GT4 PERMIS Authorisation Service Installation Instructions

Author: Sassa Otenko, George Inman

Version	Date	Comments
1.0	7 April 2005	Released by Sassa
1.1	26 March 2007	Updated by George
2.0	18 April 2007	Added GridShib by George
2.1 2.2	7 August 2008 25 February 2009	Linying: Using the o=PERMISv5,c=GB entry in the Kent LDAP Some minor changes by Stijn

Contents

1. Installation

To install GT4 PERMIS Authorisation, you need to have Globus Toolkit 4 installed first. The recommended version is GT 4.0.8 .

Download the GT4 PERMIS. Authorisation Service from http://sec.cs.kent.ac.uk/permis/integrationProjects/GT.shtml .

From the $GLOBUS_LOCATION directory, execute

```
bin/globus-deploy-gar permisAuthzGT4.gar
```

(the actual directory name is specified by you when you install GT4). This will do the following:

- Install the necessary Java libraries (including IAIK JCE provider - see the IAIK site for terms and conditions of use; at the time of writing the use of the library is free of charge for education and research purposes) All the supporting libraries that are needed by PERMIS are installed in $GLOBUS_LOCATION/lib/permis_include.
- Deploy a Permis Authorization Service with default configuration (see below how to test this)

Now you have installed the PERMIS Authorisation Service with a sample configuration. Later we will explain how to configure the PERMIS Authorisation Service for your own needs.

1.1 Testing with the Default Configuration

To test the default configuration we will show how to protect a *ShutdownService* available in the GT4 Core, so that only users with the *RemoteAdministrator* role will be able to shut down the GT4 container.

The test will consist of running

```
bin/globus-start-container
```

in one console, and

```
bin/globus-stop-container
```

in another console. If you have the correct credentials, the container will stop; otherwise, you should receive an error message that explains the reason for failure.

We recommend that you download a key pair issued to */C=GB/O=PERMISv5/CN=User0* from our website (http://sec.cs.kent.ac.uk/permis/user-keysv5.zip), so that PERMIS will be able to use the X.509 Role Assignment Attribute Certificates that we have already created and stored in our LDAP server for testing purposes. Otherwise you will need to issue a Role Assignment Attribute Certificate with the (*permisRole,RemoteAdministrator)* role to the user that will try to stop the container.

The sample private key is not encrypted (i.e. no password is set). Note that these keys are used for demonstration purposes only, which is the only reason that the public key certificate expires in five years time.

When you have the key-pair, you will need to generate the key-pair for the proxy - the Service Container will act as the proxy of the owner of the original key-pair. To do this, unzip the user-keysv5.zip in your ***home*** directory and:

1. Make sure that the *userkeyv5.pem* and *usercertv5.pem* certificates are in the directory *~/.globus/*, and that the *0935b792.0* and *0935b792.signing_policy* certificates are in the *~/.globus/certificates* directory.

   Note that on Windows platforms the directory is also the home directory for the logged in user (e.g. the user certificates should be saved in *"C:\Documents and Settings**<user name>**\.globus"*)

2. Make sure that the environmental variable *GLOBUS_LOCATION* is set and execute (from the GT4 installation directory)

   ```
   bin/grid-proxy-init –key ~/.globus/userkeyv5.pem –cert
   ~/.globus/usercertv5.pem
   ```

The GT4 container will now use the generated proxy certificate for authentication.

