How To...
Create Modules for the J2EE Adapter Engine
Version 1.00 – September 2005

Applicable Releases:
SAP NetWeaver '04
Exchange Infrastructure 3.0
Content
1 Scenario .............................................................................................................................. 2
2 Introduction ..................................................................................................................... 2
3 The Step-By-Step Solution .............................................................................................. 3
  3.1 Setting Up the Developer Studio ............................................................................. 3
  3.2 Creating the Java Class ............................................................................................ 5
  3.3 Providing the EJB Environment ............................................................................. 6
  3.4 Adding Code to the Module ................................................................................. 8
  3.5 Creating a Java Archive .......................................................................................... 8
  3.6 Creating an Enterprise Application Archive (EAR) ............................................... 9
  3.7 Using the Module ................................................................................................. 11
4 Appendix ...................................................................................................................... 12
  4.1 Documentation for Module Development ............................................................ 12
  4.2 Documentation of the Resource Adapter API ....................................................... 12
  4.3 Sample Module .................................................................................................... 12
  4.4 Code Samples ....................................................................................................... 12
    4.4.1 Reading File Names ..................................................................................... 12
    4.4.2 Writing Audit Log Entries ........................................................................... 13
1 Scenario

You want to modify the message payload using an Adapter Engine module, but there is no standard module that meets your requirements.

2 Introduction

A message from the Integration Server arrives in the Adapter Framework and is processed by the messaging service. Based on the receiver information, the corresponding module chain is selected for further processing.

The Adapter Framework contains two default module chains: one for the sender/inbound direction and one for the receiver/outbound direction. You can use these default module chains for the adapter if the entire message processing is executed within the adapter. You can enhance the default module chains with customer-specific modules.

The module processor controls the steps in the module chain by calling generic and, if defined, adapter-specific modules. The last module in the module chain forwards the message to the adapter. The adapter transfers the message to the connected system.

Message processing in the sender/inbound direction proceeds in a similar way. In this case, the adapter calls the module processor in the form of an EJB 2.0 local session bean and transfers the message object either as an XI message or in its own format. In the case of the latter, the conversion to an XI message must then take place in an adapter-specific module.

Configuration of the adapter modules is part of the communication channel in the Integration Directory or Partner Connectivity Kit. If you want to add your own modules to adapters, you integrate these modules at the relevant location in the module processor.
3 The Step-By-Step Solution

This manual guides you through the development of a simple adapter module with the help of the SAP NetWeaver Developer Studio. Make sure that you use the latest version of the Developer Studio.

By way of example, we explain how to develop an adapter module that creates an additional attachment. This is something that might be useful in the mail adapter.

3.1 Setting Up the Developer Studio

You need several libraries to compile your module.

The easiest way of getting these libraries is by downloading the latest patch for XI ADAPTER FRAMEWORK CORE 3.0 from SAP Service Marketplace.

Open the file using WinZip and extract aii_af_lib.sda, aii_af_svc.sda and aii_af_cpa_svc.sda.

Open the file aii_af_lib.sda using WinZip and extract the following library files:

- aii_af_cci.jar
- aii_af_mp.jar
- aii_af_ms_api.jar
- aii_af_ms_spi.jar
- aii_af_trace.jar

Extract the following library file from aii_af_svc.sda:

- aii_af_svc.jar

Extract the following library file from aii_af_cpa_svc.sda:

- aii_af_cpa.jar

(You can also find the SDA files on your Adapter Framework installation.)

Steps in SAP NetWeaver Developer Studio

- Open SAP NetWeaver Developer Studio.
- Create a new Project as J2EE -> EJB Module.
- Choose Project -> Properties -> Java Build Path -> Libraries to assign the libraries to your project.
- Create a package for your Java classes.

Your J2EE Explorer view in Developer Studio should look as follows:
3.2 Creating the Java Class

Even if you need a stateless session bean as an adapter module, it is useful not to let the Developer Studio create it. Instead, create a new java class and attach the following code template to the class (check that the package and class name are correct):

```java
package sample;

import javax.ejb.CreateException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

import com.sap.aii.af.mp.module.*;
import com.sap.aii.af.ra.ms.api.*;

/**
 * @ejbHome <{com.sap.aii.af.mp.module.ModuleHome}>
 * @ejbLocal <{com.sap.aii.af.mp.module.ModuleLocal}>
 * @ejbLocalHome <{com.sap.aii.af.mp.module.ModuleLocalHome}>
 * @ejbRemote <{com.sap.aii.af.mp.module.ModuleRemote}>
 * @stateless
 */
public class CreateAttachment implements SessionBean, Module{
    private SessionContext myContext;

    public void ejbRemove() {
    }

    public void ejbActivate() {
    }

    public void ejbPassivate() {
    }

    public void setSessionContext(SessionContext context) {
        myContext = context;
    }

    public void ejbCreate() throws CreateException {
    }

    public ModuleData process(ModuleContext moduleContext, ModuleData inputModuleData) throws ModuleException {
        return inputModuleData;
    }
}
```

Comment lines such as `@ejbHome` are required to enable the Developer Studio to create the ejb-jar.xml file automatically.
3.3 Providing the EJB Environment

Your class should appear in the navigation tree under **EJB Candidates**. In the context menu, choose **Add to ejb-jar.xml**.

