Microsoft Dot Net to SAP BI Interoperability Reporting using SAP Dot Net Connector 3.0

Applies to:

SAP BI 7.0, SAP Dot Net Connector 3.0 and Microsoft Dot Net Framework 3.5 Service Pack 1. For more information, visit the EDW homepage

Summary

This article explains about how to communicate between SAP BI and Microsoft Dot Net using SAP Dot Net connector. The main motive of this paper is to read the data from SAP system and then bind it to the Dot net system. Firstly to establish the connection between the two systems, Call the required functions, manipulate the data from SAP system to Dot Net and finally to get the result set in the desired form in the front end using any reporting tool.

Author: Raghavendra Nagesh D Y and Dominic Savio S

Company: PricewaterhouseCoopers Pvt. Ltd.and Olam Information Services Pvt. Ltd.

Created on: 9 June 2011

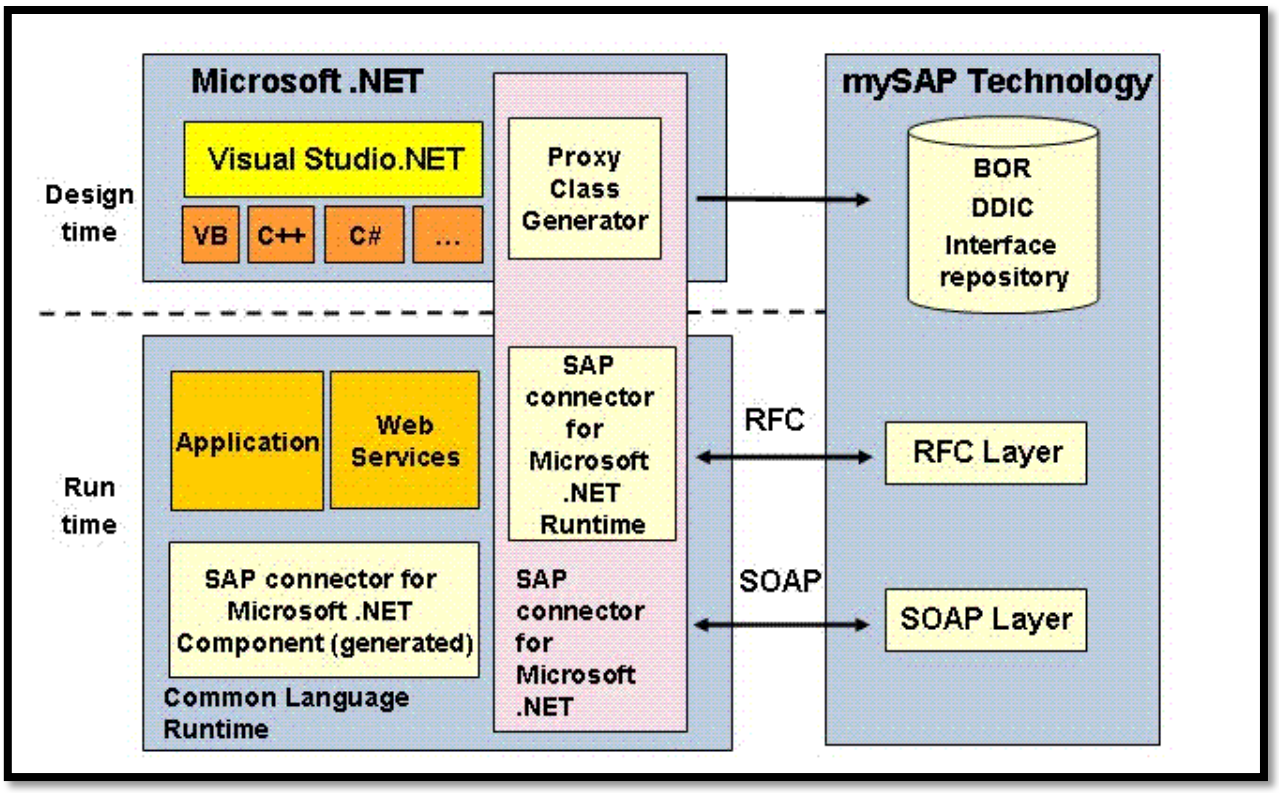
Author Bio

Raghavendra Nagesh, is currently working as a Principal Consultant in SAP BI BO Technologies in PricewaterhouseCoopers, Bangalore India.

Dominic Savio, is currently working as a Consultant in Dot Net Technologies in Olam Information Services Private Limited., Chennai India.

Table of Contents

Architecture of SAP Connector for Microsoft dot net



At run-time the SAP Connector for Microsoft .NET performs and manages all necessary communication between the .NET application and the SAP server. Transparent for the application both SOAP and SAP RFC can be used as communication protocols. Transactional RFC or queued RFC are supported. For communication security we can use Kerberos, X.509 certificates or Microsoft Passport.

