Integrating SAP Manufacturing and Integration (MII) with SAP ERP and SAP NetWeaver PI

Applies to:

SAP MII Version 12.0.3 Build (106).

Summary

Using SAP Manufacturing Integration and Intelligence (MII) at the plant site is essential towards reducing the complexity of integrating a plant site with an ERP system. This article aims at providing a combined solution, in which SAP MII handles the integration with SAP ERP system and SAP NetWeaver PI (Process Integration).

The goal of this article is to guide you through the steps, code samples of how to integrate SAP MII with SAP ERP and SAP NetWeaver PI, involving user interaction availed through MII visualization services.

This post can also be quite useful for beginners who want to learn MII on their own.

Author: Anumeha Khandelwal

Company: SAP GD

Created on: 03 April 2008

Author Bio

Anumeha Khandelwal is currently working as Associate Consultant with SAP GD, India.

Table of Contents

Scenario

Integrating SAP MII with SAP ERP, fetching the currency name and its description and displaying the result. The user is given an option to make a selection on the currency displayed, which is captured and send further to SAP Net weaver PI.

Introduction

This paper will show the process of integrating SAP MII with SAP ERP and SAP NetWeaver PI (PI). This document outlines the basic usage of the following SAP MII actions

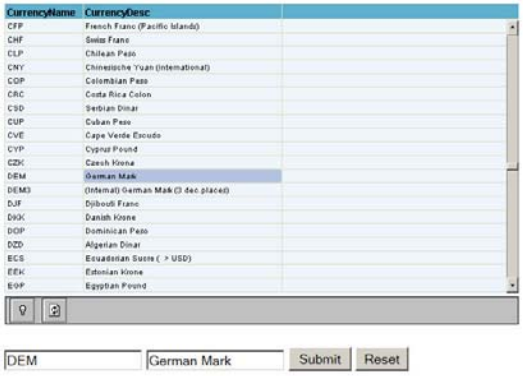
- SAP JCO Interface : This action is used to create a connection to the SAP Java Connector (JCo) interface from the ERP server and create a Remote Function Call (RFC) request.
- HTTP XI Action: This action uses the HTTP Post action to send messages to SAP NetWeaver Process Integration. Executing the action builds the URL and posts the document to the URL using HTTP.

Step 1: Two different transactions are created, one to retrieve data from SAP ERP and the other to send the data to PI.

Step 2: In order to display the retrieved data to the end client XacuteQuery is developed to create templates that take the output of the transaction and send the results to an SAP MII applet. Similarly a xacute query is created for executing the transaction that will be sending data to PI.

Step 3: Further iGrid is used to provide a grid display template for viewing data output from the xacute query. You can define color context-sensitive highlighting to accentuate data elements based on data values or ranges. The grid provides user with a selection screen from where the required item can be selected. Since the xacute query, that will be executing the transaction to send data to PI, is not required to be displayed on the screen an iCommand is used.

Step 4: FrontPage is used to customize the page to capture double click event and to retrieve the selected value.



CurrencyName	CurrencyDesc
CFP	French Franc (Pacific Islands)
CHF	Swiss Franc
CLP	Chilean Peso
CNY	Chinesische Yuan (international)
COP	Colombian Peso
CRC	Costa Rica Colon
CSD	Serbian Dinar
CUP	Cuban Peso
CVE	Cape Verde Escudo
CYP	Cyprus Pound
CZK	Czech Krona
DEM	German Mark
DEM3	(Internal) German Mark (3 dec.places)
DJF	Djibouti Franc
DKK	Danish Krone
DOP	Dominican Peso
DZD	Algerian Dinar
ECS	Ecuadorian Sucre (> USD)
EEK	Estonian Krone
EGP	Egyptian Pound

DEM German Mark

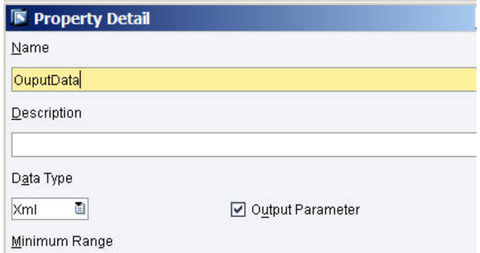
Fig 1: Screen displaying data fetched from SAP ERP and two text boxes that display the selected item and a submit button to send the selected data to PI and a reset button to clear the selection.

