Consuming SAP MII Business Logic Transaction as Web Service

Applies to:

SAP, SAP xMII Version 11.5 , Microsoft Visual Basic .Net 2005 / Express 2005
For more information, visit the Web Services homepage.

Summary

The SAP Manufacturing Intelligence and Integration (SAP MII) composite application helps manufactures to become more adaptive by connecting the SAP to the plant floor in real time and delivering actionable intelligence to production personnel. There may be situation that we need to tune our existing development environment and existing complex applications to communicate to SAP in real time. This paper explains how SAP MII business logic transaction could be exposed as a web service and how it can be accessed in our applications using Visual Basic .Net.

Author: Johnson George

Company: Greenheck Fan Corporation, USA

Created on: 12 August 2008

Author Bio

Johnson George is presently working with Greenheck Fan Corporation - a leading manufacturer of Air Movement and Control Equipments - and currently involved in developing composite applications using SAP MII. Johnson has around 11 years of experience in software application development using Delphi, Visual Studio, Java and PHP.

.

Table of Contents

Web Services: Introduction

Web services are one of the greatest technologies developed in the Internet world, which could be used to connect businesses with each other and clients in a standard way using Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI).

XML is used for structuring the data, SOAP is used to transfer the data, WSDL is used for describing the services and UDDI is used to get a list of services available. Web service allows application to communicate with each other without worrying about the underlying hardware, operating systems and programming languages.

Web service does not provide a user interface but exposes the business logic, which can be programmed for customization and hence the user is free to add his own interface to the application.

Overview

The SAP Manufacturing Integration and Intelligence (SAP MII), previously SAP xMII, offers a sophisticated yet easy-to-use software application for aggregating, transforming, and distributing plant and production information via XML and Web technologies. SAP MII is extensively used for developing composite applications for real-time plant floor visualization and for factory data collection

The web service interface of SAP MII's Business Logic Service transaction allows third party applications to call SAP MII Business Logic Service functionality using SOAP over HTTP and reads various inputs and delivers outputs from a transaction using the Web Service Definition Language (WSDL) that is automatically generated by SAP MII based on the transaction's configuration.

Since the WSDL is generated on-the-fly by SAP MII, you will have to predefine the output structure of the output XML generated by the business logic transaction prior to referencing this transaction as a web service. This is done by assigning the reference document to the transaction's output XML property.

Business Scenario

There may be many reasons that we may not be able to quickly upgrade or rewrite our existing application to SAP MII to interface with the SAP system . This could be due to the complexity or size of the system or due to the external dependencies of the system or even it could be due to the limited resources. In such a scenario, it is advisable that we can integrate the existing application to interface with SAP MII to communicate with SAP in real time. This document explains and explores the possibility of using SAP MII as a middle-ware or as a data broker for many of the existing yet complex plant floor applications by connecting these applications to SAP ERP system in real time.

In this paper, we demonstrate this by creating a new visual studio .Net project and adding a web service references to the SAP MII Business Logic Transaction from the .Net environment. In our example, we have a business logic transaction created in SAP MII which will get the variant configuration values from SAP using the standard remote enabled BAPI called CUXI_GET_SINGLE_CFG_BY_REMOTE. The name of the transaction is VcValues and it is saved under the folder PP in your SAP MII environment. This transaction accepts two parameters: GUID and MaterialNumber and renders the output through OutXml parameter.

Getting Started

Before we start, it is important that we should assign the reference document to our transaction's output property. The SAP MII generates the WSDL automatically and therefore we should predefine the output XML structure before adding a reference to the web service from the third party application .This is also useful , if the transaction is called from another transaction so that the structure of the output document of the called transaction will be visible to the calling transaction and the assignments can be made easily from the calling transaction.

