1 Appendix B – Configuration

1.1 Setting up WS-Security

The user credentials are transferred between client application and framework via WS-Security on each webservice call.

For this, WS-Security has to be set up by adding a file `axis-wss4j.xml` to folder `merge` with the following content:

```
<requestFlow>
      <handler type="java:org.apache.ws.axis.security.WSDoAllReceiver">
            <parameter name="action" value="UsernameToken"/>
            <parameter name="passwordCallbackClass"
            value="de.fiz.escidoc.common.util.security.server.PWCallback"/>
            <parameter name="passwordType" value="PasswordText" />
            <parameter name="addUTElement" value="NonceCreated" />
      </handler>
</requestFlow>
```

When executing the EJB XDoclet task, this `requestFlow` definition will be merged into each webservice deployment descriptor.

Note: WS-Security requires the WSS4J implementation. Make sure the jar file `wss4j.jar` is contained in `escidoc.ear/axis.war/WEB-INF/lib`.

1.2 Setting up JBoss AOP

Authentication / Authorization of the current user is realized by AOP interception whenever an EJB is calling a method in one of its service classes.

JBoss AOP interception has to be set up in the eSciDoc framework via the following steps:

- Create new folder `escidocAOP.aop` in folder `escidoc.ear`.
- Create new folder `META-INF` in `escidoc.ear/escidocAOP.aop`.
- Add file `jboss-aop.xml` to `escidoc.ear/escidocAOP.aop/META-INF` with the following content:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<aop>
    <pointcut expr="call(* de.fiz.escidoc.*.service.*-&gt;*(..))
       AND !call(* de.fiz.escidoc.common.*-&gt;*(..))
       AND within(de.fiz.escidoc.*.ejb.*Bean)" name="authorisation"/>
    <bind pointcut="authorisation">
        <interceptor class=
              "de.fiz.escidoc.common.util.security.aop.SecurityInterceptor"/>
    </bind>
</aop>
```

This defines the AOP pointcut `authorization` as follows:

- `call(* de.fiz.escidoc.*.service.*->;*(..))`: intercept any call to any method that is located in any class in a package that matches “de.fiz.escidoc.*.service.*”
- `AND !call(* de.fiz.escidoc.common.*-&gt;*(..))`: but don’t intercept calls into the common-package, since it only contains helper-classes
- `AND within(de.fiz.escidoc.*.ejb.*Bean)`: and only intercept in case the call originates from a class named [*]Bean (which is the naming convention for EJBs in the eSciDoc project) residing in a package that matches “de.fiz.escidoc.*.ejb.*”

This pointcut is the bound to a call into Interceptor `de.fiz.escidoc.common.util.security.aop.SecurityInterceptor`.

- Add new component `escidocAOP.aop` to `Escidoc/escidoc.ear/META-INF/application.xml`:

```
<module>
      <java>escidocAOP.aop</java>
</module>
```

1.3 Invoking JBoss AOP

JBoss AOP interception is achieved by bytecode manipulation. This means that the JBoss AOP compiler has to re-compile the class files, that are residing in `escidoc.ear` (which are already compiled by the Java compiler).

For this, the following Ant-task has been set up in `build/helper-build.xml`:

```
<!-- Compile the sources of the project -->
<target name="aop_compile">
    <echo level="debug">Compile (AOP) the classes</echo>

    <!-- Get the classpath information from the .classpath file of the project -->
    <getEclipseClasspath workspace=".."
        projectname="Escidoc"
                 classpathId="eclipse.classpath" />