You will now need to update the configuration of the *ShutdownService* so that PERMIS Authorisation will make access control decisions for it. To do that you will need to update the *etc/globus_wsrf_core/server-config.wsdd* file in the following way:

```
<service name="ShutdownService" provider="java:RPC"
    use="literal" style="document">
    ...
  <!-- comment out the existing security descriptor →
  <!-- parameter name="securityDescriptor"
        value="org/globus/wsrf/impl/security/descriptor/gsi-
security-config.xml"/ -->
    <!-- PERMIS Authz related -->
    <parameter name="securityDescriptor"
       value="etc/permisAuthzGT4/permis-security-descriptor.xml"/>
    <parameter name="permisAuthz-SOA"
      value="cn=A PERMIS Test User,o=PERMISv5,c=GB"/>
    <parameter name="permisAuthz-OID"
      value="1.2.826.0.1.3344810.6.0.0.13"/>
    <parameter name="permisAuthz-LDAP"
      value="ldap://sec.cs.kent.ac.uk/"/>
    <parameter name="permisAuthz-LDAP_AC_attribute"
      value="attributeCertificateAttribute"/>
    <parameter name="permisAuthz-LDAP_PKC_attribute"
       value="userCertificate;binary"/>
    <parameter name="permisAuthz-ROOT_CA"
      value="etc/permisAuthzGT4/rootca.cer"/>
    <parameter name="permisAuthz-GSP_useGridShib" value="false"/>
    ...
</service>
```

Now we can observe the result instantly: in one console, run (as user globus)

```
bin/globus-start-container
```

in another console, run (as the user that has generated the proxy certificate)

```
bin/globus-stop-container
```

If you used the sample key-pair and changed the *ShutdownService* configuration as explained above, you will see that the GT4 container has stopped in the first console, and that there were no error messages in another console.

If you try to use other credentials (i.e. use a different user keypair), access to *ShutdownService* will be denied, because that user will not have a corresponding X.509 Attribute Certificate issued to him with a *RemoteAdministrator* role in it, and the GT4 container will continue to run.

In the following sections we will explain how to configure your services to be protected by PERMIS, and how to configure PERMIS Authorisation to use the policy that you want.

2. Configuring PERMIS Authorisation

The test example in the previous section applies Role-Based Access Control to *ShutdownService*. There are seven important things that PERMIS needs to know in order to enforce access control:

1. Who issued the policy (**permisAuthz-SOA** parameter in the example)
2. What policy of that issuer to use (**permisAuthz-OID** parameter in the example). *The same issuer may have a number of policies for various situations or targets, so OIDs are used to distinguish between them.*
3. Where the policy is stored (**permisAuthz-LDAP**)
4. In what format the policy is stored (**permisAuthz-LDAP_AC_attribute** parameter in the example).
5. In what format the issuer's PKC certificate is stored (**permisAuthz-LDAP_PKC_attribute** parameter in the example).
6. The PKC of the Root CA that certifies the PKCs of the AC issuers (**permisAuthz-ROOT_CA** parameter in the example). Note that because Authentication and Authorization are separate domains, this Root CA may be different from the one that signs PKCs for Globus Users: this Root CA signs the signature verification keys of the Attribute Authorities that assign roles to the end users.
7. Whether the user credentials will be pulled from LDAP (as in the example) or pushed from gridshib (**permisAuthz-GSP_useGridShib** parameter in the example).
   (7a) If GridShib is to be used and you wish to use roles stored in Attribute Certificates then you will also need to specify the name of the Shibboleth attribute that contains role certificates (**permisAuthz-GSP_AC_LDAP_attribute**).

To configure PERMIS Authorisation, you will need to specify the above values. The default configuration points to our testbed LDAP server at Kent University (ldap://sec.cs.kent.ac.uk/). Our test SOA (*cn=A PERMIS Test User,o=PERMIS v5,c=GB*) has signed and published a policy, of which the XML you can find here. Our test SOA also assigned a *RemoteAdministrator* role to the test user (*cn=User0,o=PERMISv5,c=GB*). The Public Key Certificate that validates the signatures of the SOA is also kept in our LDAP, and it is signed by the Root CA (*cn=Root CA,o=PERMISv5,c=GB*), of which the PKC is in *etc/permisAuthzGT4t/cacert.cer* in your GT4 distribution.

Note that you will need to set up your own LDAP, as our server is not available for writing.

