Technical Note

Policy Manager 7.x Delegate for Apache Axis

SOA Software, Inc.

12100 Wilshire Blvd, Suite 1800
Los Angeles, CA 90025
866-SOA-9876
www.soa.com
info@soa.com

Table of Contents

1 Introduction

This document describes SOA Software Axis Delegate and Apache client solution.

The following topics are covered:

- Delegate Architecture
- Delegate Handler
- Configuration Handler Parameters
- Delegate Identity Credentials

2 SOA Software Axis Delegate

The *SOA Software Axis Delegate* and Apache Axis client solution provides message formatting and transportation based on a web service description stored in the SOA Software Policy Manager product. The solution is implemented by a set of Axis message handlers that work in conjunction with an "SOA Container." The SOA Container is an OSGi container that can support multiple features. When used as part of the Delegate solution, the SOA Software Delegate feature must be installed.

2.1 Delegate Architecture

The architecture of the Delegate supports two different deployment options:

- **Collocated Option** – Using this option, the SOA Container is deployed in the same Java Runtime Engine (JRE) as the Axis Engine.

- **Remote Option** – Using this option the SOA Container is deployed in a separate JRE from the Axis Engine.

In the collocated deployment, the com.soa.delegate.axis.LocalAxisPivotHandler is installed as an Axis "pivot" handler. It is important that the handler is installed as a "pivot" handler because it will take responsibility for the transportation of the web service messages. In this deployment the transport is actually performed by the *SOA Software Delegate* feature in the collocated SOA Container. The LocalAxisPivotHandler merely relays the web service messages between the Axis engine and the SOA Container.

2.2 Deploy Delegate Handler

To deploy a Delegate Handler a handler entry is made in the client-config.wsdd file of the Axis client. The handler is then referenced as the pivot handler. For example:

```
01) <deployment ...>
02)   ...
03)   <handler name="soapivot" type="java:com.soa.delegate.axis.LocalAxisPivotHandler>
04)     ...
05)   </handler>
06)   <transport name="http" pivot="soapivot"/>
07)   ...
08) </gwap:Report>

Lines 03 – 05 hold the LocalAxisPivotHandler declaration and configuration parameters.
Line 06 defines the LocalAxisPivotHandler as the Axis "pivot" handler.
```

2.3 Configuration Handler Parameters

The configuration parameters of the Handler are described using name/value pair elements within the handler element. Some of the parameters describe the scope of the handler and are described in the following table. These configuration parameters apply to both the LocalAxisPivotHandler and the RemoteAxisPivotHandler.

Parameter Name	Parameter Description
service_ns	Namespace of the service the handler is processing messages for.

	This should be the same namespace as the service element in the WSDL document describing the service in Policy Manager. This parameter (in conjunction with the service_name parameter) or the binding_identifier parameter is required.
service_name	Unqualified name (local part of the qualified name) of the service the handler is processing messages for. This should be the same local part as the service element in the WSDL document describing the service in Policy Manager. This parameter (in conjunction with the service_ns parameter) or the binding_identifier parameter is required.
binding_identifier	Unique identifier of the service to be used as an alternative to the service_ns and service_name parameters. See the Policy Manager Online Help for more information on binding identifiers. This parameter or the service_ns and service_name parameters are required.
interface_ns	Namespace of the interface the handler is processing messages for. This should be the same namespace as the portType element in the WSDL document describing the service in Policy Manager. This parameter is optional.
interface_name	Unqualified name (local part of the qualified name) of the interface the handler is processing messages for. This should be the same local part as the portType element in the WSDL document describing the service in Policy Manager. This parameter is optional.

In the *remote* deployment, the com.soa.delegate.axis.RemoteAxisPivotHandler is installed as an Axis "pivot" handler. The RemoteAxisPivotHandler will relay messages between the Axis engine and the remote SOA Container which will perform the actual message transportation to the service implementation.

- If the remote SOA Container is a standalone container or when there is no native delegate, the *SOA Software Delegate Access Point* feature is installed to perform the message processing.

- If the remote SOA Container is a J2EE container, the *SOA Software Delegate* feature is installed to perform the message processing as it can act as an access point as well as a collocated delegate.