SAP Dot Net Connector Overview

The main features of SAP Dot net Connector is as follows.

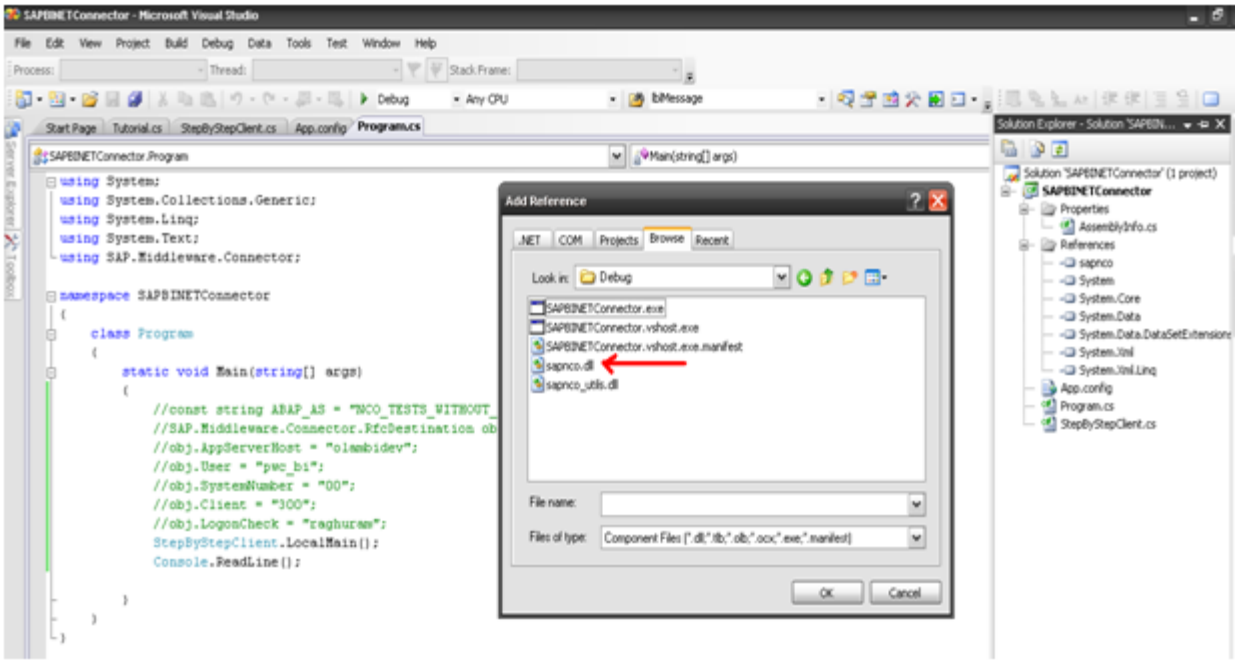
- NCo (SAP Dot net Connector) no longer distinguishes between a design time and a runtime. Instead of proxy classes and generated coding, you now program RFC calls dynamically. This has advantages (less and more easily understandable coding, robustness against changes on backend side - e.g. it's no longer necessary to re-generate the proxies and re-compile your application, if the backend moves from non-Unicode to Unicode, no dependency on a fixed Visual Studio release) as well as disadvantages (no IntelliSense support, you need to know, how the ABAP side looks like, when consuming RFMs in .NET).

- RFC protocol is re-implemented in C#, so there is no dependency on librfc32.dll any longer. This should result in better performance, as almost no marshalling between managed and unmanaged code is now necessary.

SAP to Dot net Connection and Data Output

The first step is to establish connection between SAP BI and the Dot net system. This is achieved by using the DLL, SAP Dot net Connector 3.0. Add this version of DLL to the project added in the Dot net Solution.

Create an application in Dot net and add the SAP reference library to the created solution in Dot net.



Make the pooled connection by giving the appropriate SAP system instance, host, and client user id and password.

Configure the configuration file in the Dot net screen. This is for e.g. from which SAP server needs to be connected, Host userid and password and so on.