Assigning Reference Documents

Step 1:

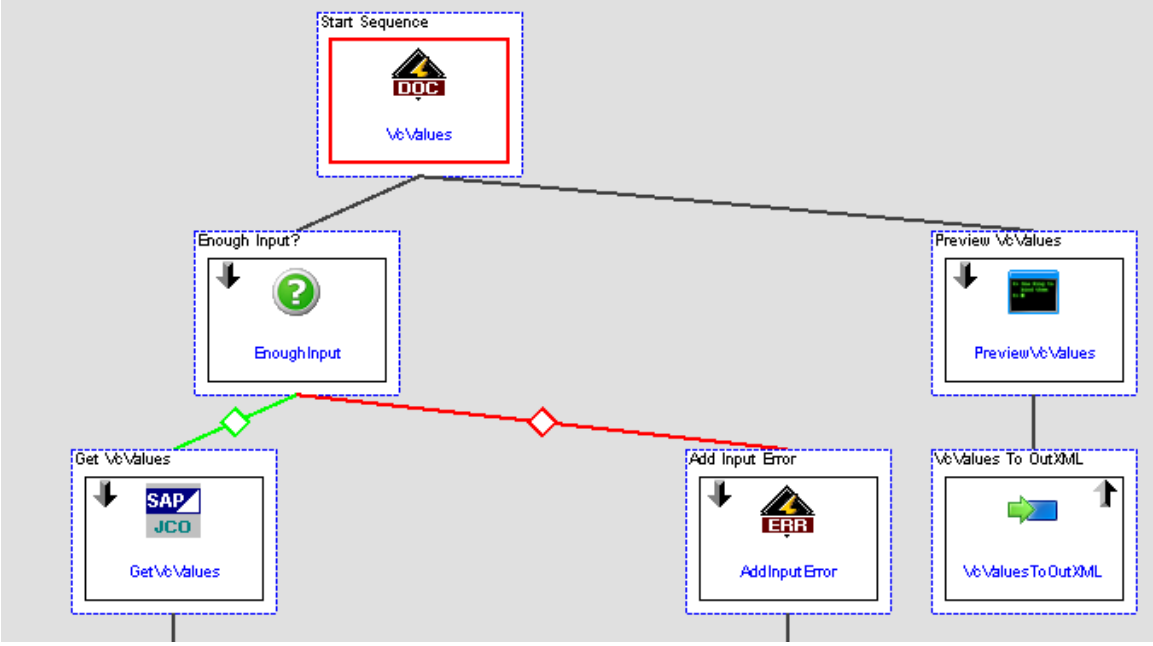
Locate the output parameter from your Transaction. Click the 'Transaction -> Properties' menu. In our example, it is OutXML which is of type xml.

Step 2:

Locate the SAP MII XML document object that is mapped to the output parameter of the transaction. For example: VcValues in the image below.

It is important to note is that the XML result generated from SAP MII is organized in a tree hierarchy known as the 'Document Object Model' or ***DOM***. This means that from a single starting point there are branches or ***nodes*** in the tree and each node can have its own child nodes and/or attributes. The SAP MII data XML file has three tree levels. A well formed XML file can have only one top level node. The SAP MII top level node is ***Rowsets***. Rowsets contains attribute values for the date the document was created, the software version etc.. Rowsets has one immediate child node named ***Rowset***. Rowset has two immediate children, ***Columns*** and ***Rows***. The Columns and Rows nodes are similar to describing the data in a spreadsheet, or grid.

The Columns node describes the data which will appear in each row. There are a series of attributes for each column of data, describing the column name, source column (data source column name), description, data type, and permissible minimum and maximum ranges. The Rows node contains the actual data returned by the query object. Each individual set of data, or row corresponding to the columns described in the Column node is returned as a separate data element within a Row node.