    <!-- Taskdefinition for the AOP compiler -->
    <taskdef name="aopc" classname="org.jboss.aop.ant.AopC"
        classpathref="others.classpath" />
    <!-- compile aop -->
    <aopc compilerclasspathref="aop.classpath" classpathref="eclipse.classpath"
          verbose="true">
        <classpath path="eclipse.classpath" />
        <src path="escidoc.ear"/>
        <include name="*.jar/de/fiz/escidoc/*/ejb/*Bean.class"/>
        <aoppath path="escidoc.ear/escidocAOP.aop/META-INF/jboss-aop.xml" />
    </aopc>
</target>
```

It recompiles all class files that are residing in `escidoc.ear/*.jar/de/fiz/escidoc/*/ejb` (where * means any number of characters) and which match the name `*Bean.class`.

This task has of course to be executed *after* the class files have been generated. This either happens by executing the compile-task (`build/java-build.xml`, task `build`) or by using Eclipse's built-in compiler (e.g. by invoking "Refresh" after the executing XDoclet).

However, the AOP compiler has to be executed *before* deploying the ear file. After the AOP compiler run also a "Refresh" should be done on the folder `escidoc.ear` to notify Eclipse of the applied changes (the execution of the Ant-task can be set up to automatically invoke refresh after completion).

Note: The recompilation only affects the EJB classes. All other classes are left as created by the Java compiler and therefore a change in those classes does not require a new run of the AOP compiler or a redeployment. However, whenever changing an EJB (either directly – which should actually never happen – or via recreating it via XDoclet) the ear file should be undeployed, the AOP compiler task should be executed and then a redeployment should happen.

Once the Security concept has been finalized, policies should be created for all Webservice methods in the eSciDoc framework. At this point, the AOP compiler should be included in the Cruise Control build script to have Cruise Control test the Security aspect as well.

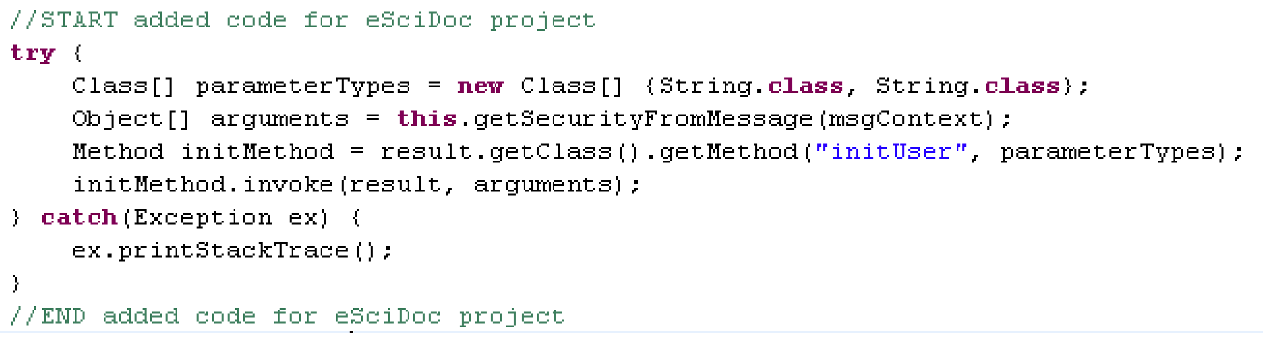
1.4 Adapting Axis to invoke initUser on EJB-create

The user credentials are as described above handed over to the framework via WS-Security. However, since Axis is accessing the system via EJB calls, the WS-Security information is not accessible any more once inside an EJB method.

Therefore the user credentials have to be injected into each EJB via calling the method `initUser`, which exists for each EJB of the eSciDoc framework (the method is defined in the XDoclet template `build/xdoclet/bean.xdt`, which is used to create the EJBs from the service classes).

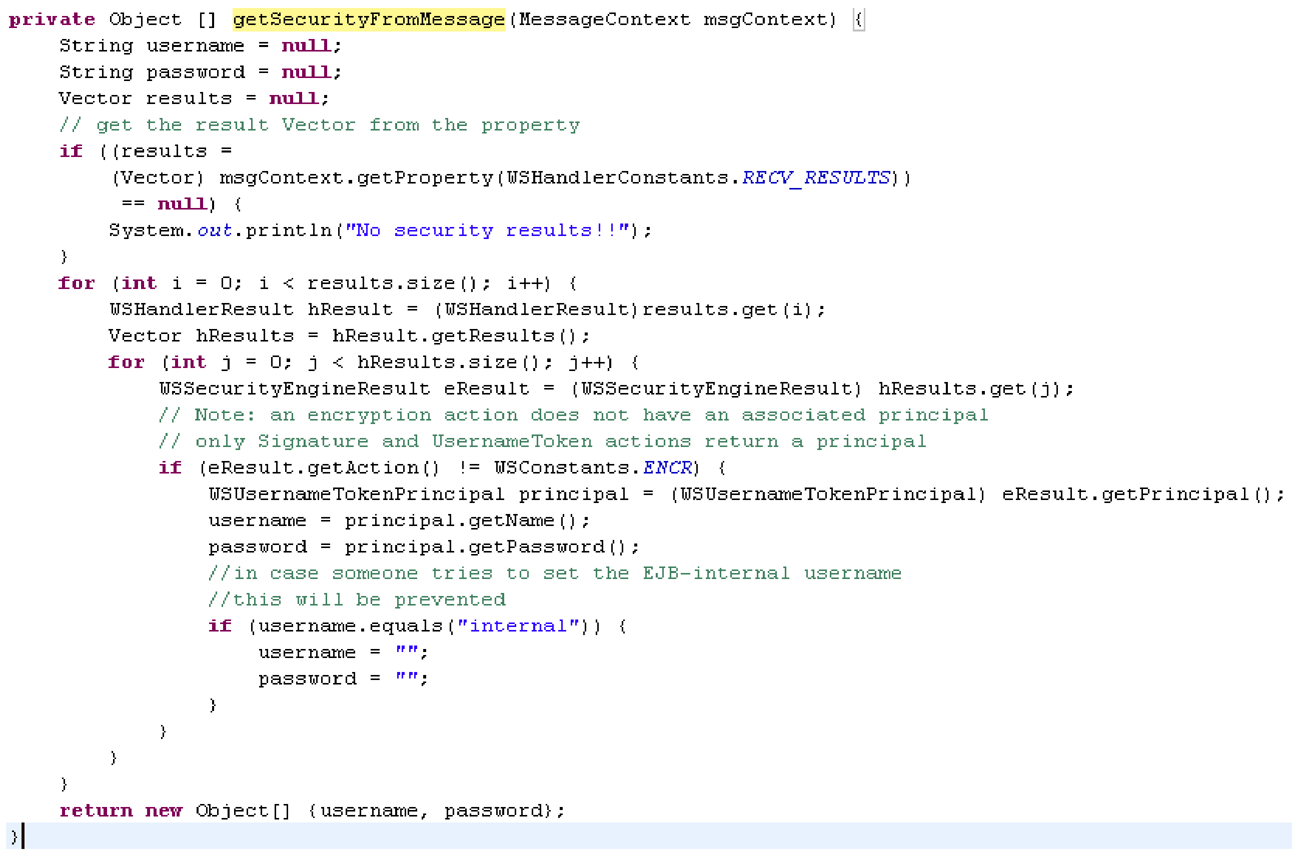
The method should be invoked directly after the create-method of the EJB is invoked and before any other EJB method is invoked. For this the Axis class `EJBProvider`, which resides inside `escidoc.ear/axis.war/WEB-INF/lib/axis-1.2.1.jar` has to be changed in the following way:

- Add the following code directly before the return-statement in method `createRemoteEJB` and in method `createLocalEJB`:



