National e-Science Centre

Tutorial 2: Globus Development

John Watt, National e-Science Centre

Tutorials Timetable

Week	Day/Time	Topic	Staff
3	Fri 11am	Introduction to Globus	J.W.
4	**Fri 11am**	**Globus Development**	**J.W.**
5	Fri 11am	Globus Development	J.W.
6	Fri 11am	Condor	J.W.
7	Tue 12pm	SAML/PERMIS (L)	A.S.
7	Wed 12pm	Portals (L)	J.J.
7	Fri 11am	Q & A Session	all
8	Fri 11am	OGSA-DAI (L)	O.A.
10	Tue 12pm	Example Systems (L)	R.S.
10	Fri 11am	Assignment Demos	all

Several points…

- Feel free to change your passwords!
  - User account already has quite strong passwords
  - Globus account is the same for everyone
    - Globus account is low-privilege, but feel free to change password
- Don’t power off your machine!
  - We run a Condor pool which runs while your machines are idle
    - More on this tool in tutorial 4
  - Just log out…
- Set proxy in your user account for internet
  - Mozilla -> Edit -> Preferences -> Connection Settings
  - http://wwwcache.gla.ac.uk/glasgow.pac

Recap

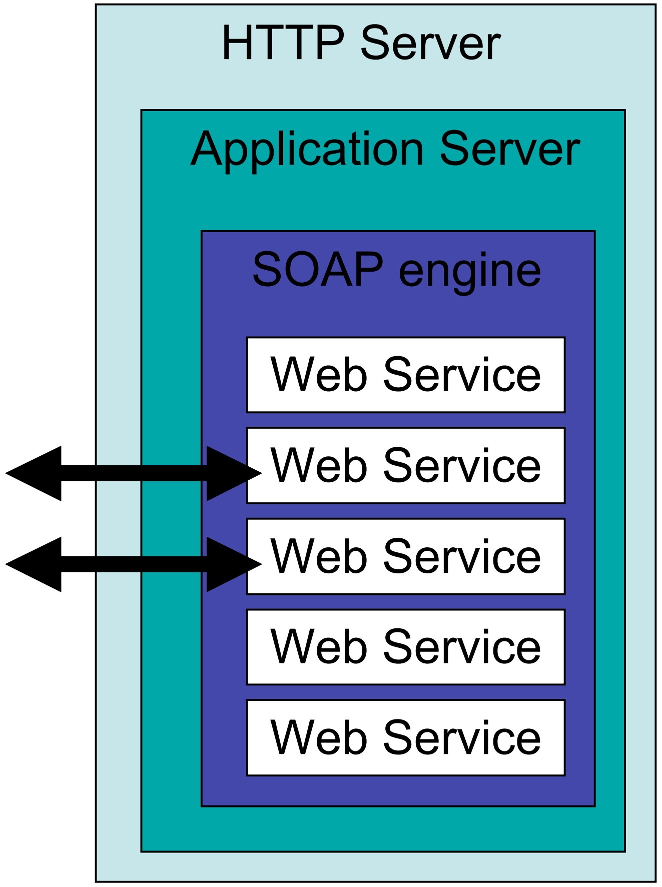
- Last time we:
  - Launched a GT4 Web Services container
  - Created a proxy certificate
  - Used a script to build a Web Service from constituent files (.WSDL, .WSDD, .java)
  - Deployed this service in the container
  - Ran a simple client to invoke the service
- This week, we'll have a closer look at what we did last week…
  - Will be referencing some security aspects from this week's lectures

The Container

- Generic term for server-side applications
- Comprises
  - An HTTP server for sending/receiving messages over the internet (e.g. Apache)
  - An application server for hosting services (e.g. Tomcat)
    - Exposes our services to the internet through the HTTP server
  - A SOAP engine for manipulating SOAP messages (e.g. Apache Axis)
    - Interprets messages from the application server
  - Our Web Services

The Container

- 'Container' is started using one command
  - `# globus-start-container`
    - Can take the '-nosec' flag for no transport security
    - We will ALWAYS use no transport security for OUR services (always local)
    - Without flag, services become HTTPS
  - Takes a while to run up…
    - Just like Tomcat/JBoss, application server initialises in a few seconds



The Container

- A few tips…
  - **Container runs as ‘globus’ user**
    - The user that installed the globus toolkit
  - **Always start the container from $GLOBUS_LOCATION**
    - Container may fail to start at all if it cannot locate directories relative to the installation directory
  - **Stop container with <Ctrl>-C**
  - **Ignore error messages from**
    - ReliableFileTransfer
    - QueryAggregator
      - These are unconfigured parts of Globus Toolkit complaining, they don’t impact on any services we will be looking at

Certificates

- Before running any services we typed
  - grid-proxy-init
    - And we got some output…

```
Your identity: /C=UK/O=Grid/O=Training/OU=GUGridComputingCourse/CN=User12
Creating proxy ........................................... Done
Your proxy is valid until: Thu Feb  1 23:08:00 2007
```