Creating Transaction to Retrieve Data from SAP ERP

Declaring Variables

Variable declaration can be done at any stage, but I'm doing it just for the sake of clarity. Go to Transaction->properties and create following output parameter of type XML.

- Output



Property Detail

Name: OuputData

Description:

Data Type: Xml

☑ Output Parameter

Minimum Range

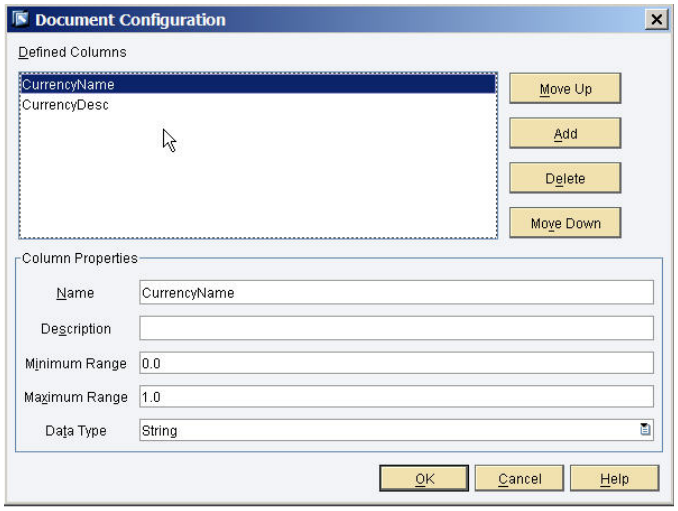
Fig 2: Do not forget to tick the Output Parameter

Adding Document Action

SAP MII Document is used to hold multiple XML rows during the course of transaction and it also converts the XML format to SAP MII understandable form.

The structure of SAP MII document should be same as the output required by you.

Here, I'm taking the *CurrencyName and CurrencyDesc* from the BAPI output.



Document Configuration

Defined Columns

CurrencyName
CurrencyDesc

Column Properties

Name: CurrencyName

Description:

Minimum Range: 0.0

Maximum Range: 1.0

Data Type: String

Fig 3: Document Configuration

Adding and Configuring JCO Action Block

JCO Action Blocks are used to call an RFC (Remote Function Call) from SAP server.

Insert SAP JCO interface action block (from SAP ERP interface tab) in second sequence and right click->configure.

In the configuration window, choose **SAP Server Name** and give search pattern and SAP RFC name as **BAPI_CURRENCY_GETLIST.** After that, click ok and hit choose generate sample XML.

Configuring Repeater

In the third sequence, add the repeater action block and configure it to item level of output from BAPI.