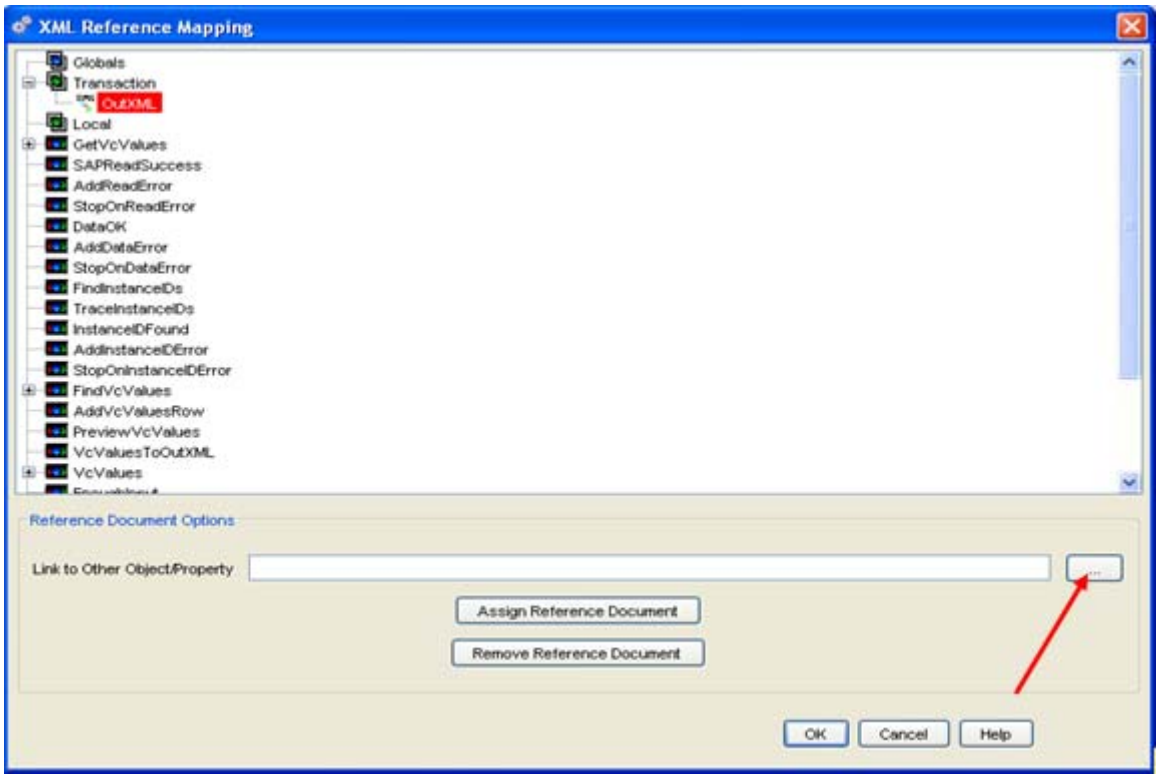


Step 3:

Click the Reference->Assign Reference Document menu.

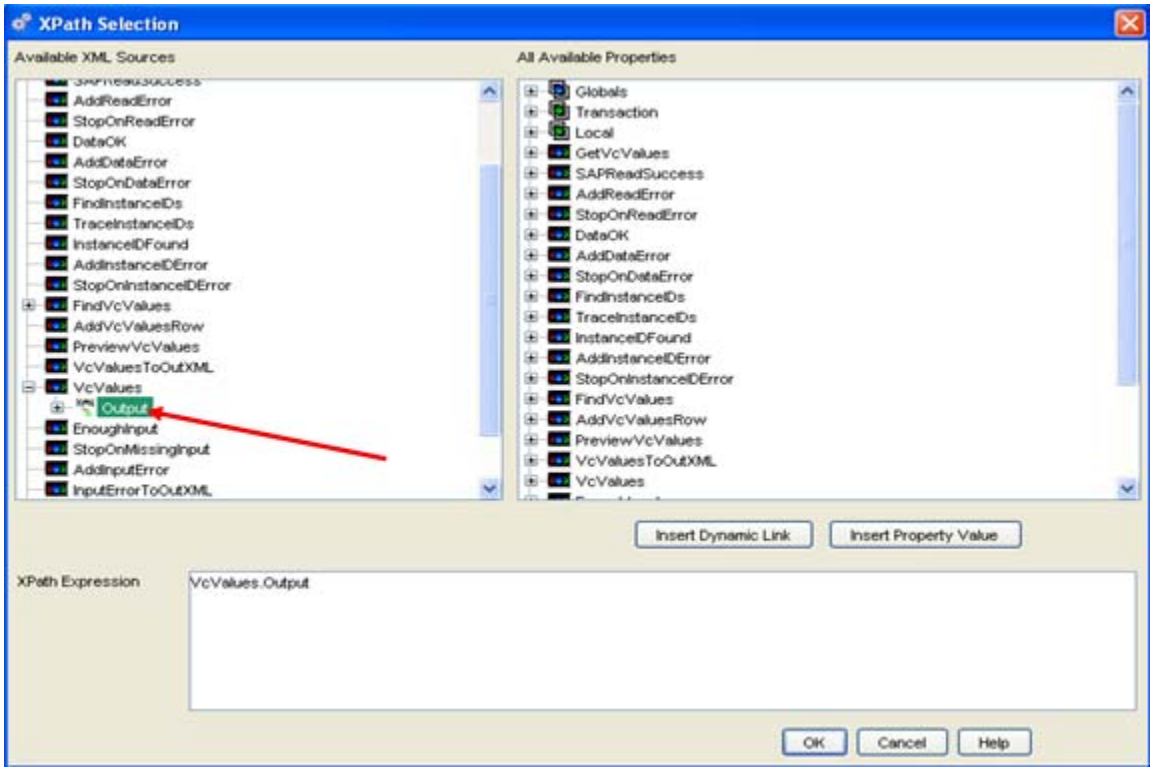
Select the output parameter (outXML) from the resultant window.

