Integrating Mobile Device With SAP XI

Applies To:

SAP NW04 – SAP XI 3.0 (SP14)

Summary

This document suggests a step-by-step approach to Integrating Mobile Device with SAP XI. For developing Mobile Interface, J2ME (Java 2 Platform, Micro Edition) Is used.

By: Tuhin Oza

Company: Infosys Technologies Limited

Date: 09 February 2006

Table of Contents

Introduction

In this document I will explain a step-by-step approach to integrating Mobile Device with SAP XI.

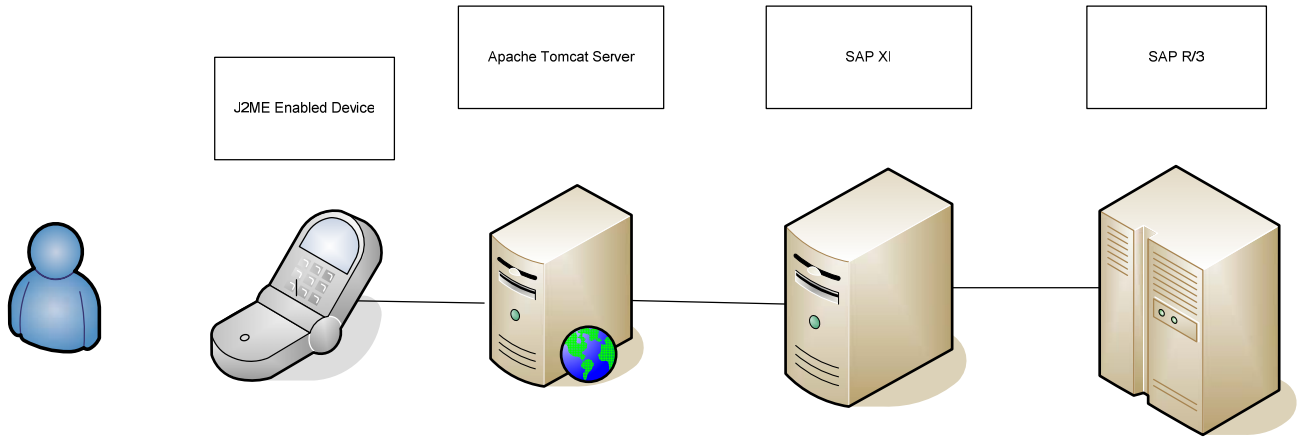
SAP XI is emerging as one of the preferred middleware in a landscape where SAP R/3 or any other SAP product already exists.

Mobile devices have limited memory so to develop an application for Mobile devices J2ME (Java 2 Platform, Micro Edition) was used.

Business Scenario

A user wants to check availability of material (Quantity of Material) using his Mobile device(Java Enabled). The following solution can be used.

Data Flow



User will enter Plant, Material Number and Quantity on mobile device.

The captured input data will be passed to Servlet as input parameter as http request.

The servlet residing in Web server will convert Input parameter into XML message and send the Message to XI.

RFC will get executed on R/3, which will check whether for given Material Number in given Plant the required quantity is available or not.

The response from R/3 will be routed back to XI and from XI to Servlet.

The Midlet (Java Application made specifically for Mobile Device) will receive response from servlet which will be displayed to user.

Steps to Implement Scenario

Develop Midlet to capture Input.

A MIDlet is a Java program for embedded devices, more specifically the J2ME virtual machine. Generally, these are games and applications that run on a cell phone.

MIDlets will run on any device that implements J2ME Mobile Information Device Profile. Like all Java programs, MIDlets are "compile once, run anywhere". To write a MIDlet, you can get Sun's Wireless Toolkit from the Java website, which is available on several platforms and is completely free.

A MIDlet has to fulfill the following requirements in order to run on a mobile phone:

The main class needs to be a subclass of javax.microedition.midlet.MIDlet

The MIDlet needs to be packed inside a .jar file (e.g. by using the jar-tool)

The .jar file needs to be pre-verified by using a preverifier.

