key = new SimpleResourceKey(  
        keyTypeName, resource.getID());  
    this.add(key, resource);  
    return key;  
}
```

Service itself

- Service itself represents a stateless Web service interacting with clients.
- It must implement the `createResource` operation which returns the endpoint reference for that resource.
- It must implement also all other operations defined in the WSDL interface.

GenericService (code fragment)

```
public EndpointReferenceType createResource(  
    CreateResource request) {  
    // localize GenericResourceHome  
    ResourceContext ctx = ResourceContext();  
    GenericResourceHome home =  
        (GenericResourceHome)ctx.getResourceHome();  
    // create a new GenericResource  
    ResourceKey key = home.create();  
    ...  
}
```

GenericService (code fragment)

```
public EndpointReferenceType createResource(  
    CreateResource request) {  
  
    ...  
  
    // create an endpoint reference  
    ResourceKey key = home.create();  
    EndpointReferenceType epr = AddressingUtils.  
        createEndpointReference(ctx, key);  
    return epr;  
}
```

GenericService (code fragment)

```
public ExecuteJobResponse executeJob(  
    ExecuteJob params) {  
    // get appropriate resource  
    GenericResource resource = (GenericResource)  
        ResourceContext.getResourceContext().getResource();  
    // set inputSandbox property  
    String input = params.getInp();  
    resource.setInputSandbox(input);  
  
    ...  
}
```

GenericService (code fragment)

```
public ExecuteJobResponse executeJob(  
    ExecuteJob params) {  
  
    ...  
  
    // submit job for execution  
    GenericTask task = new GenericTask(  
        resource.getInputSandbox());  
    task.submitTask();  
  
    // get task output  
    String toutput = task.getTaskOutput();  
  
    ...  
}
```

3. Deployment of the Web Service

- The deployment phase is specified with the **Web Service Deployment Descriptor** (WSDD) file, and **Java Naming and Directory Interface** (JNDI) file.

WSDD and JNDI Descriptors

- **WSDD** file tells the Web service container how to present the service to the outer world. It defines the location of the interface and implementation of the service, and other GT4-specific features.
- **JNDI** file is used to locate resource homes. It relates the Web service to a particular resource home and resource.

4. Creating a Grid Archive (GAR) file

- The GAR file is a single file which contains all the files and information the Web service container needs to deploy the service and make it available to the whole world.
- GAR file is constructed with Ant tool.

Creating a GAR file includes:

- Processing the WSDL file.
- Creating the stub classes from WSDL.
- Compiling the stub classes.
- Compiling the service implementation.
- Organizing all the files into a very specific directory structure.

5. Deploying the service into a Web service container

- Deployment is done with the Ant tool which unpacks the GAR file and copies the files within (WSDL, compiled stubs, service implementation, WSDD) into key locations in the GT4 directory tree.
- It also configures the Web service container to take the new service into account.

GenericService Clients

- **ClientSubmit** -
will create the resource and submit the job
for execution.
- **ClientTransfer** -
will transfer the job results .
- **ClientView** -
will show the resource properties of the
service.

ClientSubmit

- Parse the command-line arguments.
- Invoke creating a new GenericResource.
- Get the endpoint reference (EPR) from the new resource.
- Write the EPR to the file for later usage.
- Set the resource properties.
- Execute job with resource referenced by EPR.

ClientSubmit (code fragment)

```
public class CleintSubmit extends BaseClient {  
    final static ClientSubmit client = new ClientSubmit();  
    final static GenericServiceAddressingLocator loc = ...  
  
    public static void main(String[] args) {  
        // parse the commandline  
        String jobScript = ...; String outFile = ...;  
        // create the GenericResource  
        GenericPortType port =  
            loc.getGenericPortTypePort(client.getEPR());
```

ClientSubmit (code fragment)

// write epr to file

```
EndpointReferenceType epr = port.createResource(  
    new CreateResource());  
writeEPR(epr);
```

// submit job

```
GenericPortType job =  
    loc.getGenericPortTypePort(epr);  
job.executeJob(new ExecuteJob(JobScript, outFile));
```

ClientTransfer

- Parse the command-line arguments.
- Get the resource GenericResource specified by the EPR-file argument.
- Copy file specified by sourceURL argument to file specified by destinationURL argument.

ClientView

- Parse the command-line arguments.
- Get the resource GenericResource specified by the EPR-file argument.
- Print the contents of all GenericResource properties.

GenericService command-line Clients

- ClientSubmit – is configured to generic-submit
- ClientTransfer – is configured to generic-transfer
- ClientView – is configured to generic-view

Invocation of command-line Clients

- generic-submit -s <serviceURI> \
 <jobScript> <outputFilename>
- generic-transfer -e <eprFilename> \
 <sourceURI> <destinationURI>
- generic-view -e <eprFilename>

Process of Constructing the Generic Service and Clients

- Creating GAR file for GenericService.
- Deploying the GenericService into the Web service container.
- Compiling the Generic clients.
- Configuring the Generic clients to command-line clients.

Ant – Java build tool

(<http://jakarta.apache.org/ant>)

- Ant allows to build executables from source files.
- Individual steps of the process are described in a build file in XML format.
- Ant reduces the whole process to one step.



Enabling Grids for E-sciencE

**Thank you for your
attention.**

www.eu-egee.org



Information Society