**ejb-jar.xml** should look as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc./DTD Enterprise JavaBeans 2.0//EN"
 "http://java.sun.com/dtd/ ejb-jar_2_0.dtd">
<ejb-jar>
  <description>EJB JAR description</description>
  <display-name>EJB JAR</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>CreateAttachment</ejb-name>
      <home>com.sap.ail.ai.mp.module.ModuleHome</home>
      <remote>com.sap.ail.ai.mp.module.ModuleRemote</remote>
      <local-home>com.sap.ail.ai.mp.module.ModuleLocalHome</local-home>
      <local>com.sap.ail.ai.mp.module.ModuleLocal</local>
      <ejb-class>sample.CreateAttachment</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```
You should assign a JNDI name in ejb-j2ee-engine.xml. You will need this name for the configuration of the module. An example of this is the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-j2ee-engine SYSTEM "ejb-j2ee-engine.dtd">
<ejb-j2ee-engine>
  <enterprise-beans>
    <enterprise-bean>
      <ejb-name>CreateAttachment</ejb-name>
      <jndi-name>CreateAttachment</jndi-name>
      <session-props/>
    </enterprise-bean>
  </enterprise-beans>
</ejb-j2ee-engine>
```

After setting up the EJB environment, you receive the error message: *Bean problem: No interface classes found.* This occurs because there is no Java source available for the EJB interface classes.

- Choose the *Package Explorer* view in the Developer Studio.
- Choose your project, and in the context menu, choose *Close Project*.
- Choose *Open Project* again.
- Finally, switch back to the *J2EE Explorer* view.
3.4 Adding Code to the Module
You can now add a short example code. You can change the process method for this purpose:

```java
public ModuleData process(ModuleContext moduleContext,
                          ModuleData inputModuleData)
    throws ModuleException {

    // create a second attachment for the receiver mail adapter
    try {
        // get the XI message from the environment
        Message msg = (Message) inputModuleData.getPrincipalData();
        // create a new payload
        TextPayload attachment = msg.createTextPayload();
        // provide attributes and content for the new payload
        attachment.setName("Attachment");
        attachment.setContentType("text/plain");
        attachment.setText("Hello World");
        // add the new payload as attachment to the message
        msg.addAttachment(attachment);
        // provide the XI message for returning
        inputModuleData.setPrincipalData(msg);
    } catch (Exception e) {
        // raise exception, when an error occurred
        ModuleException me = new ModuleException(e);
        throw me;
    }
    // return XI message
    return inputModuleData;
}
```

3.5 Creating a Java Archive
Select the package and in the context menu, choose Build EJB Archive, as shown in the following figure:
3.6 Creating an Enterprise Application Archive (EAR)

In order to deploy your adapter module to the J2EE Adapter Framework, you have to create an EAR file:

- Create a new project as J2EE -> Enterprise Application Project.
- Select your EJB project(s) as reference.
- In this project, you need to assign references in application-j2ee-engine.xml.
- Open this file, choose the source tab page, and apply the following code:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application-j2ee-engine SYSTEM "application-j2ee-engine.dtd">
<application-j2ee-engine>
  <reference>
    <reference-type="hard">
      <reference-target
        provider-name="sap.com"
        target-type="library">com.sap.aii.af.lib</reference-target>
    </reference>
    <reference>
      <reference-type="hard">
        <reference-target
          provider-name="sap.com"
          target-type="service">com.sap.aii.adapter.xi.svc</reference-target>
      </reference>
      <reference>
        <reference-type="hard">
          <reference-target
            provider-name="sap.com"
            target-type="service">com.sap.aii.af.svc</reference-target>
        </reference>
        <reference>
          <reference-type="hard">
            <reference-target
              provider-name="sap.com"
              target-type="service">com.sap.aii.af.cpa.svc</reference-target>
          </reference>
        </reference>
        <fail-over-enable
          mode="disable"/>
    </reference>
  </application-j2ee-engine>
```

If you want to use additional libraries for your adapter module project, you may need to apply them here as well:
• Select the project and choose New -> META-INF/log-configuration.xml. This is particularly useful if you want to add trace information to your module. The following code is an example for the file log-configuration.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log-configuration SYSTEM "log-configuration.dtd">
<log-configuration>
  <log-formatters>
    <!-- This formatter produces human readable messages. -->
    <log-formatter
      name="trc"
      pattern="%26d %150l [%t] %10s: %m"
      type="TraceFormatter"/>
  </log-formatters>

  <log-destinations>
    <!-- Destination for Trace Information of this sample -->
    <log-destination
      count="5"
      effective-severity="DEBUG"
      limit="2000000"
      name="sample.trc"
      pattern="./log/applications/sample/default.trc"
      type="FileLog">
      <formatter-ref
        name="trc"/>
    </log-destination>
  </log-destinations>