The attached code captures(Material No, Plant and Quantity) from user and passes the same to Servlet as arguments. The response coming back from servlet can be seen by User.

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;   // Import required libraries

public class MatMidlet extends MIDlet implements CommandListener,Runnable {

private Command cmExit;
private Command cmCheck;
private Display display;
TextField textMat = null;
TextField textPlant = null;
TextField textQuan = null;
TextField res = null;

 Form newForm =null;

    String url = null;

    public MatMidlet() {

// Constructor to initialize first form for display and crate 3 input boxes and  2 command buttons
```

```
        display = Display.getDisplay(this);
         Form form = new Form("Form");

         textMat = new TextField("Material Number","P-502", 20, 0);

         textPlant = new TextField("Plant","3200", 20, 0);

         textQuan = new TextField("Quantity","50", 20, 0);

         cmCheck = new Command("Check", Command.SCREEN,1);

        cmExit = new Command("Exit", Command.EXIT, 1);

         form.append(textMat);

         form.append(textPlant);

         form.append(textQuan);

         form.addCommand(cmCheck);

         form.addCommand(cmExit);

         form.setCommandListener(this);
//Add command buttons and input boxes on current form
       display.setCurrent(form);


    }


    /**

     * Initialization. Invoked when we activate the MIDlet.

     */

    public void startApp() {

    }
public void commandAction(Command c, Displayable s){
//On exit gracefully terminate the application
    if (c == cmExit){
      destroyApp(false);
     notifyDestroyed();
    }
```

```
    else if( c == cmCheck){
                Thread t = new Thread(this);
                t.start();
      }

  }

    public void run(){

    //URL for http connection with input parameters as arguments

                try {

   url =
"http://172.21.86.145:8080/examples/servlet/MatServlet?Mat="+textMat.getString()+"&Plant="+te
xtPlant.getString()+"&Quan="+textQuan.getString();

                                                                invokeServlet(url);

                                            } catch (IOException e) {

                                               System.out.println("IOException " + e);

                                               e.printStackTrace();
                                    }

      }

    /**

     * Pause, discontinue ....

     */

    public void pauseApp() {

    }

    /**

     * Destroy must cleanup everything.

     */

    public void destroyApp(boolean unconditional) {

    }

    /**

     * Prepare connection and streams then invoke servlet.
```

```
*/
 void invokeServlet(String url) throws IOException {
    HttpConnection c = null;
    InputStream is = null;
    OutputStream os = null;
    int ch;
    StringBuffer b = new StringBuffer();
  //Open http connection and  pass data to Servlet.
    try {
        c = (HttpConnection)Connector.open(url);
        c.setRequestMethod(HttpConnection.GET);
        c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
        c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
        c.setRequestProperty("Content-Language", "en-CA");
        is = c.openDataInputStream();
        cmExit = new Command("Exit", Command.EXIT, 1);
        newForm = new Form("Form");
        newForm.addCommand(cmExit);
        newForm.setCommandListener(this);
       // Use input stream to read response getting back from servlet.
  //Create from to display response and display response on to screen.
     while ((ch = is.read()) != -1) {
                        b.append((char) ch);
                                                }
  //Create texfield to display response and append texfield to form.
                        res = new TextField("Response",b.toString(),1024,0);
                        newForm.append(res);
```

```
        } finally {
          if(is!= null) {
            is.close();
          }
          if(c != null) {
            c.close();
          }
        }
        display.setCurrent(newForm);
      }
    }
```

Steps to create Midlet

- Use any text editor to write the code.
- Crate a new project in wireless toolkit for example “MatMidlet”.
- Store .java file into \src folder of “MatMidlet” Project.
- Build the project. Run and Launch the application.
- The screenshot shows the running application on J2ME Emulator.

Input parameters (Material No,Plant,Quantity) as Request:Midlet->Servlet-> SAP XI

Response SAP XI->Servlet->Midlet

Development of Servlet To process input parameters.

A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

We will be using Apache tomcat server to run the servlet.

The attached code of Servlet passes the input parameters to method *checkMaterial(material no,plant,quantity)* and passes response back to Midlet.

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*; // Import required libraries

public class MatServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException,
ServletException
{
      response.setContentType("text/plain");
      PrintWriter out = response.getWriter();
      CheckMaterial cm =  new CheckMaterial();

//Capture the input parameter from request and pass the data to checkMaterial method,
send the response back to midlet.

       try{
out.println(cm.checkMaterial(request.getParameter("Mat"),request.getParameter("Plant"),
request.getParameter("Quan")));

          }

                              catch(Exception e){
                              System.out.println(e);
                              }
}

}
```