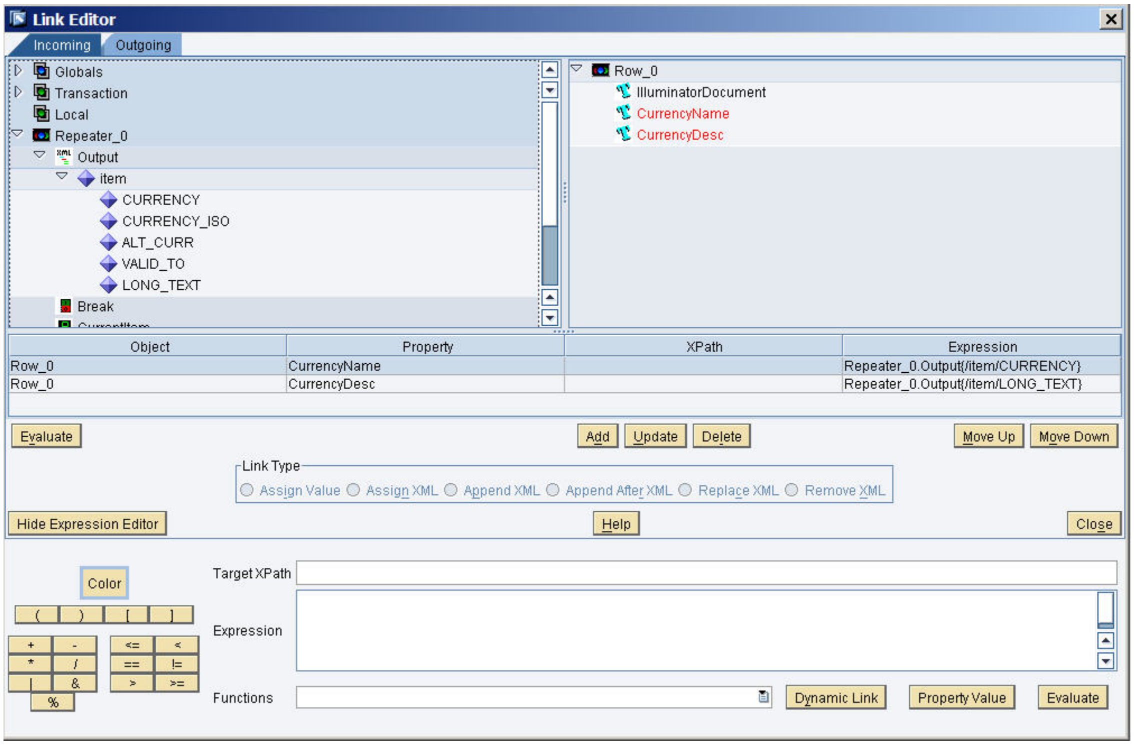
Populating Data in SAP MII Row Action

Since the data comprises of more than one record, so we have to use loop the RFC output line by line.

So for this we will be using **repeater** (from logic) and r**ow** (from SAP MII XML output)

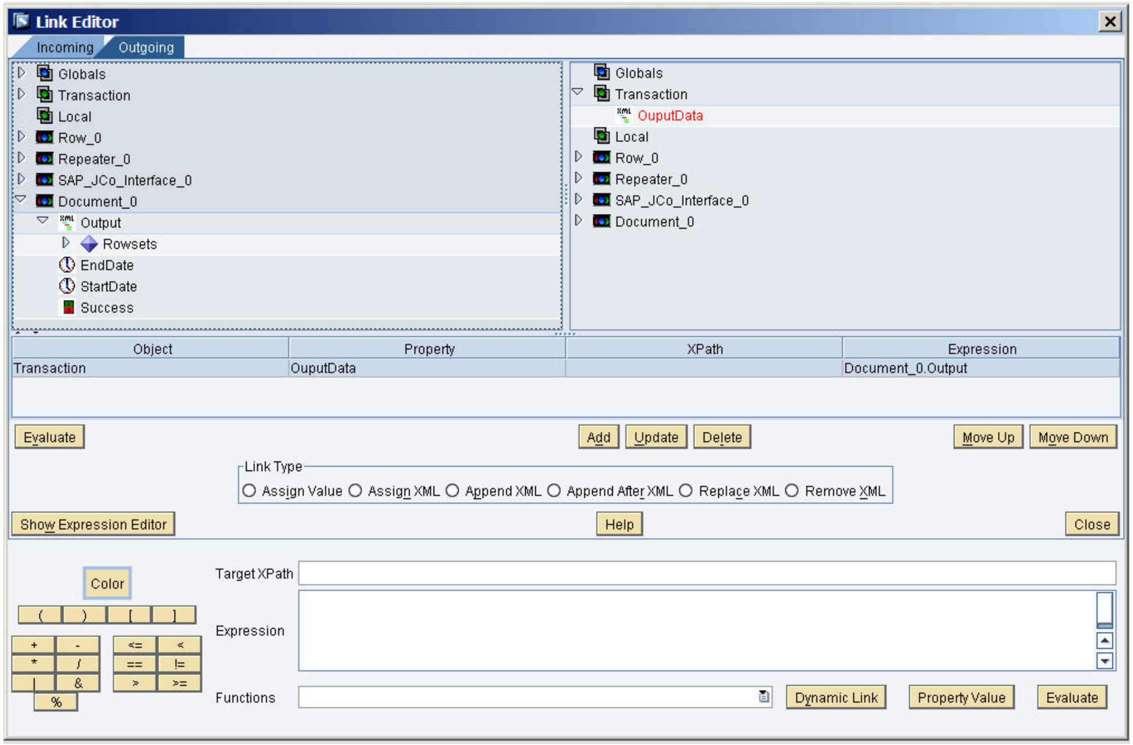
Linking of Fields

Link each field of the row to the corresponding field from the output of the repeater



Object	Property	XPath	Expression
Row_0	CurrencyName		Repeater_0.Output{/item/CURRENCY}
Row_0	CurrencyDesc		Repeater_0.Output{/item/LONG_TEXT}

Fig 4: Row, Link editor, Incoming



Object	Property	XPath	Expression
Transaction	OuputData		Document_0.Output

Fig 5: Row, Link editor, Outgoing

The Final Transaction will look like this

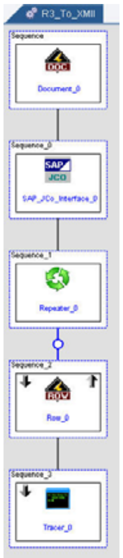


Fig 6: R3_to_xmii transaction

You can add tracer any where in the transaction for logging purpose.

