New Features Guide EAServer 6.2

Document ID: DC38032-01-0620-01

Last revised: June 2008

Methods As Stored Procedures interface

EAServer 6.2 supports the Methods As Stored Procedures (MASP) interface which allows you to execute component methods as if they were database stored procedures.

Each MASP invocation creates a component instance, invokes the method, and destroys the component instance.

Tabular Data Stream™ (TDS) and MASP clients access EAServer through the TDS listener. EAServer 6.2 uses TDS version jTDS 6.0. The default EAServer jTDS listener port number is 2005.

HTTP session ID encryption strength

EAServer provides an autogenerated HTTP session ID. In versions earlier than 5.*x*, the encryption key strength of an HTTP session ID is 64-bit. The 64-bit session identifier does not provide a secured HTTP session.

In EAServer 6.2, the algorithm increases the strength of encryption for a HTTP session ID to 128-bit. The default length of the HTTP session ID is 64-bit. Use the Web Management Console to modify the length to a multiple of 8 between 64 and 2048.

See the *EAServer 6.0 Security Administration and Programming Guide*.

Web Service Security for Java

EAServer 6.2 implements the Web Service Security (WS-Security) specification with Web Services Security for Java (WSS4J) framework to secure Web services. WSS4J includes special support for Axis and can be deployed in any application server.

WSS4J includes preconfigured handlers that you can use to integrate with Axis-based Web services. *WSDoAllSender* and *WSDoAllReceiver* are the main Axis handlers for creating and interpreting secure SOAP requests.

To add a WS-Security layer to the Web service, add *WSDoAllSender* and *WSDoAllReceiver* handlers to a Web Service Deployment Descriptor (*.wssd*) file. The WSSD provides the specified information to control the security processing.

Deploying WSS4J Axis handlers

On the server side, modify the deployment descriptor file to deploy the *WSDoALLReceiver* handler. For example:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
   xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
     <service name="stock-wss-01" provider="java:RPC" style="document" use="literal">
        <requestFlow>
          <handler type="java:org.apache.ws.axis.security.WSDoAllReceiver">
             <parameter name="passwordCallbackClass" value="PWCallback"/>
             <parameter name="action" value="UsernameToken"/>
           </handler>
         </requestFlow>
            <parameter name="className"
                   value="samples.stock.StockQuoteService"/>
            <parameter name="allowedMethods" value="getQuote"/>
            <parameter name="scope" value="application"/>
      </service>
</deployment>
```

On the client side, create the deployment descriptor file (*client_deploy.wsdd*) for deploying the *WSDoALLSender* handler for the SOAP request. For example:

```
<!-- define the service, use the WSDoAllSender security handler in request flow -->
  <service name="Ping1">
       <requestFlow>
           <handler type="java:org.apache.ws.axis.security.WSDoAllSender">
               <parameter name="action" value="UsernameToken"/>
               <parameter name="user" value="werner"/>
               <parameter name="passwordType" value="PasswordText" />
               <parameter name="passwordCallbackClass"
                       value="org.apache.ws.axis.oasis.PWCallback1"/>
            </handler>
        </requestFlow>
  </service>
```

Axis parses the deployment descriptor and provides the parameters and their value to the handler. Each service can have its own request and response flow definition, which provides a flexible setup of the security parameters.

Table 1: WSDoAllSender handler parameters

Parameter	Description
action	Defines the security action. The valid value is "`UsernameToken`", which directs the handler to insert a token into the SOAP request.
user	Specifies the user name to include in the token.
passwordType	Defines the encoding type of password for the UsernameToken. The valid values are: • PasswordText – sends the password in plain text. • PasswordDigest – sends the password in digest mode.
passwordCallbackClass	Contains the name of a class that implements a method to get the user's password.

For more information, see:

- WSS Username Token Profile at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

- PW_CALLBACK_CLASS at http://ws.apache.org/wss4j/xref/org/apache/ws/security/handler/WSHandlerConstants.html
- OASIS Web Services Security (WSS) TC at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss