WSMetacatService – a GT4 Web Service Wrapper for Metacat

Author: Terry Fleury (tfleury@ncsa.uiuc.edu)
Date: October 3, 2005

Summary

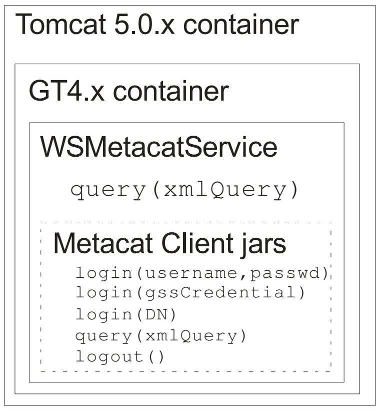
In addition to the GSI-enabling of the https connection to Metacat, work was done to allow the Metacat client API to be accessible via web services. To do this, a Globus Toolkit 4 Java WS Core "wrapper" was placed around the Metacat client jars. Additionally, two new "`login`" methods were coded to circumvent the need to pass the user's password to the Metacat server.

On September 20, 2005, the LTERgrid Pilot application was demonstrated showing how the connection between a Metacat client and a Metacat server could be secured using a GSI-enabled https connection. Bill Baker (bbaker@ncsa.uiuc.edu) implemented many changes within the core Metacat client/server code to allow a new protocol "`httpg`" to be used.

At the same time, I worked on making Metacat accessible via web service protocols. By using the Java WS Core portion of the Globus Toolkit 4.0.1, I created a web services front-end for Bill's modified Metacat client. This WSMetacatService was not deployed anywhere and was done solely as an exercise.

Web Service Overview

The WSMetacatService GT4 web service relies on the connection between client and server to be performed over a GT4 gsi-secure https connection. This is done using an X.509 credential when connecting from a web service client to the WSMetacatService. When deployed, the web service is wrapped as in the following diagram.



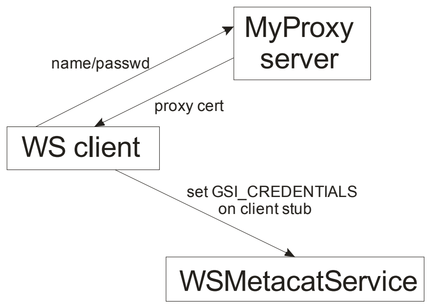
First, the Metacat client jars (utilities.jar, metacat.jar, and metacat-client.jar) are compiled incorporating the changes made by Bill Baker to allow login with either a gssCredential or a distinguished name (DN).

Then the WSMetacatService is compiled with these jars. This generates a .gar file which can be deployed to a GT4 container.

Finally, this Globus container can be deployed to a modified (to allow gsi-https connections) Tomcat 5.0.x installation, resulting in a web service implementation of Metacat.

Information for configuring Tomcat 5.0.x to accept Globus gsi-https connections can be found in the GT4 Java WS Core System Administrator's Guide.

On the client side, a web service client first connects to a MyProxy server where it gets a user's proxy certificate using a username/password (presumably entered in by the user). The client then gets an endpoint reference and porttype for the WSMetacatService and sets the `GSI_CREDENTIALS` property on the client stub. Then, by connecting to the Tomcat server on the gsi-https port, this credential gets passed to the WSMetacatService.



How MUCH of the credential gets passed depends on the type of connection made. For example, the `login(gssCredential)` method requires a full credential, thus the proxy certificate needs to be delegated from the client to the server. To do this easily, GT4 has an authentication method called `GSISecureConversation`. This provides a mechanism for generating a security session, i.e. the negotiation of a shared secret which may be used to secure a set of subsequent messages. It is based on WS-Trust and WS-SecureConversation. It creates a delegation service collocated with the web service such that the gssCredentials specified on the client side can be accessed on the server side.

Alternatively, the `login(distinguishedName)` method relies on the assumption that the WSMetacatService and the Metacat server are collocated, so that no authentication mechanism is needed. Here, all we need to do is set the transport mechanism to use encryption, still setting the `GSI_CREDENTIALS` on the client stub. Since we are not doing full delegation of credentials, only the distinguished name (DN) of the proxy certificate is available to the WSMetacatService.

On the server side, only the `query(xmlQuery)` method is exposed as a public method. This is because the `query()` method implicitly calls the Metacat `login()/logout()` methods. The `login()` method to be used must be decided on the server side at Tomcat startup because the security settings for a GT4 WS service are declared in a security descriptor and cannot be changed at runtime. This server-side setting then dictates which authentication method must be used by the WS client.

Implementation Details

As with any web services application, and in particular a GT4 WS application, there are many interrelated files that need to be created. I created a directory named "`ws`" at the same level as the "`metacat`" and "`utilities`" directories. Really, it can be placed just about anywhere since it doesn't reference any of the Metacat stuff directly. Here is a list of the server-side files, listed in a top-down directory format.

```
ws/globus-build-service.py
ws/globus-build-service.sh
ws/build.xml
```

> These three files are part of the Globus Service Build Tools project (http://gsbt.sourceforge.net/) and are basically a front-end to 'ant' to build Globus Web Services applications. I used the 0.2.5 version of the scripts, unmodified.

`ws/build.mappings`

> This file is used by the globus-build-service scripts so that all one needs to do to ‘compile’ the service is type “`globus-build-service wsmetacat`”.