3. Configuring Your Service

Configuring your service is very simple. In the deployment descriptor of your service you need to refer to the security descriptor that specifies the authorisation to use, and the parameters to the authorisation module. So you may be happy to use the *permis-security-descriptor.xml* as in the ShutdownService example - it specifies that all users must use strong authentication, and that PERMIS Authorisation will be used to make access control decisions. Then you will need to specify *permisAuthz-** parameters to configure PERMIS Authorisation for your needs (**remember:** SOA, policy OID, AC repository, and Root CA). This is explained in great detail in the previous section.

If you want to specify a more advanced security descriptor, please refer to the GT4 documentation.

4. PERMIS Authorisation Service for Use with SAML

So far we have discussed only how PERMIS Authorisation can be used locally i.e. how to enforce access control with PERMIS Authorisation run locally. In some scenarios it is beneficial to use an authorisation service provided by a centralised authority, and query the authorisation service remotely. This is especially useful in widely distributed resources, or in cases when your resource is part of a Virtual Organisation, as then the security system has a single PERMIS Authorisation Service to configure, rather than at a number of points scattered around the world.

(In particular, in GT4 it is possible to invoke a chain of authorisation mechanisms. This will allow you, for example, to enforce the local policy as well as the policy of a Virtual Organisation that your resource is part of. We refer you to the GT4 documentation on how to configure this.)

To facilitate remote authorisation, GT4 core has a special kind of authorisation, *samlAuthz*, which contacts a remote authorisation service to make an access control decision. PERMIS Authorisation is compatible with this mode of authorisation. In fact, by deploying PERMIS Authorisation into your system a PERMIS Authorisation service has been installed, and it is configured with the default settings used in the test scenario mentioned before.

To see how this works, we will assume that you have read and performed the "local" test scenario. Now we will change the *ShutdownService* configuration so that *samlAuthz* will be used to invoke the PERMIS Authorisation Service. So it will still be protected by PERMIS Authorisation, but in this case *indirectly*, and the PERMIS Authorisation Service can therefore be located virtually anywhere in the world.

To test the remote invocation of PERMIS Authorisation, you will need to update the *etc/globus_wsrf_core/server-config.wsdd* file in the following way:

```
<service name="ShutdownService" provider="java:RPC"
    use="literal" style="document">
    ...
  <!-- parameter name="securityDescriptor"
          value="org/globus/wsrf/impl/security/descriptor/gsi-
security-config.xml"/ -->


    <!-- comment out all PERMIS related parameters -->
    <!-- PERMIS Authz related -->
    <!-- parameter name="securityDescriptor"
        value="etc/permisAuthzGT4/permis-security-descriptor.xml"/>
    <parameter name="permisAuthz-SOA"
       value="cn=A PERMIS Test User,o=PERMISv5,c=GB"/>
    <parameter name="permisAuthz-OID"
       value="1.2.826.0.1.3344810.6.0.0.13"/>
    <parameter name="permisAuthz-LDAP"
       value="ldap://sec.cs.kent.ac.uk/"/>
    <parameter name="permisAuthz-LDAP_AC_attribute"
       value="attributeCertificateAttribute"/>
    <parameter name="permisAuthz-LDAP_PKC_attribute"
       value="userCertificate;binary"/>
    <parameter name="permisAuthz-ROOT_CA"
       value="etc/permisAuthzGT4/cacert.cer"/>
    <parameter name="permisAuthz-GSP_useGridShib" value="false"/> -->

    <!-- instead reference SAML Authz security descriptor -->
    <!-- SAML Authz related -->
    <parameter name="securityDescriptor"
        value="etc/permisAuthzGT4/saml-security-descriptor.xml"/>
    <parameter name="samlAuthz-authzService"

value="https://localhost:8443/wsrf/services/PermisAuthorizationServic
e"/>
    ...
</service>
```

Now we can observe the result instantly: in one console, run (again as user globus)

```
bin/globus-start-container
```

in another console, run (as the user that has generated the proxy certificate)

```
bin/globus-stop-container
```

If you used the sample key-pair and changed the *ShutdownService* configuration as explained above, you will see that the GT4 container has stopped in the first console, and that there were no error messages in the other console.