  - First line states your SUBJECT DN
  - Second line generates your proxy
    - Automatic as we have turned off private key encryption
  - Third line states the validity of the short-lived credential (12 hours default)

Certificates

- **Your certificates are stored in `~/.globus/`**
  - **usercert.pem**
    - Your e-Science PKI public user certificate
  - **userkey.pem**
    - Your e-Science PKI private key (notice permissions!)
  - **/certificates/cb398b31.0**
  - **/certificates/cb398b31.signing_policy**
    - These files contain information about the CERTIFICATE AUTHORITY that issued your e-Science certificate
      - First is the CA ROOT CERTIFICATE
      - Second is the Subject DN scope that this CA refers to
        - » Certificates outwith this policy cannot be verified here…

Certificates

- grid-proxy-init creates a 'proxy' certificate
  - This is a short-lifetime certificate to restrict damage should it be compromised
  - Stores it in `/tmp/x509up_$UID`
  - You can have a look inside any certificate you own with:

```
# openssl x509 –in <certificate> –noout –text
```

  - <u>Exercise</u>: Compare your proxy certificate and your user certificate using this command.
    - What TWO things in particular do you notice??

Certificates

- **Is all this necessary?**
  - **We turned off security in the container, didn't we?**
  - **No! We only turned *transport* security off**
    - We still need to AUTHENTICATE to globus to run clients
  - **But don't we need server side authentication? We didn't do a grid-proxy-init for the container…**
    - Yes! But Globus does that automatically for us
  - **Then where are the server side credentials?**
    - They are in **`/etc/grid-security`**
    - Globus owns its certificate and key here
    - And root CA details are stored in `certificates/`

grid-mapfile

- There is something else in `/etc/grid-security` of interest to us…
  - Run command:

```
# more /etc/grid-security/grid-mapfile
```

  - You should see a line with your Certificate Subject DN in inverted commas followed by your user account

  - This file maps your identity to a local account that your jobs will run in
    - This is the AUTHORISATION step in GT4
    - Is this a good way of doing this?

grid-mapfile issues

- Imagine Tescos have a ‘grid-mapfile’ for their loyal customers to get 10% off at their stores

```
Michael Balzary gets 10% off goods
James Jamerson gets 10% off goods
John-Paul Jones gets 10% off goods
Carole Kaye gets 10% off goods
Les Claypool gets 10% off goods
Kris Novoselic gets 10% off goods
John Wardle gets 10% off goods
Colin Greenwood gets 10% off goods
Etc.. Etc… etc… etc… etc……………
```

  - Mapping of privilege to user done at <u>resource</u>

grid-mapfile issues

- This doesn't happen. Tescos issue a 'loyalty card' entitling user to 10% off goods
  - Mapping of privilege to user done at <u>user</u>
  - 'grid-mapfile' would then look like:

```
Loyalty Card holder gets 10% off goods
```

  - The resource access control statement is only one line, as opposed to a line for each user
  - This is Role Based Access Control (RBAC)
    - RBAC is redefining how authorisation is done on the Grid

Web Services

- **Recall the constituents of our services**
  - **A WSDL document**
  - **A WSDD document**
  - **An implementation**
  - **Build settings/scripts**

  - **There is another constituent of Web Services that we haven't discussed**
    - This is because it is generated automatically for us
    - The clue lies in the invocation command from last week:

```
# java –classpath ./build/stubs/classes/:$CLASSPATH org.globus……
```

Stubs

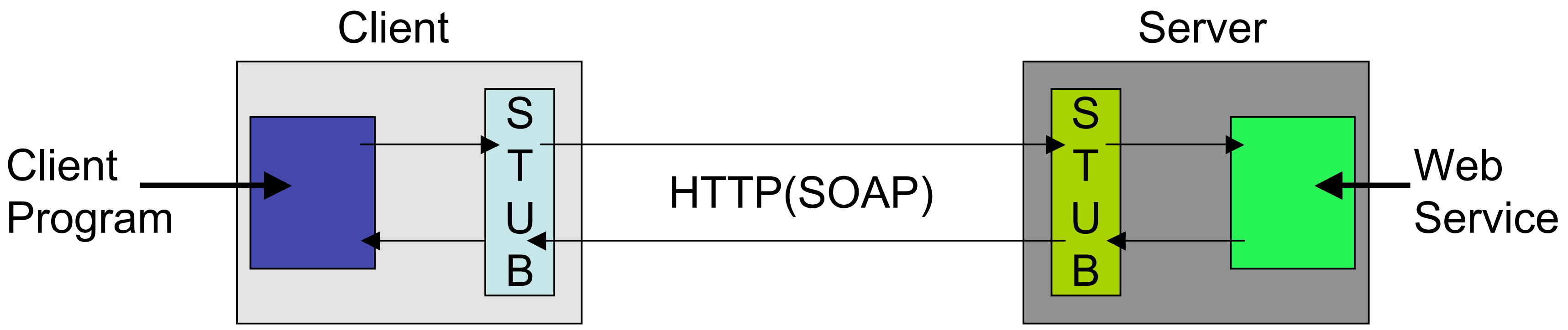
National e-Science Centre

- Stubs perform SOAP interpretation on our behalf
- Imagine a simple Web Service invocation…
  - Web Service is located (Discovery process)
  - WSDL of service is read (Description)