The deployment of a remote access point is ideal for Axis clients that are not deployed in a J2EE container as the SOA Container required to run the Delegate feature requires a J2EE container or its own container to function.

In addition to the parameters described for the LocalAxisPivotHandler, the RemoteAxisPivotHandler has an additional parameter named proxy.address. This parameter holds a URL to the remote Delegate container.

The following is an example of the parameter usage:

```
01) <handler ...>
02)   <parameter name="service_ns" value="http://soapinterop.org/"/>
03)   <parameter name="service_name" value="echo"/>
```

```
04)   <parameter name="interface_ns" value="http://soapinterop.org/"/>
05)   <parameter name="interface_name" value="InteropTestPortType"/>
06)   <parameter name="proxy.address" value="http://host1:9905/soapdelegate"/>
07)   ...
08) </handler>
```

2.4 Delegate Identity Credentials

The Delegate supports multiple forms of identity credentials that can be provided by the client to be used in messages. The format of the credentials included in the messages is dictated by the policies of the endpoint being communicated with.

Multiple sets of credentials may be required for communications with an endpoint. For example, the X.509 certificate for the client application may be required for HTTPS communication and a username token for the logged in user may be required in the SOAP message. These credentials are distinguished by a customizable categorization scheme described using URI's. Two standard categories that also serve as examples are urn:org:federatedgovernance:security:subject-category:consumer and urn:org:federatedgovernance:security:subject-category:enduser.

The Axis Delegate integrates with the client to obtain credentials through JAAS Callback Handlers. Some out-of-the-box Callback Handlers are provided with the Axis Delegate, however Callback Handlers developed by third parties can also be used. The Delegate instantiates the Callback Handlers using Callback Handler Factories. Each credential that will be provided to the AxisPivotHandler will be associated with a Callback Handler Factory. The factory creates the Callback Handler that will provide the credentials to the Delegate. Each Callback Handler Factory may have its own set of parameters. For example, a factory that creates a Callback Handler to retrieve X.509 credentials from a Java KeyStore will have parameters for the alias to retrieve information for and the password to the keystore.

The configuration of Callback Handler Factories and their association with credential categories is described using additional parameters in the handler element. First, credential categories are listed in a parameter. These categories are identified with symbolic names, not URI's. For example, instead of listing "urn:org:federatedgovernance:security:subject-category:enduser", simply "user" may be listed. All authentication credentials parameters are prefixed with "auth." The credential categories are listed using a parameter named "auth.categories." The categories are listed in the parameter value delimited by commas.

For each category a set of parameters will be supplied. The parameters for each category listed in "auth.categories" are prefixed with auth.<category name>. For example, for the "user" category, all parameters are prefixed with "auth.user." All credential categories must have at a minimum the "uri" and "factory" parameters. The "uri" parameter specifies the literal category name as a URI, such as "urn:org:federatedgovernance:security:subject-category:enduser." The "factory" parameter specifies the Callback Handler Factory class that will be used to obtain the credentials. Any number of additional parameters specific to each factory can also be listed. These additional parameters must be prefixed with the same prefix as the factory parameter.

Three Callback Handler Factories are provided with the Axis Delegate, com.soa.delegate.auth.callback.handler.X509KeyStoreCBHandlerFactory, com.soa.delegate.auth.callback.handler.UserPropertiesCBHandlerFactory, and urn com.soa.delegate.auth.callback.handler.AgentSubjectPropertiesCBHandlerFactory. The UserPropertiesCBHandlerFactory retrieves a username and password from properties, or parameters, named "username" and "password" respectively. The following is an example of a UserPropertiesCBHandlerFactory configuration.

```
<handler ...>
  ...
  <parameter name="auth.categories" value="user"/>
  <parameter name="auth.user.uri" value="urn:org:federatedgovernance:security:subject-category:enduser"/>
  <parameter name="auth.user.factory" value="com.soa.delegate.auth.callback.handler.UserPropertiesCBHandlerFactory"/>
  <parameter name="auth.user.username" value="jane"/>
  <parameter name="auth.user.password" value="doe"/>
  ...
</handler>
```

The X509KeyStoreCBHandlerFactory retrieves an X.509 private key and certificate from a Java keystore. It takes the following parameters:

- keystore.path – Path to the keystore to retrieve the information from.
- keystore.alias – Alias of the private key certificate pair.
- keystore.password – Password required to open the keystore.