The result seems no different from the previous experiment, only now the interaction is more complex. Instead of directly querying PERMIS Authorisation about access control, GT4 makes a remote invocation of the PERMIS Authorisation Service, which makes the decision for the container, which is enforced by the GT4 container.

To configure your own service to be protected by the remote PERMIS Authorisation Service, you will need to add lines similar to the two lines we added to the *ShutdownService* above.

Note that *samlAuthz* has advanced configuration parameters, which can be found in the GT4 documentation.

Note also that it makes sense to use remote PERMIS Authorisation only in cases when the service is located elsewhere, and the example in this section is only for testing purposes. If you use a remote invocation of PERMIS Authorisation that is run locally, the user of your service will lose a significant amount of time because of the additional interactions between GT4 and the PERMIS Authorisation Service.

In the following section we will explain how to configure PERMIS Authorisation Service to provide access control decisions for remote clients.

4.1 Configuring PERMIS Authorisation Service

You will find the deployment descriptor of the PERMIS Authorisation Service in *etc/permisAuthzGT4/server-config.wsdd*. Among GT4 parameters, it has the following parameters:

```
<service name="PermisAuthorizationService" provider="Handler"
     use="literal" style="document">
    ...
    <parameter name="permisAuthz-SOA" value="cn=A PERMIS Test
User,o=PERMISv5,c=GB"/>
    <parameter name="permisAuthz-OID"
value="1.2.826.0.1.3344810.6.0.0.13"/>
    <parameter name="permisAuthz-LDAP"
value="ldap://sec.cs.kent.ac.uk/"/>
    <parameter name="permisAuthz-LDAP_AC_attribute"
value="attributeCertificateAttribute"/>
    <parameter name="permisAuthz-LDAP_PKC_attribute"
value="userCertificate;binary"/>
    <parameter name="permisAuthz-ROOT_CA"
value="etc/permisAuthzGT4/rootca.cer"/>
    <parameter name="permisAuthz-GSP_useGridShib" value="false"/>
    ...
</service>
```

Their meaning and interpretation is the same as explained in Section 2, only the parameter names should remain without the "*permisAuthz-*" prefix. You will need to change the SOA, policy OID, LDAP and the location of the Root CA certificate for real life scenarios.

Note that in so-called "local" authorisation PERMIS is invoked by GT4 as a Policy Decision Point (PDP) with a special name "*permisAuthz*", as specified by the security descriptor. Hence the prefix to the parameter names, so that PERMIS Authorisation can distinguish parameters directed for its own use from the parameters to the actual service. However, when PERMIS is used to deliver access control decisions to remote parties, it acts as a GT4 service, hence there is no prefix to the parameter names, since it is not a GT4 PDP in this case.

5. GridShibPERMIS

The GT4 PERMIS authorisation service can also be used in conjunction with the GridShib for Globus Toolkit plugin which allows users to both query and receive attributes from a GridShib for Shibboleth IDP. The PERMIS authorisation service can then be used to make decisions based on the returned attributes. These instructions assume that the Permis Authorisation Service has already been installed and that there is already an interoperation GridShib 0.5.2 IDP and SP in place. For more information on GridShib please refer to the GridShib website (http://gridshib.globus.org/index.html).

5.1 Configuring the GridShib for Shibboleth IDP for use with the example Policy

In this section we will configure your GridShib for Shibboleth IDP to return permisRole plaintext attributes from the University of Kent's LDAP server (ldap://sec.cs.kent.ac.uk/c=gb) for use with the default policy included with the release package.

Step 1: Add the Test user User0 to the Shibboleth Origin Configuration File gridshib-name-mapping.txt

This file can be found at <$IDP_HOME/etc/gridshib-idp/mappings> in the Shibboleth origin site computer. Add the following entry into it

```
"CN=User0,O=PERMISv5,C=GB" user0
```

Where user0 is the account name of the user whose certificates matches the example certificates provided above.