Creating Transaction to Send Data to PI

Declaring Variables

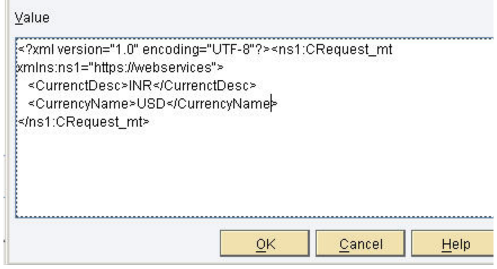
Variable declaration can be done at any stage. Go to Transaction->properties and create following input parameter of type String.

- CurrencyName
- CurrencyDesc

Go to Transaction>Local Properties and create following variable of type xml

- Http_Send

Set value to the xml structure that will be sent to PI.



Value

```
<?xml version="1.0" encoding="UTF-8"?><ns1:CRequest_mt
xmlns:ns1="https://webservices">
  <CurrenctDesc>INR</CurrenctDesc>
  <CurrencyName>USD</CurrencyName>
</ns1:CRequest_mt>
```

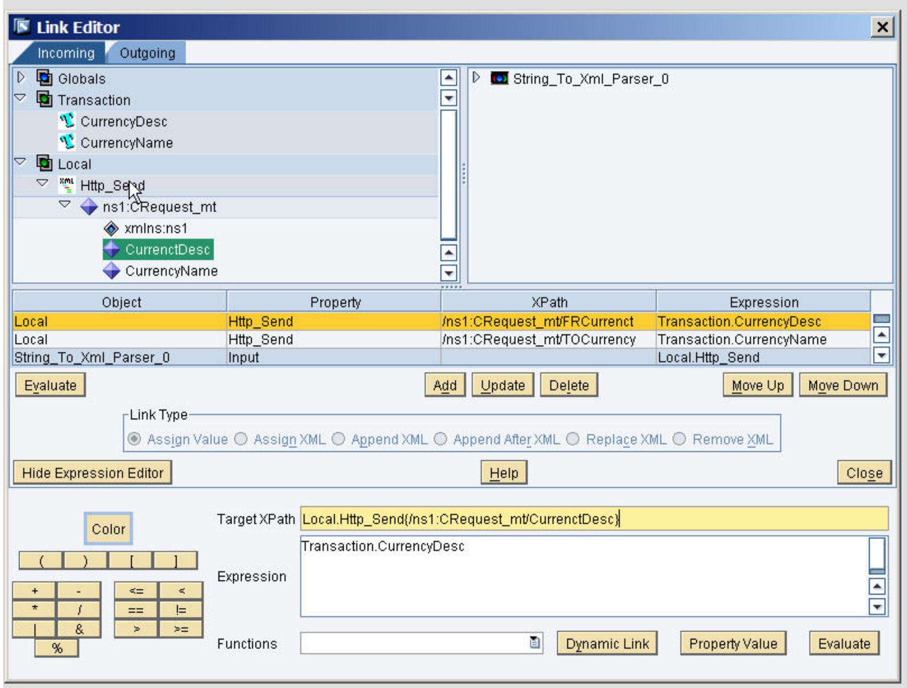
Fig 7: Local Properties

Adding and Configuring String to Xml Parser Action Block

This action is used to convert a string that contains XML content into an XML document

Insert this action in the first sequence and right click->Links

In the Link editor assign Transaction input variables to the Local variable xml tags respectively. Link the input field of String to Xml Parser to the Http_Send, Local variable.