  - At this point a CLIENT STUB will be generated from the service WSDL (automatically if required)
    - This will communicate with the Web Service via SOAP
  - This stub may be reused as many times as needed
  - They save your application having to do message encoding/decoding

Stubs

- The server requires a stub too



- The server stub is created when you build your service
  - Stub is said to *marshall* or *serialise* the SOAP requests for us
  - They are placed in the ./build/stubs directory

Stubs

- Stubs are used to map your WSDL service interface definition to your actual implementation
  - WSDL contains no information about how your service is implemented
    - But the stubs do!
  - Enter the namespace.mappings file (in $TUT_DIR)
  - It maps WSDL namespaces to real stub classes

  - Note that stubs classes are generated AFTER you build the service, so you have to be careful how you construct this file!

WSDL

- WSDL files describe the operations that a service provides
- Comprises:
  - A definitions element
  - A portType element
  - A messages element
  - A types element

  - *Bindings element is generated automatically by our build scripts*
  - *Services is defined in the deployment descriptor*

WSDL

- **<definitions>**
  - **Root element of WSDL file**
  - **We are interested in ‘name’ and ‘targetNamespace’**
    - These define the name and targetNamespace of the WSDL file itself (not the portType interface – this is later)
  - **All the other attributes within the <definitions> tag are required by every Web Service**
    - Some depend on which WSRF specs you wish to import into your service i.e. WS-ResourceProperties, WS-ResourceLifetime
      - *You will never need any more than these two specs*
    - These are listed as <wsdl:import/> tags immediately after the <definitions> tag

WSDL

- <portType>
  - Defines our operations
  - Main tag has the name of the portType, a WSDL pre-processor definition, and a ServiceResourceProperties attribute (in <types>)
  - We also have <operation> tags
    - These define which messages correspond to each operation

```
<operation name=“do_something”>
  <input message=“tns:do_somethingInputMessage”/>
  <output message=“tns:do_somethingOutputMessage”/>
</operation>
```

WSDL

- **<messages>**
  - **Defines the messages our operations will use (which have been defined in the portType tag)**
    - Using the operation already defined, the “do_somethingInputMessage” will contain the “do_it” element
    - We only use single element ‘parts’

```
<message name=“do_somethingInputMessage”>
  <part name=“parameters” element=“tns:do_it”/>
</message>
<message name=“do_somethingOutputMessage”>
  <part name=“parameters” element=“tns:do_itResponse”/>
</message>
```

WSDL

- <types>
  - Defines the response and request types
  - Declares the resource properties
    - As required by the ServiceResourceProperties attribute in the portType definition
  - Contains an <xsd:schema> tag (standard WSDL)

```
<xsd:element name="do_it" type="xsd:string"/>     (do_it type)


<xsd:element name="last_action" type="xsd:string"/>    (resource properties)
<xsd:element name="ServiceResourceProperties">
   <xsd:complexType>
     <xsd:sequence>
              <xsd:element ref="tns:last_action" minOccurs="1" maxOccurs="1"/>
     </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Web Service Addresses

- When you start your container you get a numbered list of services
  - These are all Web Services listed as URIs (Uniform Resource Identifiers)
  - They look like normal Web addresses

```
http://130.209.58.100:8080/wsrf/core/services/MyService
```

  - But a Web Service needs to be invoked in a certain way, so if you typed this URI into Mozilla you wouldn’t see anything
    - These URIs are for the use of OTHER SERVICES
    - “Web PAGES for humans / Web SERVICES for computers”

Deployment

- GT4 requires two pieces of information to deploy your service in its container
  - A WSDD deployment descriptor
  - A JNDI deployment file

  - We won’t be using the functionality in the JNDI file, but we need it defined (you will change the service name in this file and nothing else)

  - The deployment descriptor contains publishing information

WSDD

National e-Science Centre

-
- **Contains standard namespace defs + a <service> tag**
  - **<service name=“tutorial/Service”>**
    - This defines what the URI for this service will be
    - It gets appended to the baseURL for the service
    - So for example, if the baseURL was

http://130.209.58.100:8080/wsrf/services

    - The service URI would become

http://130.209.58.100:8080/wsrf/services/tutorial/service

WSDD

- Inside <service>:
  - A Parameter tag for className points at the class which implements our service

```
<parameter name="className"
  value="org.globus.services.Service.impl.Service"/>
```

  - A wsdlFile tag points to the NEW WSDL file generated by the build script

```
<wsdlFile>schema/Service/Service_service.wsdl</wsdlFile>
```

    - Note that this is NOT the WSDL file we wrote

Next Week

- Service Implementation
  - This is where you program your Java service
  - We will look at the extras you need to include for GT4 to understand your programming!
- Problem Set 3 Assessment
  - The last 15 minutes of the tutorial we will come round and check your simple calculator service is working
  - You <u>MUST</u> be able to show your service incrementing with each invocation