Step 2: Change The Shibboleth Origin Configuration File resolver.ldap.xml

This file can be found at <$IDP_HOME/etc/> in the Shibboleth origin site computer. Add the following entry into it:

```
<SimpleAttributeDefinition id="permisRole"
smartScope="kent.ac.uk">
    <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>
```

Please note the ***smartScope*** is needed for the example PERMIS policy used in this document. If it is absent here, then the origin site host name will be used as the domain name which may not be recognized by the example PERMIS policy. In this case you should modify the PERMIS policy to allow the origin site host domain name to be recognized.

In the same file, modify the ***JNDIDirectoryDataConnector*** entry as follows:

```
<JNDIDirectoryDataConnector id="directory"
mergeMultipleResults="true">
     <Search filter="uid=&percnt;PRINCIPAL&percnt;">

<ControlssearchScope="SUBTREE_SCOPE"returningObjects="false"/>
     </Search>
     <Property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
     <Property name="java.naming.provider.url"
value="ldap://sec.cs.kent.ac.uk/c=gb"/>
</JNDIDirectoryDataConnector>
```

Note in the above entry that ***uid*** should be used as the search filter and the LDAP should be ldap://sec.cs.kent.ac.uk/c=gb so that the LDAP server at the University of Kent can be used.

Step 3: Change Shibboleth Origin Configuration File arp.site.xml

This file ***arp.site.xml*** can be found at <$IDP_HOME/etc/arps/> in the Shibboleth origin site computer. Add the following entry into it:

```
<Attribute name="urn:mace:dir:attribute-def:permisRole">
  <AnyValue release="permit"/>
</Attribute>
```

These steps will allow the Shibboleth origin site to recognise the test user and to release the ***permisRole*** attribute to the target site.

Note: in order to make use of ***resolver.ldap.xml***, make sure that in the Shibboleth configuration file ***idp.xml*** the following entry is present:

resolverConfig="/conf/resolver.ldap.xml"

5.2 Configuring the GridShib GT4 service for use with the example Policy

In this section we will configure the shutdown method of the GT4 container to use both the GridShib for GlobusToolkit GT4 plugin and the Permis Authorisation Service to make decisions based on user attributes.

Step 1: Configure the Shutdown method to use the GridShib PIP

The Permis Authorisation service uses the GridShib PIP (org.globus.gridshib.PIP) to query attributes from configured IDPs. In this guide we will focus on the use of

SAML2 metadata to configure the GridShib PIP but if you wish to configure the GridShib PIP without the use of an IDP Metadata file please refer to Table1 in the GridShib Admin guide (http://gridshib.globus.org/docs/gridshib-gt-0.5.2/admin-index.html#shib-wsdd-parameters) for more details.

Now you will need to update the configuration of the *ShutdownService* in order to allow the GridShib PIP to fetch Attributes for it. To do that you will need to update the *etc/globus_wsrf_core/server-config.wsdd* file in the following way:

```
<service name="ShutdownService" provider="java:RPC"
    use="literal" style="document">
...
        <!-- GridShib PIP Attribute Query Parameters -->

        <parameter name="shibAuthn-metadata-dir" value="{PATH TO IDP
METADATA FOLDER}"/>
        <parameter name="shibAuthn-SPproviderId" vale="{SP Provider
ID}"/>

...
</service>
```

The first variable metadata-dir is used to trigger the PIP into using a metadata file to obtain the AA information instead of using explicit configurations and should point to a folder which contains a metadata file defining the IDP you wish to use. The second parameter SPproviderId is used to set the value of the resource attribute of the SAML <AttributeQuery> element in the attribute query sent to the Shib AA. This value should correspond to the value configured in your IDP's service provider metadata.

Please Note: The scope (shibAuthn) of the parameter is used to identity which element of the service's Security Descriptor the parameter is used by.