Apache Http client is used for sending request to SAP XI.

```
import java.io.*;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.*;
import org.apache.commons.httpclient.methods.PostMethod;

public class CheckMaterial {

public String checkMaterial(String mat,String plant,String quan)throws Exception

{

// URL with all details about interface for XI

        String strURL = "http://<XI server host name>:<http
port>/sap/xi/adapter_plain?namespace=http%3A//training.com/XI&interface=MI_OUT_Material&service=Mate
rial_Check_Service&party=&agency=&scheme=&QOS=BE&sap-client=100&sap-language=EN&sap-
user=*****&sap-password=*****";

        StringWriter sw = new StringWriter();
        PostMethod post = new PostMethod(strURL);
        sw.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
        sw.write("<ns0:Material_MT xmlns:ns0=\"http://training.com/XI\">");
        sw.write("<Material>");
        sw.write("<Material_No>"+mat+"</Material_No>");
        sw.write ("<Plant>"+plant+"</Plant>");
        sw.write ("<Quantity>"+quan+"</Quantity>");
        sw.write ("</Material>");
        sw.write ("</ns0:Material_MT>");

// Soap message with material no,plant and quantity as parameter

        post.setRequestEntity(new StringRequestEntity(sw.toString()));

        post.setRequestHeader("Content-type", "text/xml; charset=ISO-8859-1");

        //Create Instance Of  HTTP client

        HttpClient httpclient = new HttpClient();

try {

            int result = httpclient.executeMethod(post);   //Execute request

            DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();

            //Create new XML document out of response coming from XI which will come as String


            Document doc = db.parse(new InputSource(new StringReader(post.getResponseBodyAsString())));
```

```
        // Get the Response node
        NodeList nl = doc.getElementsByTagName("Response");
        Element elt = (Element) nl.item(no);
        // Passing the value for Response node back to Servlet.
        return(((Text)elt.getFirstChild()).getData().trim());
      } finally {
        // Release current connection to the connection pool once you are done
        post.releaseConnection();
      }
    }
}
```

Steps To Implement XI Scenario.

Integration Repository

Step1

Create Data Type with following structure for Request and Response.

Structure	Category	Type	Occurrence
Material_DT	Complex Type		
Material	Element		0..1
Material_No	Element	String	0..1
Plant	Element	String	0..1
Quantity	Element	Integer	0..1

Structure	Category	Type	Occurrence
Material_Res_DT	Complex Type		
Material	Element		0..1
Response	Element	String	0..1

Step2

Under Message Types right click and create ‘Material_MT’ and select the data type as ‘Material_DT’ and ‘MaterialRes_MT’ and select the data type as ‘Material_Res_DT. Save it.

Step3

Create “MI_OUT_Material” Message Interface with Category as outbound and mode as Synchronous. Refer ‘Material_MT’ as request message type and ‘MaterialRes_MT’ as response message type.

Step4

ZMATERIAL_AVAILABLITY_CHECK is remote enabled function module with Material_no,Plant and Quantity as input parameters.

The function module queries table ‘MARC’ and checks quantity of material for Given Plant and Material combination.

It returns response in form of String stating whether required quantity of material is available within the plant entered by user or quantity is not available within plant.

Import RFC ‘ZMATERIAL_AVAILABLITY_CHECK’ from R/3 into XI repository.

Step5

Create Request Message Mapping “MM_Material_Query”.