  <log-controllers>
    <!-- Trace Location sample -->
    <log-controller
      effective-severity="DEBUG"
      name="sample">
      <associated-destinations>
        <destination-ref
          association-type="LOG"
          name="sample.trc"/>
      </associated-destinations>
    </log-controller>
    <!-- Logging Category: none, use the default XILOG -->
  </log-controllers>
</log-configuration>
```

• Select the project and in the context menu, choose Build Application Archive. Make sure that the libraries are not inside the generated EAR file, or delete them using WinZip.
• Deploy the EAR file from the deploy service in the Visual Administrator tool, or with the Software Deployment Manager (SDM).
3.7 Using the Module

- If you want to use the module in an adapter, open the Integration Directory and create a new communication channel.
- Enter the adapter-specific parameters and choose the Module tab page. A default entry contains the required adapter module.
- Apply your module before the specific-adapter module:
  - Enter `localebjs/<JNDI-Name>` as module name.
  - Choose the type `Locale Enterprise Bean`.
  - Choose any module key. Using your module key, you can assign parameters to your module in the Module Configuration table.
4 Appendix

4.1 Documentation for Module Development
- In the SAP Help Portal, choose the Documentation tab page.
- Choose the SAP NetWeaver tab page, then SAP NetWeaver '04 and English as the language.
- Navigate to SAP NetWeaver -> Process Integration -> SAP Exchange Infrastructure -> Runtime -> Connectivity -> Partner Connectivity Kit -> Adapter and Module Development.

4.2 Documentation of the Resource Adapter API
- When you download the latest patch for XI ADAPTER FRAMEWORK 3.0 from SAP Service Marketplace, you receive a description of the classes of the resource adapter API.
- Using WinZip, recursively unpack sample_ra.sda -> sample_ra.rar -> sample_ra.jar. Extract the latter to your local PC. You can now access the JavaDoc by opening the index.html file in a Web browser.

4.3 Sample Module
You receive sample code for an adapter module (which is already deployed and ready to use) from the same resource mentioned above.
- Recursively extract sample_module.sda -> sample_module.jar -> ConvertCRLFfromToLF.java

4.4 Code Samples

4.4.1 Reading File Names
In the file/FTP sender adapter, you can access the name of the processed file. Apply the following code to your module:

```java
import java.util.Hashtable;
...

public ModuleData process(ModuleContext moduleContext,
ModuleData inputModuleData)
throws ModuleException {

Message msg = (Message) inputModuleData.getPrincipalData();
Hashtable mp = (Hashtable) inputModuleData.getSupplementalData("module.parameters");
String fx = (String) mp.get("FileName");

...}
```
4.4.2 Writing Audit Log Entries

To write to the audit log of the message monitor of the Adapter Framework, you apply the following code:

```java
import com.sap.aui.af.service.auditlog.*;
...
public ModuleData process(ModuleContext moduleContext,
                          ModuleData inputModuleData)
    throws ModuleException {
    Message msg = (Message) inputModuleData.getPrincipalData();
    AuditMessageKey amk = new AuditMessageKey(msg.getMessageId(),
                                             AuditDirection.INBOUND);
    Audit.addAuditLogEntry(amk, AuditLogStatus.SUCCESS,
                           "CreateAttachment: Module called");
}
```

This results in an entry in the audit log:

<table>
<thead>
<tr>
<th>Time Stamp</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-07-06 15:32:08</td>
<td>Success</td>
<td>The message was successfully received by the messaging system. Profile: XI URL: <a href="http://p12989-Sep000000">http://p12989-Sep000000</a> companies/afwafw0/1</td>
</tr>
<tr>
<td>2005-07-06 15:22:08</td>
<td>Success</td>
<td>Using connection AFW. Trying to put the message into the receive queue.</td>
</tr>
<tr>
<td>2005-07-06 15:32:08</td>
<td>Success</td>
<td>Message successfully put into the queue.</td>
</tr>
<tr>
<td>2005-07-06 15:22:00</td>
<td>Success</td>
<td>The message was successfully retrieved from the receive queue.</td>
</tr>
<tr>
<td>2005-07-06 15:32:08</td>
<td>Success</td>
<td>The message status set to DLNO.</td>
</tr>
<tr>
<td>2005-07-06 15:22:00</td>
<td>Success</td>
<td>Delivering to channel MailReceiver</td>
</tr>
<tr>
<td>2005-07-06 15:32:08</td>
<td>Success</td>
<td>CreateAttachment: Module called</td>
</tr>
<tr>
<td>2005-07-06 15:22:00</td>
<td>Success</td>
<td>Mail message entering the Adapter</td>
</tr>
<tr>
<td>2005-07-06 15:32:08</td>
<td>Success</td>
<td>Mail Receiver adapter entered with QOS ExactlyOnce</td>
</tr>
<tr>
<td>2005-07-06 15:22:00</td>
<td>Success</td>
<td>Mail calling the adapter for processing</td>
</tr>
</tbody>
</table>

Total: 16 Entries 10 messages displayed per page; this is page 1 of 2 page(s)
www.sdn.sap.com/irj/SDN/howtoguides