Object	Property	XPath	Expression
Local	Http_Send	/ns1:CRequest_mt/FRCurrenct	Transaction.CurrencyDesc
Local	Http_Send	/ns1:CRequest_mt/TOCurrency	Transaction.CurrencyName
String_To_Xml_Parser_0	Input		Local.Http_Send

Link Type: ● Assign Value ○ Assign XML ○ Append XML ○ Append After XML ○ Replace XML ○ Remove XML

Target XPath: Local.Http_Send{/ns1:CRequest_mt/CurrenctDesc}

Expression: Transaction.CurrencyDesc

Functions:

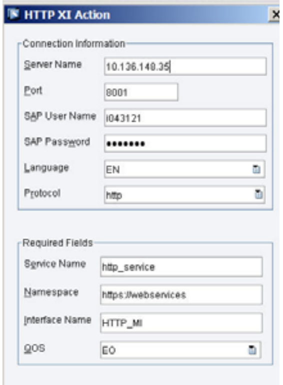
Fig 8: Link Editor, String To Xml Parser

Adding and Configuring HTTP_XI Action Block

This action uses the HTTP Post action to send messages to SAP NetWeaver Exchange Infrastructure (SAP NetWeaver XI). Executing the action builds the URL and posts the document to the URL using HTTP.

Insert the action block to the next sequence and right click>configure

In the configuration window give the PI server connection information



Connection Information

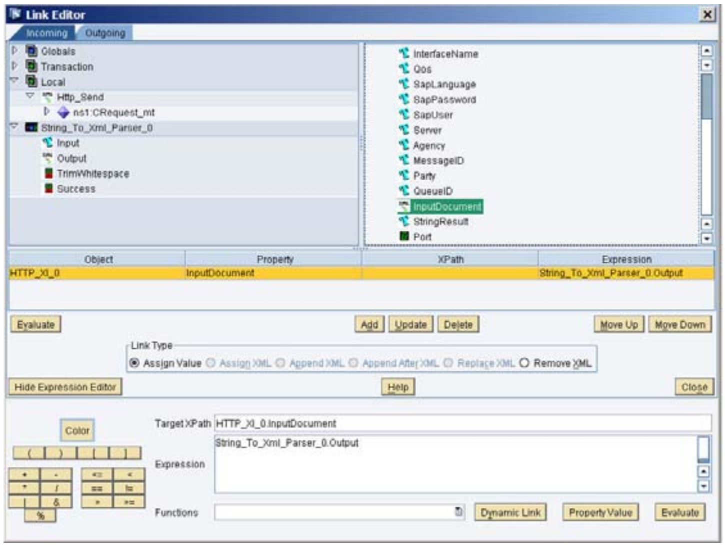
Server Name: 10.136.140.35
Port: 8001
SAP User Name: I043121
SAP Password: •••••••
Language: EN
Protocol: http

Required Fields

Service Name: http_service
Namespace: https://webservices
Interface Name: HTTP_MI
QOS: EO

Fig 9: HTTP_XI, Configuration Window

After this go to Link editor of HTTP_XI Action Block. Link the InputDocument field of HTTP_XI to output field of String To Xml Parser.



Object	Property	XPath	Expression
HTTP_XI_0	InputDocument		String_To_Xml_Parser_0.Output

Link Type: ● Assign Value ○ Assign XML ○ Append XML ○ Append After XML ○ Replace XML ○ Remove XML

Target XPath: HTTP_XI_0.InputDocument
Expression: String_To_Xml_Parser_0.Output
Functions:

Fig 10: Http_XI Link Editor

The transaction will look like this:

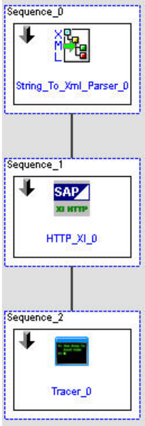


Fig 11: xmii_to_xi transaction

You can add tracer at any step for logging purpose.