The following is an example of an X509KeyStoreCBHandlerFactory configuration.

```
<handler ...>
  ...
  <parameter name="auth.categories" value="app"/>
  <parameter name="auth.app.uri" value="urn:org:federatedgovernance:security:subject-category:consumer"/>
  <parameter name="auth.app.factory" value="com.soa.delegate.auth.callback.handler.X509KeyStoreCBHandlerFactory"/>
  <parameter name="auth.app.keystore.path" value=".keystore"/>
  <parameter name="auth.app.keystore.alias" value="jdoe"/>
  <parameter name="auth.app.keystore.password" value="password"/>
  ...
</handler>
```

The AgentSubjectPropertiesCBHandlerFactory retrieves credentials from a JAAS Subject that was created from an Agent as a result of authenticating credentials in an inbound message. The AgentSubjectPropertiesCBHandlerFactory has only one parameter named "category." This parameter lists the literal credential category to pull from the JAAS Subject. The following example illustrates using the inbound consumer credentials for the outbound enduser.

```
 <handler ...>
  ...
  <parameter name="auth.categories" value="agent"/>
  <parameter name="auth.agent.uri" value="urn:org:federatedgovernance:security:subject-category:enduser"/>
```

```
05)   <parameter name="auth.agent.factory"
value="com.soa.delegate.auth.callback.handler.AgentSubjectPropertiesCBHandlerFactory"/>
06)   <parameter name="auth.agent.category"
value=":org:federatedgovernance:security:subject-category:consumer"/>
07)   ...
08) </handler>
```

In the above example, line 04 states that the credentials retrieved from this Callback Handler will be used for the enduser credentials. Line 06 states the consumer credentials in the Subject created by the Agent should be retrieved for this purpose.

In the Axis architecture there is only one "pivot" handler for all services consumed by the client. As described previously, the Delegate handlers require parameters that provide the context for the service being consumed. Specifying these parameters in the "pivot" handler will restrict the client to consuming only the one service. If the client consumes more than one service, the SOA pivot handlers must be used in conjunction with the com.soa.delegate.axis.AxisServiceHandler. The sole purpose of the AxisServiceHandler is to provide the consumed service context to the pivot handler so that the pivot handler does not have to be tied to just one consumed service. When deployed the AxisServiceHandler will be configured with the following parameters already described above and the pivot handler (LocalAxisPivotHandler or RemoteAxisPivotHandler) will not: service_ns, service_name, binding_identifier, interface_ns, interface_name, and all authentication parameters. If preferred, the authentication parameters can be defined for the pivot handler even when the service handler is present and configured with service context parameters.

The AxisServiceHandler is configured in the Axis client wsdd file within the "service" element. The following is an example of the configuration.

```
01) <deployment ...>
02)   ...
03)   <handler name="soapivot" type="java:com.soa.delegate.axis.LocalAxisPivotHandler>
04)     ...
05)   </handler>
06)   <transport name="http" pivot="soapivot"/>
07)   <service name="..." provider="...">
08)     ...
09)     <requestFlow>
10)       <handler name="soaservice" type="java:com.soa.delegate.axis.AxisServiceHandler>
11)         <parameter name="binding_identifier" value="myservice"/>
12)       </handler>
13)     </requestFlow>
14)     ...
15)   </service>
16)   ...
17) </gwap:Report>

Lines 03 – 05 hold the LocalAxisPivotHandler declaration and configuration
parameters. These parameters will not include any service context. Lines 07 – 15
hold a service declaration. Lines 10 – 12 hold the AxisServiceHandler
declaration and configuration parameters. Line 11 specifies that the service is
identified in Policy Manager with a binding identifier of "myservice."
```

3 About SOA Software

SOA Software is a leading provider of unified SOA governance and API Management products that enable organizations to successfully plan, build, and run enterprise services and Open APIs. The world's largest companies including Bank of America, Verizon, and Pfizer use SOA Software solutions to transform their business. For more information, please visit http://www.soa.com.