Click on the 'Browse' button pointed in the image below.



Step 4:

Expand and select the SAP MII Document object described in step 2 , select the 'Output' node and click 'Ok' (see the image below). This will take you back to the window in step# 3 with the link filled-in. Click the 'Assign Reference Document' button and click 'Ok'.

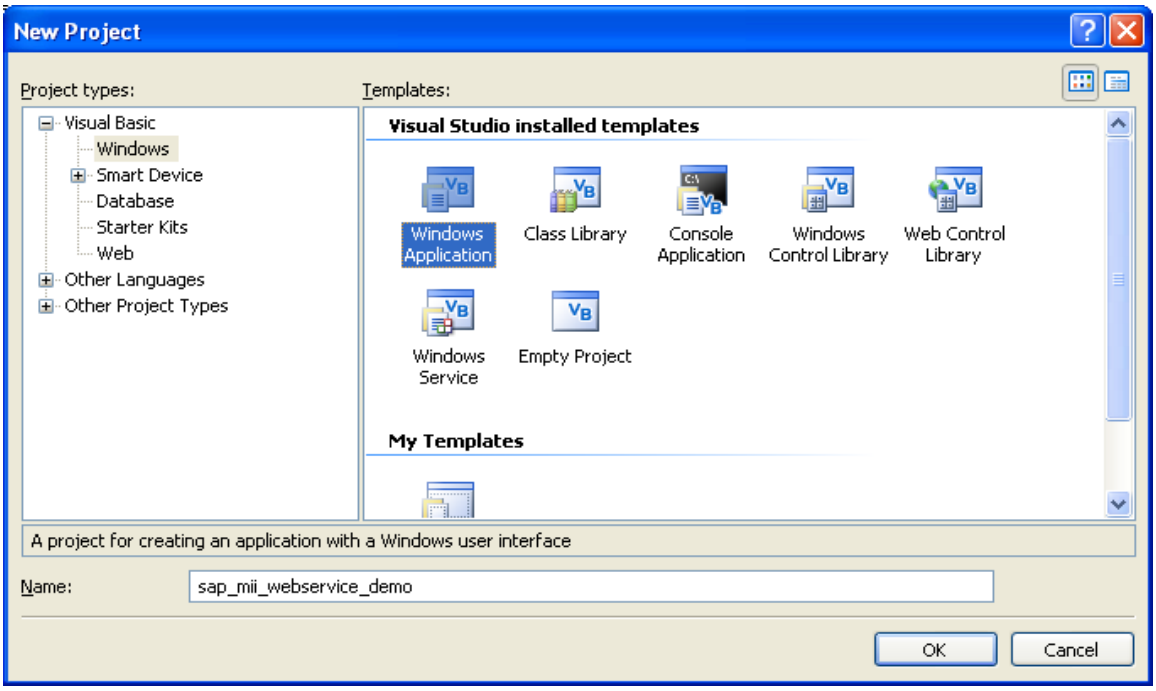


If the reference document is correctly assigned, you will see the complete structure of the output transaction property - OutXML while expanding the node from the link editor. The same will be visible when this transaction is called from another transaction.