```
//START added code for eSciDoc project
try {
    Class[] parameterTypes = new Class[] {String.class, String.class};
    Object[] arguments = this.getSecurityFromMessage(msgContext);
    Method initMethod = result.getClass().getMethod("initUser", parameterTypes);
    initMethod.invoke(result, arguments);
} catch(Exception ex) {
    ex.printStackTrace();
}
//END added code for eSciDoc project
```

- Add private method `getSecurityFromMessage`:



```
private Object [] getSecurityFromMessage(MessageContext msgContext) {
    String username = null;
    String password = null;
    Vector results = null;
    // get the result Vector from the property
    if ((results =
        (Vector) msgContext.getProperty(WSHandlerConstants.RECV_RESULTS))
         == null) {
        System.out.println("No security results!!");
    }
    for (int i = 0; i < results.size(); i++) {
        WSHandlerResult hResult = (WSHandlerResult)results.get(i);
        Vector hResults = hResult.getResults();
        for (int j = 0; j < hResults.size(); j++) {
            WSSecurityEngineResult eResult = (WSSecurityEngineResult) hResults.get(j);
            // Note: an encryption action does not have an associated principal
            // only Signature and UsernameToken actions return a principal
            if (eResult.getAction() != WSConstants.ENCR) {
                WSUsernameTokenPrincipal principal = (WSUsernameTokenPrincipal) eResult.getPrincipal();
                username = principal.getName();
                password = principal.getPassword();
                //in case someone tries to set the EJB-internal username
                //this will be prevented
                if (username.equals("internal")) {
                    username = "";
                    password = "";
                }
            }
        }
    }
    return new Object[] {username, password};
}
```

A changed version of this class can be found in `common/src/org.apache.axis.providers.java`. This class has to be compiled and then copied to `axis-1.2.1.jar` into the same package to replace the existing class-file.

1.5 Adapting Spring to invoke initUser on EJB-create

Similar to Axis, also Spring has to be extended to invoke `initUser` when it creates a new EJB.

For this the classes `LocalStatelessEjbProxyFactoryBean` and `RemoteStatelessEjbProxyFactoryBean` have been created, which are located in `common/src/de.fiz.escidoc.common.util.service`. The eSciDoc-specific Spring mapping has been set up to use these classes instead of the standard Spring EJB proxies when accessing EJBs.