Final transactions will look like this

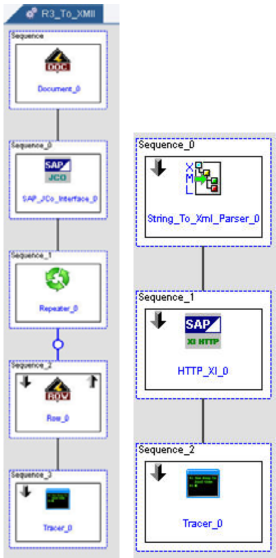
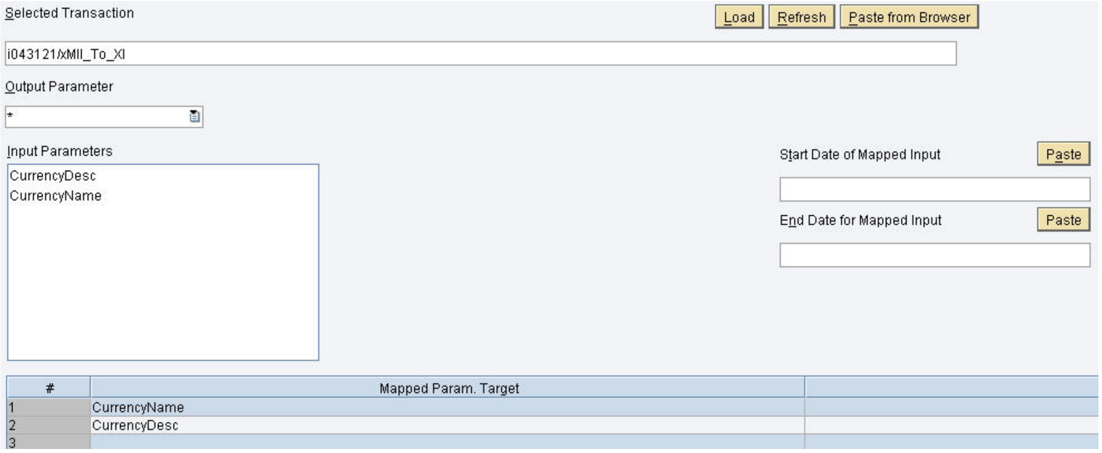


Fig 12:Final transactions

Creating Xacute Query

You use the *XacuteQuery* in the *SAP Manufacturing Integration and Intelligence (SAP MII) Workbench* to create templates that take the output of a transaction and send the results to an SAP MII applet.

Create xacute query for both the above transactions. In case to xmii_to_xi transaction, add the input parameters of the transaction to Mapped Param.Target. This will be used to set input parameter at runtime.



Selected Transaction: i043121/xMII_To_XI

Output Parameter: *

Input Parameters:
CurrencyDesc
CurrencyName

Start Date of Mapped Input:

End Date for Mapped Input:

#	Mapped Param. Target	
1	CurrencyName	
2	CurrencyDesc	
3		

Fig 13:Xacute_xmii_to_xi

Creating iCommand

The iCommand is more efficient than a grid since there are no display elements to process and the applet does not initially execute the template query request. Since I don't want the **xacute_xmii_to_xi** query data to be displayed on the grid, I create an iCommand **iCommand_xmii_to_xi** for the same.

Execute the iCommand right click on the applet displayed > view source, copy the code for applet created and save it.

Creating and Configuring iGrid

iGrid provides a grid display template for viewing data. You can use a grid to present a single or multiple selection list box to the user.

Create and iGrid **R3_xmii_xi**

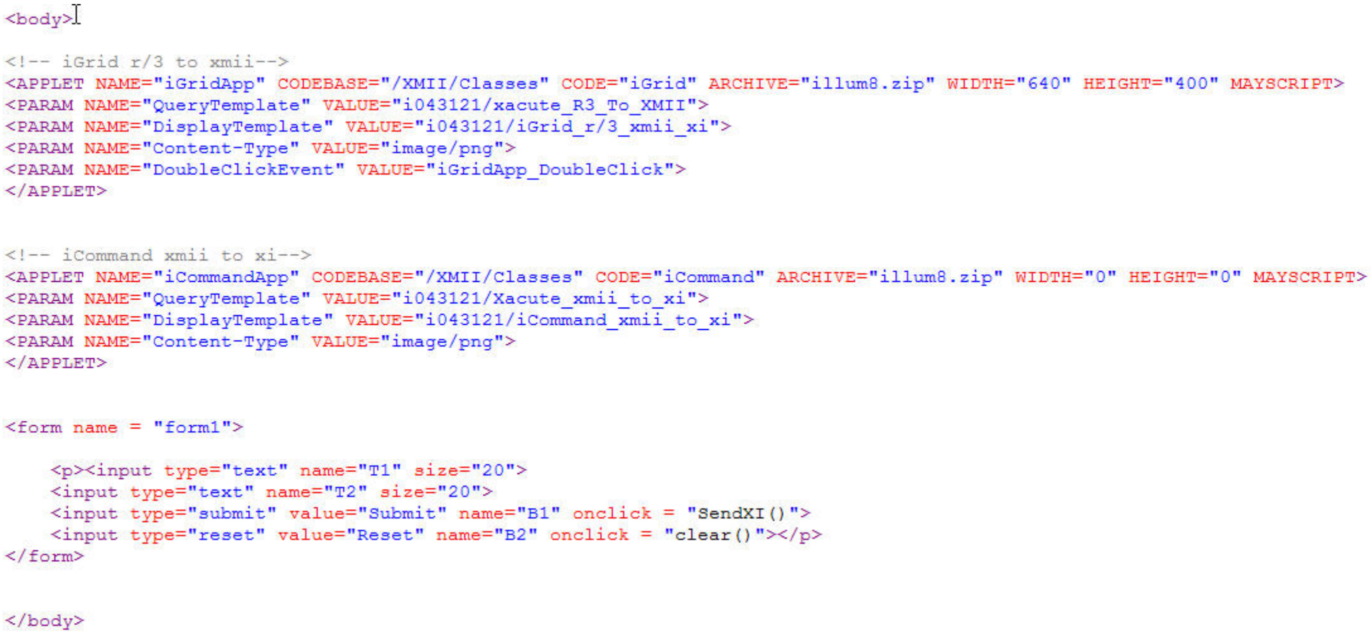
1. Under Layout option load the **xacute_R3_to_XMII** query
2. Add both the output parameters as column names.
3. Under UI Behavior check “Allow Multiple Selections” option.
4. Execute the iGrid, right click on the applet created> view source. Save the code generated for the applet.