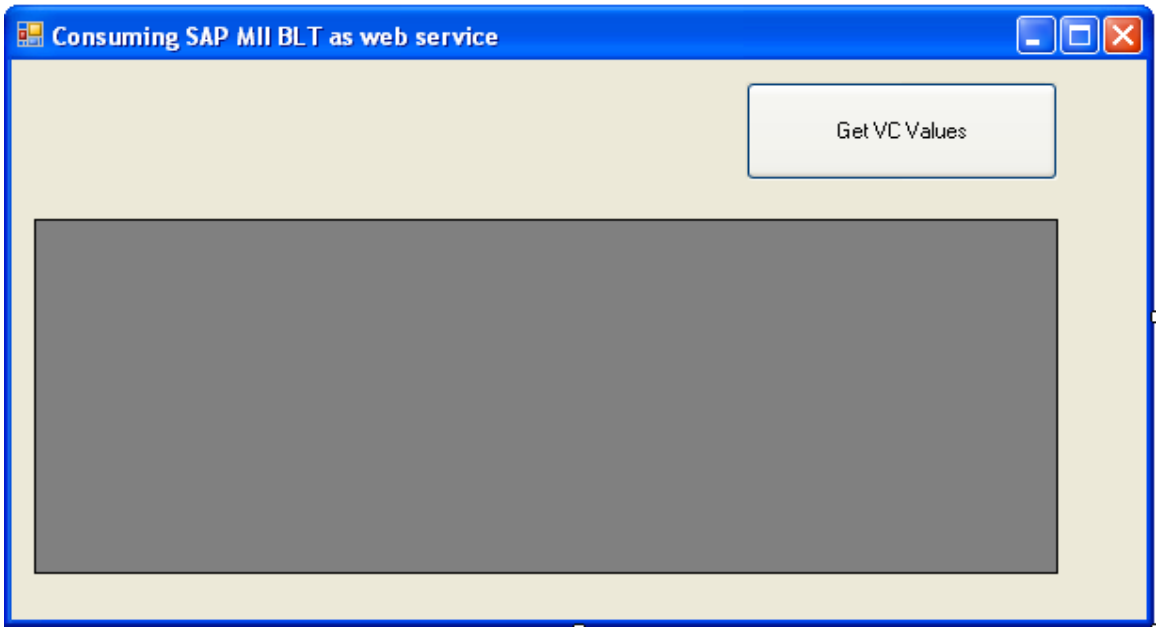
Consuming the Web Service from Visual Basic .Net

Visual Basic .Net is an object-oriented development tool which supports significantly expanded use of XML both internal to Visual Studio .Net and externally as a medium of data inter-change. Lets take a look on how we can access a web service (it is important to note that XML is the corner stone of all web services) from a Visual Basic .Net application.

Open your Visual Studio .Net , create a new Windows Application and let us call the new project as **sap_mii_webservice_demo**



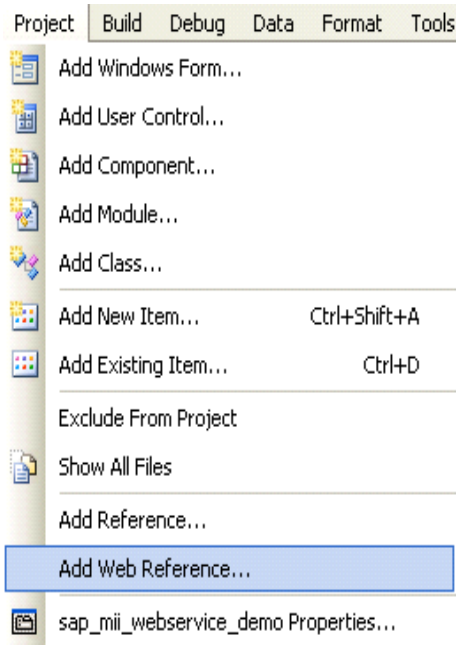
Now lets add a button and a DatagridView to the form.