Source Message Type Element	Target Message Type Element	Functions to be used
Material_MT	ZMATERIAL_AVAILABLITY_CHECK	
Material		
Material_No	MATERIAL_NO	
Plant	PLANT	
Quantity	QUANTITY	

dCreate Response Message Mapping “MM_Material_Query”.

Source Message Type Element	Target Message Type Element	Functions to be used
ZMATERIAL_AVAILABLITY_CHECK.Response	MaterialRes_MT	
RESPONSE	Material	
	Response	

Step6

Create Interface Mapping “IM_Material_Query” .Select “MI_OUT_Material” as source and ZMATERIAL_AVAILABLITY_CHECK as target interface. Select ‘MM_Material_Query’ as source mapping program and ‘MM_Material_Query’ as target mapping program.

Integration Directory.

In the Integration Directory create a scenario “Material_Check”.

Step1

Select Service Without Party-> Business Service. Right click create business services 'Material_Check_Service'.

Configure Material_Check_Service as sender service and add "MI_OUT_Material" as sender interface.

No need to configure sender communication channel for HTTP adapter.

P.S: This is because HTTP adapter is not part of SAP J2EE engine. It reside directly on SAP ABAP engine and part of ICF service. So no need to configure sender communication channel for HTTP adapter. Also we don't have to define any sender agreement for HTTP adapter.

Step2

Assign R/3 business system to scenario and configure the RFC receiver communication channel.

Step3

Create **Receiver Agreement** With Following Detail.

Sender

Service: Material_Check_Service

Receiver

System: SAP R/3(R/3 Application server)

Interface: ZMATERIAL_AVAILABLITY_CHECK

Namespace: urn:sap-com:document:sap:rfc:functions

Receiver Communication Channel: RFC_Receiver

Step4

Create **Interface Determination** and use the below objects;

Sender

Service: Material_Check_Service

Interface: MI_OUT_Material

Namespace: http://training.com/xi/

Receiver

Service: SAP R/3(R/3 Application server)

And select Inbound Interface ZMATERIAL_AVAILABLITY_CHECK and Interface Mapping as IM_Material_Query.

Step5 Create **Receiver Determination** and use the below objects;