Customizing the Page for User Input

FrontPage is used for customizing the page to be displayed.

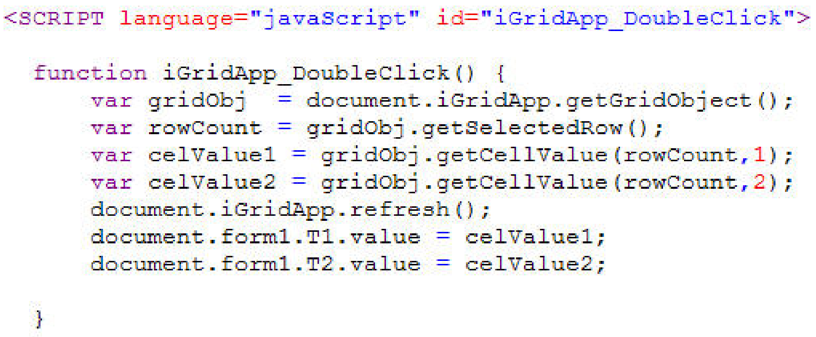
1. Open Ms FrontPage and create a new page.
2. Paste the code saved in the above steps within the <Body> tag of the page.
3. Insert a text box, a submit and reset buttons.

The code should look like this:



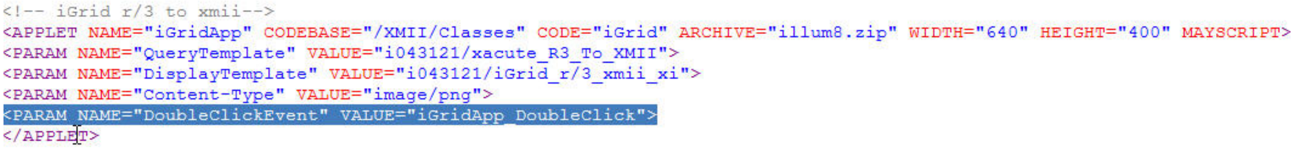
```
<body>

<!-- iGrid r/3 to xmii-->
<APPLET NAME="iGridApp" CODEBASE="/XMII/Classes" CODE="iGrid" ARCHIVE="illum8.zip" WIDTH="640" HEIGHT="400" MAYSCRIPT>
<PARAM NAME="QueryTemplate" VALUE="i043121/xacute_R3_To_XMII">
<PARAM NAME="DisplayTemplate" VALUE="i043121/iGrid_r/3_xmii_xi">
<PARAM NAME="Content-Type" VALUE="image/png">
<PARAM NAME="DoubleClickEvent" VALUE="iGridApp_DoubleClick">
</APPLET>


<!-- iCommand xmii to xi-->
<APPLET NAME="iCommandApp" CODEBASE="/XMII/Classes" CODE="iCommand" ARCHIVE="illum8.zip" WIDTH="0" HEIGHT="0" MAYSCRIPT>
<PARAM NAME="QueryTemplate" VALUE="i043121/Xacute_xmii_to_xi">
<PARAM NAME="DisplayTemplate" VALUE="i043121/iCommand_xmii_to_xi">
<PARAM NAME="Content-Type" VALUE="image/png">
</APPLET>


<form name = "form1">

    <p><input type="text" name="T1" size="20">
    <input type="text" name="T2" size="20">
    <input type="submit" value="Submit" name="B1" onclick = "SendXI()">
    <input type="reset" value="Reset" name="B2" onclick = "clear()"></p>
</form>


</body>
```