`ws/namespace2package.mappings`

> Needed by GT4 WS projects, this file maps namespace URIs to client stub names.

`ws/schema/wsmetacat/WSMetacatService/WSMetacat.wsdl`

> This is a ‘standard’ Web Services Description Language file which lists the types, messages, and porttypes provided by the wsmetacat web service. As there is only one method currently available (`query`), this file is very basic. The only special thing about it is the “`wsdlpp:extends`” attribute, which is a directive to the WSDL preprocessor provided by Globus so we don't have to copy/paste definitions from the WSRF specs.

`ws/edu/lternet/wsmetacat/deploy-jndi-config.xml`

> This ‘standard’ WS file specifies the resource home for the service. However, since no resource properties are currently used by the service, this is a very basic file. Note that the directory structure ‘`edu/lternet/wsmetacat`’ was chosen arbitrarily.

`ws/edu/lternet/wsmetacat/deploy-server.wsdd`

> This ‘standard’ WS Deployment Descriptor file describes how the web services container should publish the wsmetacat service. There's not a lot special going on here.

`ws/edu/lternet/wsmetacat/etc/wsmetacat.properties`

> Files in this `etc` directory will be available in the WS container's `etc` directory so that an administrator can easily change values in these properties files and change the behavior of the web service without having to recompile/redeploy. This properties file contains the URL of the GSI-enabled Metacat web service.

`ws/edu/lternet/wsmetacat/etc/wsmetacatsecurity.xml`

> This file specifies the methods of authentication and authorization required by any client connecting to the wsmetacat service. If the authentication mechanism is `GSISecureConversation`, then the authorization mechanism cannot be ‘`none`’. `GSISecureConversation` is required if the `login(gssCredential)` method is used. Otherwise, no authentication is needed when using the `login(distinguishedName)` method.

`ws/edu/lternet/wsmetacat/impl/WSMetacatQNames.java`

This file describes the Qualified Names for the wsmetacat service. Since we are not currently using any resource properties, this file is very basic and really didn't need to be coded as a separate interface. I put it in a separate .java file just in case future methods require resource properties.

`ws/edu/lternet/wsmetacat/impl/WSMetacatService.java`

I saved the best for last. This is the main implementation of the wsmetacat web service. There is currently only one public method, `query()`, which is basically a wrapper around the 'normal' Metacat API `query()` method. Functionally, it differs from the 'normal' Metacat API method in the following ways:

- Both the input parameter and returned value are standard Java Strings, not Readers. This is to facilitate the passing of data to/from the web service.
- The method calls the `login()` and `logout()` methods automatically, which is why those two methods are not exposed to the client programmer.
- In order to do the automatic `login()`, the user's credential must have been set on the client side. This is done by setting the `GSIConstants.GSI_CREDENTIALS` value to the user's credential on the client stub.

Bill Baker wrote two new `login()` methods, one taking a gssCredential as the parameter, and the other taking a DN (distinguished name) as the parameter. In order to use the `login(gssCredential)` method, a delegation service must be used. By using the `GSISecureConversation` authentication mechanism, a delegation server is automatically created in the GT4 container on the server side, and thus the client can pass the credential to the web service server. On the server side, the gssCredential can be retrieved as follows:

```
import javax.security.auth.Subject;
import org.globus.gsi.jaas.JaasGssUtil;
import org.globus.wsrf.impl.security.authentication.Constants;
gssCredential = JassGssUtil.getCredential(
  (Subject)MessageContext.getCurrentContext().getProperty(
    Constants.PEER_SUBJECT));
```

If instead we use the `login(DN)` method, no authentication is required. The DN can then be retrieved as follows:

```
import org.globus.wsrf.security.SecurityManager;
distinguishedName = SecurityManager.getManager().getCaller();
```

After we have logged in, we call the standard Metacat API `query()` method followed by the `logout()` method, and then return the result of the query to the client.

`ws/edu/lternet/wsmetacat/client/Client.java`

> This is a little test client application I wrote for testing out the wsmetacat service. It's not really meant for primetime. Basically, it gets a user's proxy certificate from the MyProxy server on roadrunner.lternet.edu, sets the `GSI_CREDENTIALS` on the client stub appropriately, and calls the `query()` method. Originally, only the `login(gssCredential)` method was available to me. So, the trickiest part was figuring out how to get a MyProxy gssCredential from the client side to the server side so I could call that method. In order to do that, I had to use `GSISecureConversation` which automatically sets up a delegation service colocated with the GT4 container (inside the web service container). But `GSISecureConversation` requires some sort of authorization as well, so that property had to be set appropriately. All of that goes away when using the `login(DN)` method.

References

Globus Toolkit 4.0.1 Java WS Core:
http://www.globus.org/toolkit/docs/4.0/common/javawscore/
http://www.globus.org/toolkit/downloads/4.0.1/#wscore_bin
http://www-unix.mcs.anl.gov/~gawor/javawscore/HEAD/doc/javadocs/

GT4 Tutorials:
http://gdp.globus.org/gt4-tutorial/singlehtml/progtutorial_0.1.1.html
http://www.gridnexus.org/web/external/gt4-security.html

LTERgrid Project:
http://grid.ncsa.uiuc.edu/LTER/

Metacat:
http://knb.ecoinformatics.org/software/metacat/