Sender

Service: Material_Check_Service

Interface: MI_OUT_Material

Namespace: http://training.com/xi/

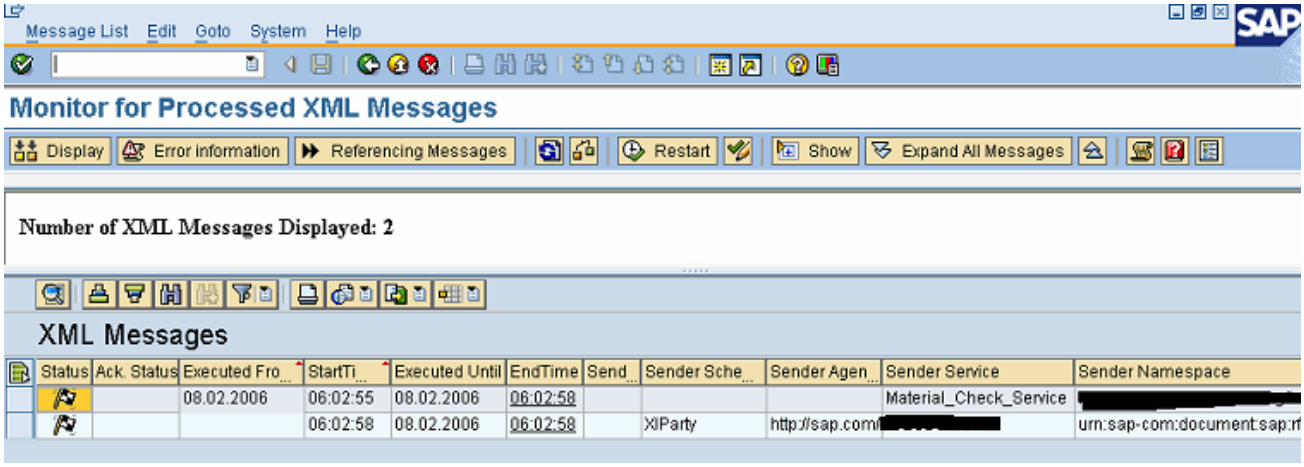
Receiver

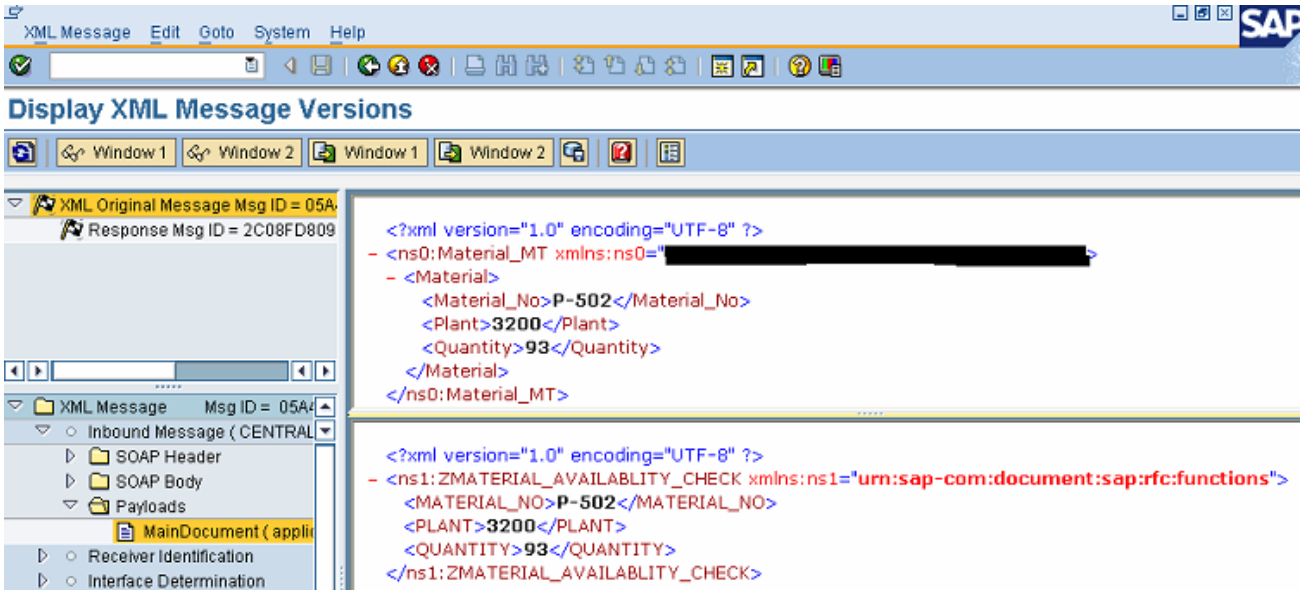
Party: *

Service: *

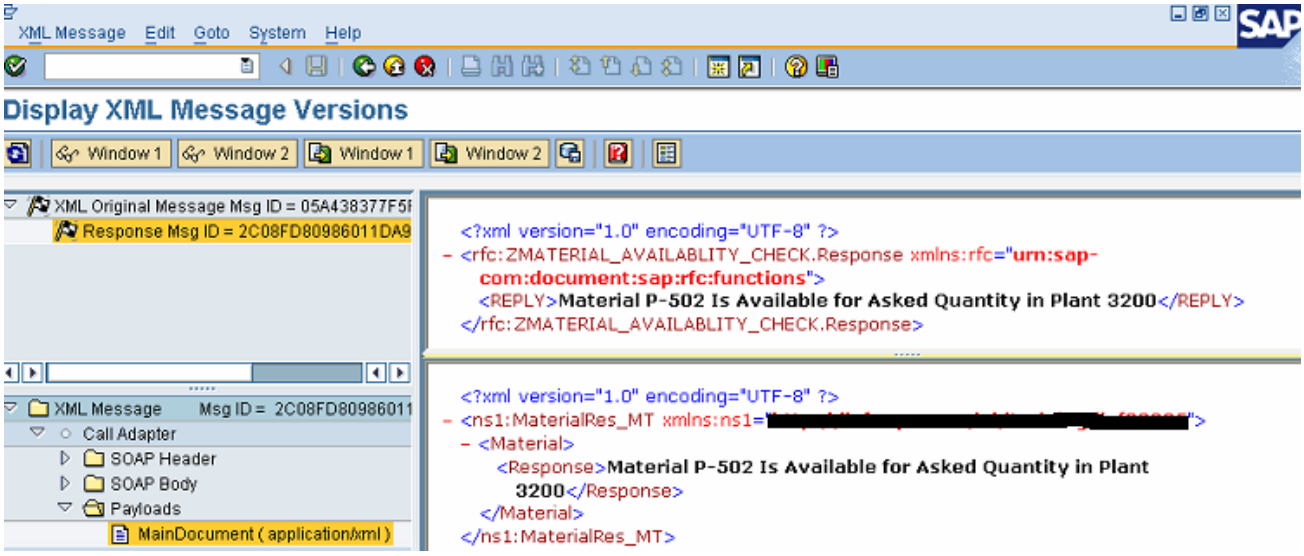
And in Configured Receivers select service as SAP R/3. Save it.

After building the scenario send the request through mobile device. As shown in screen shot data will pass through SAP XI.