1. Use MII script assistant to insert code for capturing double click event on the applet and display the selection in the text box (which can be edited by the user)



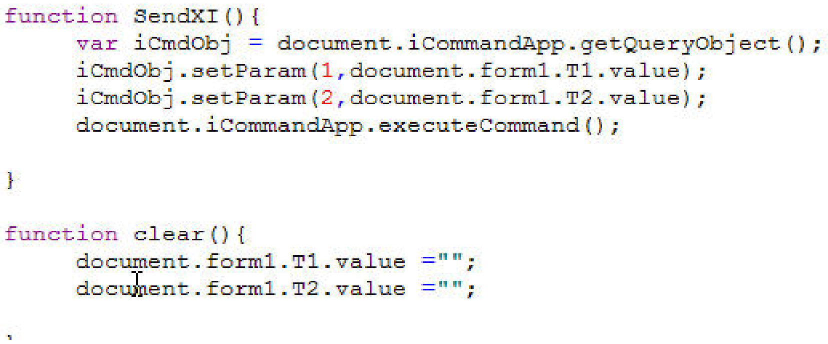
```
<SCRIPT language="javaScript" id="iGridApp_DoubleClick">

  function iGridApp_DoubleClick() {
      var gridObj  = document.iGridApp.getGridObject();
      var rowCount = gridObj.getSelectedRow();
      var celValue1 = gridObj.getCellValue(rowCount,1);
      var celValue2 = gridObj.getCellValue(rowCount,2);
      document.iGridApp.refresh();
      document.form1.T1.value = celValue1;
      document.form1.T2.value = celValue2;

  }
```

When you write the code for double click event on the iGrid applet make sure not to skip the following highlighted line of code



```
<!-- iGrid r/3 to xmii-->
<APPLET NAME="iGridApp" CODEBASE="/XMII/Classes" CODE="iGrid" ARCHIVE="illum8.zip" WIDTH="640" HEIGHT="400" MAYSCRIPT>
<PARAM NAME="QueryTemplate" VALUE="i043121/xacute_R3_To_XMII">
<PARAM NAME="DisplayTemplate" VALUE="i043121/iGrid_r/3_xmii_xi">
<PARAM NAME="Content-Type" VALUE="image/png">
<PARAM NAME="DoubleClickEvent" VALUE="iGridApp_DoubleClick">
</APPLET>
```

2. Write the code for executing the iCommand iCommand_xmii_to_xi which sends data to PI



```
function SendXI(){
        var iCmdObj = document.iCommandApp.getQueryObject();
        iCmdObj.setParam(1,document.form1.T1.value);
        iCmdObj.setParam(2,document.form1.T2.value);
        document.iCommandApp.executeCommand();

}

function clear(){
        document.form1.T1.value ="";
        document.form1.T2.value ="";
```

This function is executed when the user presses the "**Submit**" button. Function clear is called when the user clicks on "**Reset**" button.

3. Save the htm page as r3_xmii_xi.htm and publish on SAP MII server using Publish Assistant.

Running the HTM Page

You can execute and test your page from the MII menu page.

1. Change the URL from:

   http://<*server ip*>:<*Port*>/<*XMII*>/Menu.jsp

   to

   http://<*server ip*>:<*Port*>/<*XMII*>/ CM/<*page location*>/r3_xmii_xi.htm

2. a. Logon to Xmii Work bench

   b. Click on Web tab

   c. Goto ur folder where u have saved the htm page

   d. Right click on the page name> test.

Related Content

http://help.sap.com/saphelp_xmii120/helpdata/en/43/e80b59ad40719ae10000000a1553f6/frameset.htm

For more information, visit the Manufacturing homepage.

Copyright

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines/strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the 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.