SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies.

Data syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.
1 Scenario

You want to build a Java application that performs analytics on data in an XMLA provider and displays this data in a J2EE application deployed to the Web Application Server. You use the BI Java SDK to custom build this application and you establish the connection with the BI XMLA Connector. All you need for this scenario is included in SAP NetWeaver '04.

2 Introduction

This document provides detailed instructions on how to use the BI Java SDK and its BI Java Connectors in a J2EE application. It contains step-by-step instructions for creating a servlet that uses the BI XML A Connector to connect to a BW system.

In this document, we focus on the procedure for establishing a connection in the J2EE environment. After you establish this connection, refer to the tutorials and documentation for the BI Java SDK to expand upon your scenarios.

Although the screenshots in this document depict the process using the SAP NetWeaver Developer Studio, note that you can use the Java development tool of your choice to create the sample servlet.

3 The Step By Step Solution

In the first section below you prepare your system, and in the second section you create your servlet.

3.1 Prepare the System

To prepare the system, you first look up the JNDI name of your connector in the J2EE Visual Administrator, and then configure the connection properties.

In this example, we use the BI XML A Connector to connect to an SAP BW system and retrieve a list of schemas.
Use the J2EE Visual Administrator to determine the JNDI name of the BI XMLA Connector:

1. Select `<server node> → Services → Connector Container`.

2. On the Runtime tab, in the Connector Container section, locate the entry for the BI XMLA Connector, sap.com/com.sap.ip.bi.sdk.dac.connector.xmla.BI_SDK_XMLA, and double-click it to open the connector definition.

3. Select Managed Connection Factory → Connection Definition and locate the JNDI name in the Connector JNDI name field.

   For the BI XMLA Connector, this is `SDK_XMLA`.

Configure the connection properties using the J2EE Visual Administrator:

4. Still in on the Managed Connection Factory tab, select the Properties tab and set the required connection properties as shown to the right.

   Refer to the documentation for the BI XMLA Connector in its howto.html file or in the BW Installation Guide for information on the properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource</td>
<td>default</td>
</tr>
<tr>
<td>Statefulness</td>
<td>false</td>
</tr>
<tr>
<td>Language</td>
<td>EN</td>
</tr>
<tr>
<td>Password</td>
<td>(password)</td>
</tr>
<tr>
<td>User Name</td>
<td>(username)</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://server:port/sap/bw/xml/soap/xmla">http://server:port/sap/bw/xml/soap/xmla</a></td>
</tr>
</tbody>
</table>

5. Save the settings.
3.2 Create the Sample Servlet

6. In the SAP NetWeaver Developer Studio, use the New Project wizard to create a Web Module Project.

7. In the J2EE Explorer, right-click on the new Web Module project, and select New → Servlet from the context menu to create a new servlet component.
Edit the servlet’s web.xml file to add a resource reference to the BI XMLA Connector:

8. In the J2EE Explorer, double-click the web.xml file listed under your Web Module project, and select the Resource tab.
9. In the Resource Reference Name field, enter the JNDI name for your connector (see Step 3, above).
10. From the Resource Type drop-down list, select javax.resource.cci.ConnectionFactory.

11. In the servlet’s source code, look up the XMLA Connection Factory by its JNDI name, and establish the connection using the default connection properties. Use the code below to get connection and retrieve schemas.

   Note: The BI Java SDK libraries are required to compile this scenario, and can be found in <your NetWeaver Developer Studio installation folder>eclipse\tools\bi_sdk\bi_sdk.zip, or can be downloaded from the SAP Developer Network at http://www.sdn.sap.com/.

   ```java
   IConnectionFactory connectionFactory = null;
   IConnectionSpec connectionSpec = null;
   IBIConnection connection = null;
   Context initctx = new InitialContext();
   // perform JNDI lookup to obtain connection factory
   connectionFactory = (IConnectionFactory) initctx.lookup("java:comp/env/SDK_XMLA");
   // get connection using default connection properties
   connection = (IBIConnection) connectionFactory.getConnectionEx(null);
   IBIOlap olap = ((IBIConnection) connection).getOlap();
   List schemas = olap.getSchema();
   connection.close();
   ```
12. In the *New Project* wizard, create a new J2EE Enterprise Application Project.

![New Project wizard](image)

13. In the *J2EE Explorer*, right-click on the Enterprise Application Project, and select *Add Modules* to add your servlet to this Web application.

![J2EE Explorer](image)
Configure the required library references for your Web application:

14. In the J2EE Explorer, double-click on the application-j2ee-engine.xml file listed under your Web application.

15. On the General tab, add the library references as displayed to the right.

16. Deploy and test your servlet. When you run this servlet, you should retrieve a list of schemas in your system.
4 Appendix: Source Code

Below is source code for example servlet, web.xml, and application-j2ee-engine.xml files which demonstrate this scenario.

4.1 Exercise_4.java servlet

```java
package teched.bi_sdk.exercises;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.omg.cwm.analysis.olap.Schema;
import com.sap.ip.bi.sdk.dac.connector.IBIConnection;
import com.sap.ip.bi.sdk.dac.connector.IBIOlap;
import com.sapportals.connector.connection.IConnectionFactory;
import com.sapportals.connector.connection.IConnectionSpec;

public class Exercise_4 extends HttpServlet {
    IConnectionFactory connectionFactory = null;
    IConnectionSpec connectionSpec = null;
    IBIConnection connection = null;

    public void init() throws ServletException {
    }

    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML>");
        out.println("<HEAD><TITLE>Sample Servlet</TITLE></HEAD>");
        out.println("<BODY>");
        try {
            Context initctx = new InitialContext();
            // perform JNDI lookup to obtain connection factory
            connectionFactory = (ICredentialsFactory) initctx.lookup("java:comp/env/");
            // get connection using default connection properties
            connection = (IBIConnection) connectionFactory.getConnectionEx(null);
```

IBIOlap olap = ((IBIConnection) connection).getOlap();

List schemas = olap.getSchema();
Schema schema = null;

for (int i = 0; i < schemas.size(); i++) {
    schema = (Schema) schemas.get(i);
    out.println("schema: " + schema.getName() + "<br>");
}

} catch (Exception e) {
    e.printStackTrace(out);
} finally {
    try {
        connection.close();
        out.println("</BODY></HTML>");
    } catch (Exception e) {
        e.printStackTrace(out);
    }
}

} catch (Exception e) {
    e.printStackTrace(out);
} finally {
    try {
        connection.close();
    } catch (Exception e) {
        e.printStackTrace(out);
    }
}

out.println("</BODY></HTML>");
out.close();

} public void destroy() {
}

} // class

4.2  web.xml file

<!--xml version="1.0" encoding="UTF-8"?-->
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <display-name>WEB APP</display-name>
    <description>WEB APP description</description>
    <servlet>
        <servlet-name>Exercise_4</servlet-name>
        <servlet-class>teched.bi_sdk.exercises.Exercise_4</servlet-class>
    </servlet>
    <resource-ref>
        <res-ref-name>SDK_XMLA</res-ref-name>
        <res-type>javax.resource.cci.ConnectionFactory</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>
</web-app>

4.3  application-j2ee-engine.xml file

<!--xml version="1.0" encoding="UTF-8"?-->
<!DOCTYPE application-j2ee-engine SYSTEM "application-j2ee-engine.dtd">
<application-j2ee-engine>