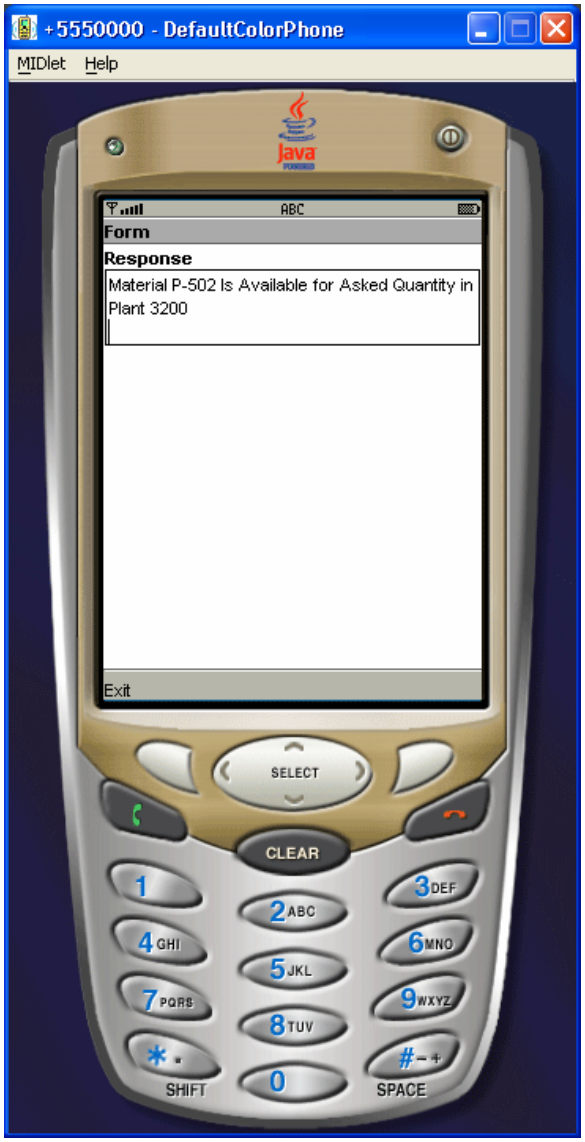




Source and Target Request Messages after Request Message Mapping.



Source and Target Response Messages after Response Message Mapping.



The response will be displayed back to user in terms of message stating availability of material.

Author Bio

Tuhin is working as XI development consultant with Infosys Technologies Limited, India. His areas of expertise include ABAP, BSP, Java, J2ME and SAP XI.

Disclaimer & Liability Notice