Step 2: Configure the Shutdown method to use Grid Shib and PERMIS

Now you will need to update the configuration of the *ShutdownService* so that PERMIS Authorisation will make access control decisions based on the attributes returned from the GridShib IDP. To do that you will need to update the *etc/globus_wsrf_core/server-config.wsdd* file in the following way:

```
<service name="ShutdownService" provider="java:RPC"
    use="literal" style="document">
...
        <!-- Permis GridShib Security Descriptor -->

        <parameter name="securityDescriptor"
          value="etc/permisAuthzGT4/permis-gridshib-security-
descriptor.xml"/>


        <!-- GridShib PIP Attribute Query Parameters -->

        <parameter name="shibAuthn-metadata-dir" value="{PATH TO IDP
METADATA FOLDER}"/>
```

```
        <parameter name="shibAuthn-SPproviderId" vale="{SP Provider
ID}"/>

        <!--  Permis Authz Policy Parameters -->

        <parameter name="permisAuthz-SOA"
          value="cn=A PERMIS Test User,o=PERMISv5,c=GB"/>
        <parameter name="permisAuthz-OID"
          value="1.2.826.0.1.3344810.6.0.0.16"/>
        <parameter name="permisAuthz-LDAP"
          value="ldap://sec.cs.kent.ac.uk/c=gb"/>
        <parameter name="permisAuthz-LDAP_AC_attribute"
          value="attributeCertificateAttribute"/>
        <parameter name="permisAuthz-LDAP_PKC_attribute"
          value="userCertificate;binary"/>
        <parameter name="permisAuthz-ROOT_CA"
          value="etc/permisAuthzGT4/rootca.cer"/>

        <!-- Permis Authz GridShib Parameters -->

        <parameter name="permisAuthz-GSP_useGridShib" value="true" />
        <parameter name="permisAuthz-GSP_LDAP_AC_attribute"
          value="urn:mace:dir:attribute-
def:attributeCertificateAttribute"/>

...
</service>
```

Now we can observe the result instantly: in one console, run

```
bin/globus-start-container
```

in another console, run

```
bin/globus-stop-container
```

If you used the sample key-pair and changed the *ShutdownService* configuration as explained above, you will see that the GT4 container has stopped in the first console, and that there were no error messages in another console. In the background the GridShibPIP will have queried the Shibboleth IDP's attribute resolver and retrieved the users attributes before calling the PERMIS PDP to make a decision based on the returned values.

If you try to use other credentials (i.e. use a different user keypair), access to *ShutdownService* will be denied, because that user will not have a corresponding X.509 Attribute Certificate issued to him with a *RemoteAdministrator* role in it, and the GT4 container will continue to run.

In the following sections we will explain how to configure your services to be protected by GridShibPERMIS, and how to configure PERMIS Authorisation to use the policy that you want.

5.3 Configuring Your Own Service to use GridShibPERMIS

Configuring your service is very simple. In the deployment descriptor of your service you need to refer to the security descriptor that specifies the authorisation to use, and the parameters to the authorisation module. So you may be happy to use the *permis-gridshib-security-descriptor.xml* as in the ShutdownService example above - it specifies that all users must use strong authentication, that the GridShib PIP will be used to resolve attributes and that the PERMIS PDP will be used to make access decisions. It also specifies that parameters used by the GridShib PIP have the scope shibAuthn and that parameters used by PERMIS have the scope permisAuthz. You will need to specify both the *shibAuthn* parameters explained above and the *permisAuthz-** parameters to configure GridShibPERMIS Authorisation for your needs (**remember:** SOA, policy OID, AC repository, LDAP_AC_attribute, LDAP_PKC_attribute, Root CA, GSP_useGridShib) as explained in Section 2.

If you want to specify a more advanced security descriptor, please refer to the GT4 documentation.

5.4 Configuring GridShibPERMIS to use Attribute Certificates pushed from a GridShib IDP

GridShibPERMIS is also capable of making decisions based on role certificates pushed from the GridShib IDP for an additional layer of security. Using X.509 ACs ensures that the user attributes are tamperproof since they are digitally signed by the issuer. These certificates should have been signed by an SOA designated the right to allocate permisRole attributes in the policy and the SOA's public key certificate should be stored in the repository containing your policy.