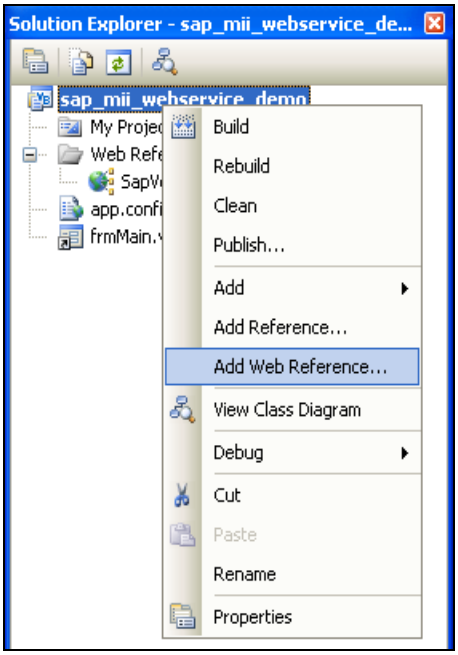
Adding Web Reference

We will now add the Web Reference to our SAP MII Business Logic Transaction. This is almost like adding a reference to a COM/ActiveX objects. By adding a Web Reference we will get access to the business logic transaction which is basically a XML web service on the SAP MII server.

Click on 'Add Web Reference' on the 'Project' menu. This will open the 'Add Reference' dialog for you to enter the URL for the web service.



Alternatively, you could use the 'Add Web Reference' from the Solution Explorer window by right clicking the project.



Type-in the URL of your web service on the resultant window:

```
http://<< your server >>/Lighthammer/WSDLGen/<<your transaction>>
```

or

```
http://<< your server >>/XMII/WSDLGen/<<your transaction>> (for Version 12)
```

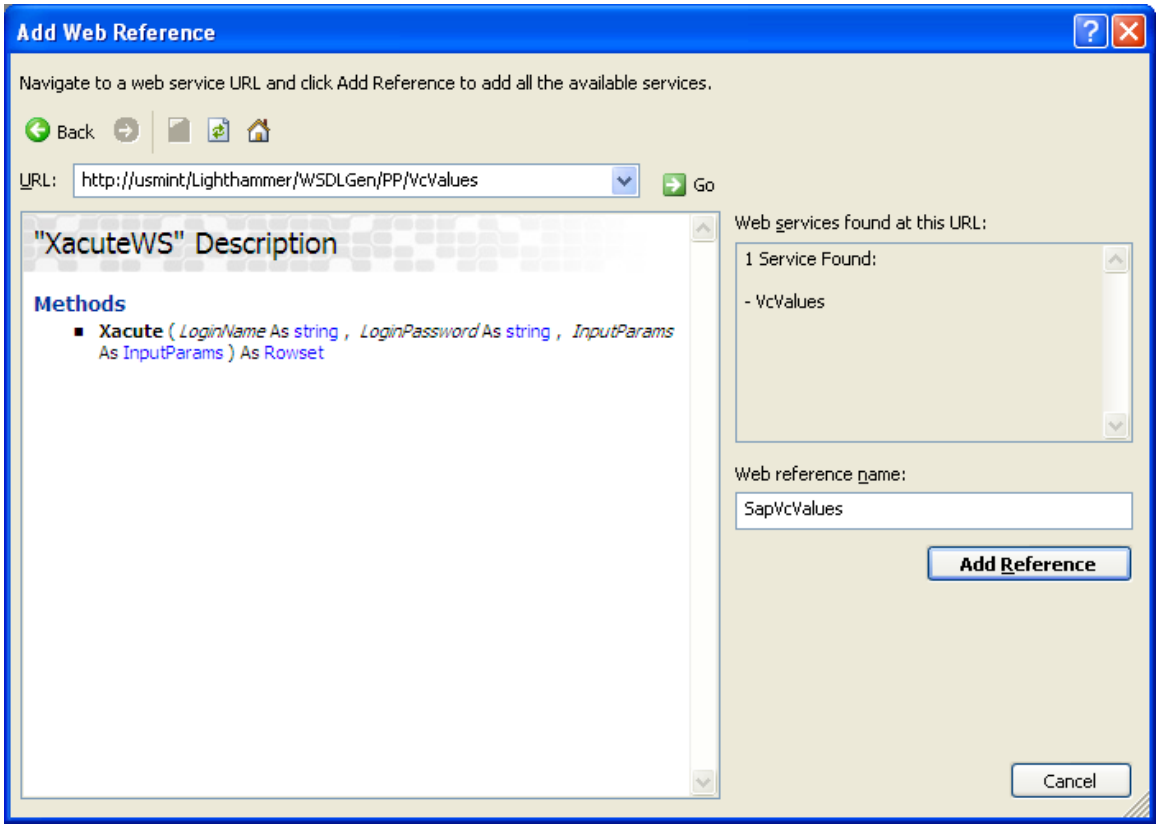
In our example, http://usmint/Lighthammer/WSDLGen/PP/VcValues , where usmint is the SAP MII server name and click the 'Go' button:

As seen in the image below, you now have access to the Xacute Method of the VcValues service. In the Web reference name field, enter a name that you will use in your code to access the selected web service programmatically. Lets call the web reference as SapVcValues.

The Xacute method requests three parameters:

LoginName	-	Your SAP MII Login Name
LoginPassword	-	Your SAP MII Login Password
InputParams	-	Input parameters - as defined in the Transaction->Properties In our example, GUID and MaterialNumber

Click the Add Reference button to create a reference in your application to the selected Web service - VcValues. The new reference will appear in the Solution Explorer under the Web References node , named as 'SapVcValues'.



Source Code

Now that you have added the web reference to the business logic transaction, invoking the Xacute method is simple and can be done by writing just a few lines of code. Add the following code to the 'Get Vc Values' button click event.

```
Dim wsVcValues As New SapVcValues.XacuteWS
Dim myInParams As New SapVcValues.InputParams
Dim varResult As SapVcValues.Rowset
'Pass your input parameters to the transaction
myInParams.GUID = "2E64F437FF33BD42AC0CD69B716C7B76"
myInParams.MaterialNumber = "000000000000687453"
'Now lets call the transaction using the guest account
varResult = wsVcValues.Xacute("guest", "guest", myInParams)
grdResult.DataSource = varResult.Row
grdResult.Refresh()
varResult = Nothing  'Cleanup
```

The above code will populate the result XML to a grid. If you would like to parse or process the XML result,

```
Dim iIndex As Integer
Dim sCharDesc As String
Dim sCharValue As String
For iIndex = 0 To varResult.Row.Length - 1
    'Charactersitics and Value are columns defined in the IllumDoc
    sCharDesc = varResult.Row(iIndex).Characteristics
     sCharValue = varResult.Row(iIndex).Value
    ……..
Next
```

Conclusion

Today's developers face the challenge of targeting a broad range of platforms and crafting applications that quickly deliver value to the business. Instead of rewriting the complex applications entirely, one may choose to integrate the existing application by exploiting the web service interface capability that SAP MII offers to build connected applications. Similarly, the SAP MII's web service interface can be well utilized by the existing high-performance non-SAP applications for interfacing with SAP in real time and quickly react to the demands of the business while taking advantage of advanced concepts/technologies like multi-threading, true inheritance, Language Integrated Query (LINQ), run-time diagnostics etc. with which these applications are built. There is no doubt about the integration capabilities that SAP MII offers and it's ability to develop rich, interactive and powerful web applications but as many of the pundits have noted, success on the web is built on successful partnerships.

Related Content

SAP xMII help

SAP xMII Getting Started Guide

For more information, visit the Web Services homepage.

For more information, visit the Manufacturing homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.