Step 1: Change The Shibboleth Origin Configuration File resolver.ldap.xml

This file can be found at <$IDP_HOME/etc/> in the Shibboleth origin site computer, where IDP_HOME should be your Shibboleth home directory. Add the following two entries into it:

```
      <SimpleAttributeDefinition id="urn:mace:dir:attribute-def:dn">
               <DataConnectorDependency requires="directory"/>
        </SimpleAttributeDefinition>

      <SimpleAttributeDefinition id="urn:mace:dir:attribute-
def:attributeCertificateAttribute"
            valueHandler="edu.internet2.middleware.shibboleth.aa.attr
      resolv.provider.Base64ValueHandler">
        <DataConnectorDependency requires="directory"/>
      </SimpleAttributeDefinition>
```

In the same file, modify the ***JNDIDirectoryDataConnector*** entry as follows:

```
      <JNDIDirectoryDataConnector id="directory">
        <Search filter="uid=&percnt;PRINCIPAL&percnt;">
               <ControlssearchScope="SUBTREE_SCOPE"
      returningObjects="false"/>
         </Search>
```

```
  <Property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
  <Property name="java.naming.provider.url"
value="ldap://sec.cs.kent.ac.uk/c=gb"/>
  <Property name="java.naming.ldap.attributes.binary"
value="attributeCertificateAttribute" />
</JNDIDirectoryDataConnector>
```

Note in the above entry that ***uid*** should be used as the search filter and the LDAP should be ***ldap://sec.cs.kent.ac.uk/c=gb*** so the LDAP at the University of Kent can be used.

Note also in order to make use of ***resolver.ldap.xml***, make sure that in the Shibboleth configuration file ***idp.xml*** the following entry is present:

resolverConfig="/conf/resolver.ldap.xml"

Step 2: Change Shibboleth Origin Configuration File arp.site.xml

This file ***arp.site.xml*** can be found at <$IDP_HOME/etc/arps/> in the Shibboleth origin site computer. Add the following two entries into it:

```
<Attribute name="urn:mace:dir:attribute-
def:attributeCertificateAttribute">
  <AnyValue release="permit"/>
</Attribute>


<Attribute name="urn:mace:dir:attribute-def:dn">
          <AnyValue release="permit"/>
</Attribute>
```

Steps 1 and 2 will allow the Shibboleth origin site to release the ***dn*** and ***attributeCertificateAttribute*** attributes to the target site.

Note In this configuration the returned attributes have the name "*urn:mace:dir:attribute-def:attributeCertificateAttribute*" if the name of this attribute is changed at the origin site then the ***GSP_LDAP_AC_attribute*** parameter should be updated in the services *server-config.wsdd*.

After the above configurations, restart Shibboleth at the target site.

6. Advanced Options

6.1 Enabling Debug Statements

You can set a debug directive in the container-log4j.properties file located in the ***$GLOBUS_LOCATION*** directory to let the PERMIS Authorisation Service output debug information. There are six different levels of Debug statements in increasing order of information that is output: OFF, FATAL, ERROR, WARN, INFO and DEBUG. In order to use debug mode you should insert one of the following directives

```
Log4j.categorg.issrg=OFF
Log4j.categorg.issrg=FATAL
Log4j.categorg.issrg=ERROR
Log4j.categorg.issrg=WARN
Log4j.categorg.issrg=INFO
Log4j.categorg.issrg=DEBUG
```

into the container-log4j.properties file. This will switch on different levels of PERMIS debugging. Unless specifically stated by the user then the logging of debug statements is disabled.

The six levels of debugging are as follows:

OFF - Debugging is turned off
FATAL - prints out serious fatal error messages that will lead to the program terminating
ERROR - prints out serious non fatal errors
WARN - prints out less important errors
INFO - prints out statements at initialisation to show the current configuration, and pertinent information about access attempts
DEBUG - prints out all the verbose debug information in the PERMIS Authorisation Service.

Last updated by George Inman on 26 March 2007.