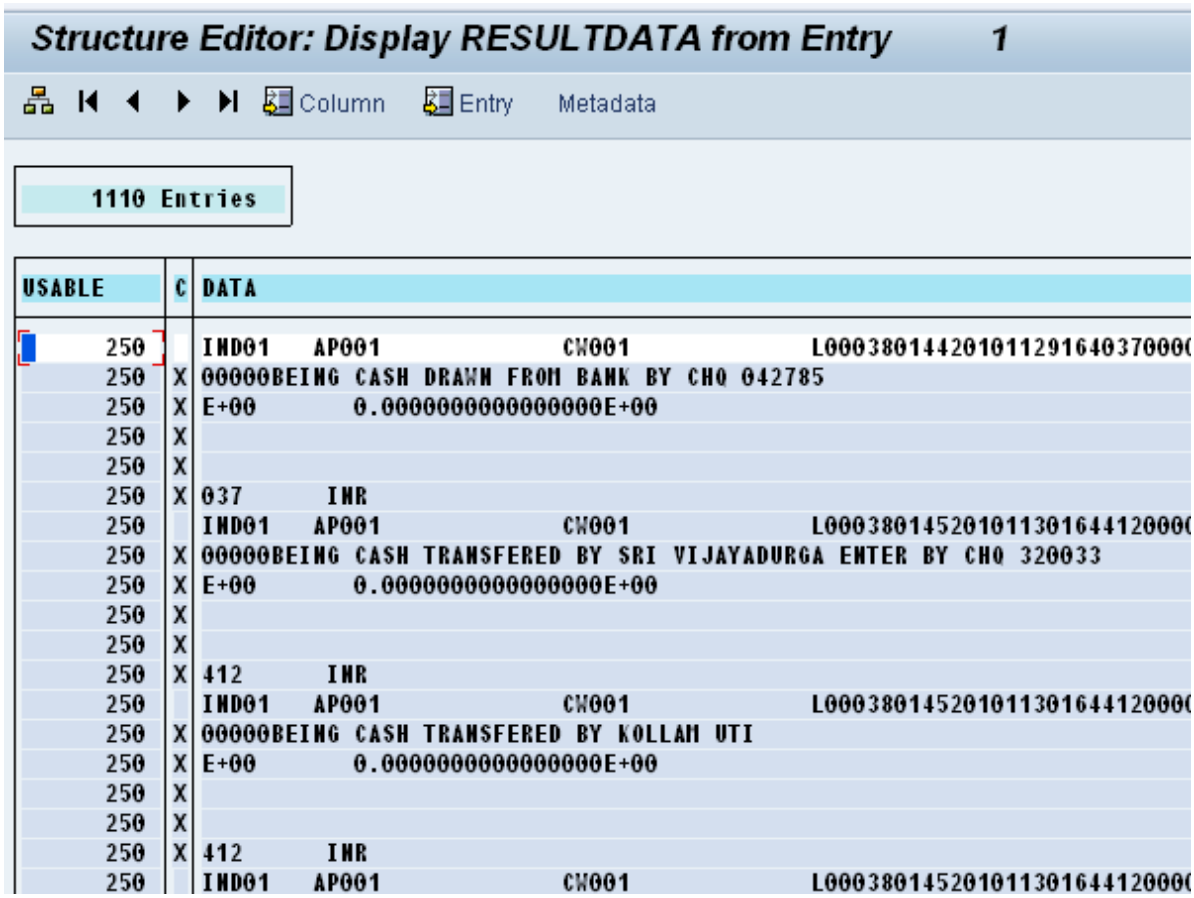
```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="SAP.Middleware.Connector">
      <sectionGroup name="ClientSettings">
        <section name="DestinationConfiguration" type="SAP.Middleware.Connector.RfcDestinationConfiguration,
                  sapnco"/>
      </sectionGroup>
      <sectionGroup name="ServerSettings">
        <section name="ServerConfiguration" type="SAP.Middleware.Connector.RfcServerConfiguration, sapnco"/>
      </sectionGroup>
    </sectionGroup>
  </configSections>
  <SAP.Middleware.Connector>
    <ClientSettings>
      <DestinationConfiguration>
        <destinations>
          <add NAME="NCO_TESTS_WITHOUT_POOL" USER="pwc_bi" PASSWD="raghuram" CLIENT="300" LANG="EN" ASHOST="192.168.0.72" SYSNR="00" MAX_PO
          <add NAME="NCO_TESTS" USER="pwc_bi" PASSWD="raghuram" CLIENT="300" LANG="EN" ASHOST="192.168.0.72" SYSNR="00" MAX_POOL_SIZE="10"/
          <add NAME="NCO_LB_TESTS" USER="NCOTEST" PASSWD="myPassWord" CLIENT="100" LANG="EN" MSHOST="hostsys1" SYSID="PRD" GROUP="PUBLIC" M
        </destinations>
      </DestinationConfiguration>
    </ClientSettings>
    <ServerSettings>
      <ServerConfiguration>
        <servers>
          <add NAME="NCO_SERVER" GWHOST="hostsys1" GWSERV="sapgw00" PROGRAM_ID="NCoServer" REPOSITORY_DESTINATION="NCO_TESTS" REG_COUNT="1"
```

After establishing connection between the two systems. Call the RFC Destination class from the provider.

Call the BAPI "**BAPI_ODSO_READ_DATA_UC**" and pass the attributes to the BAPI function "**ODSOBJECT**" and "**SELECTALLINFOOBJECTS**" as 'A'.

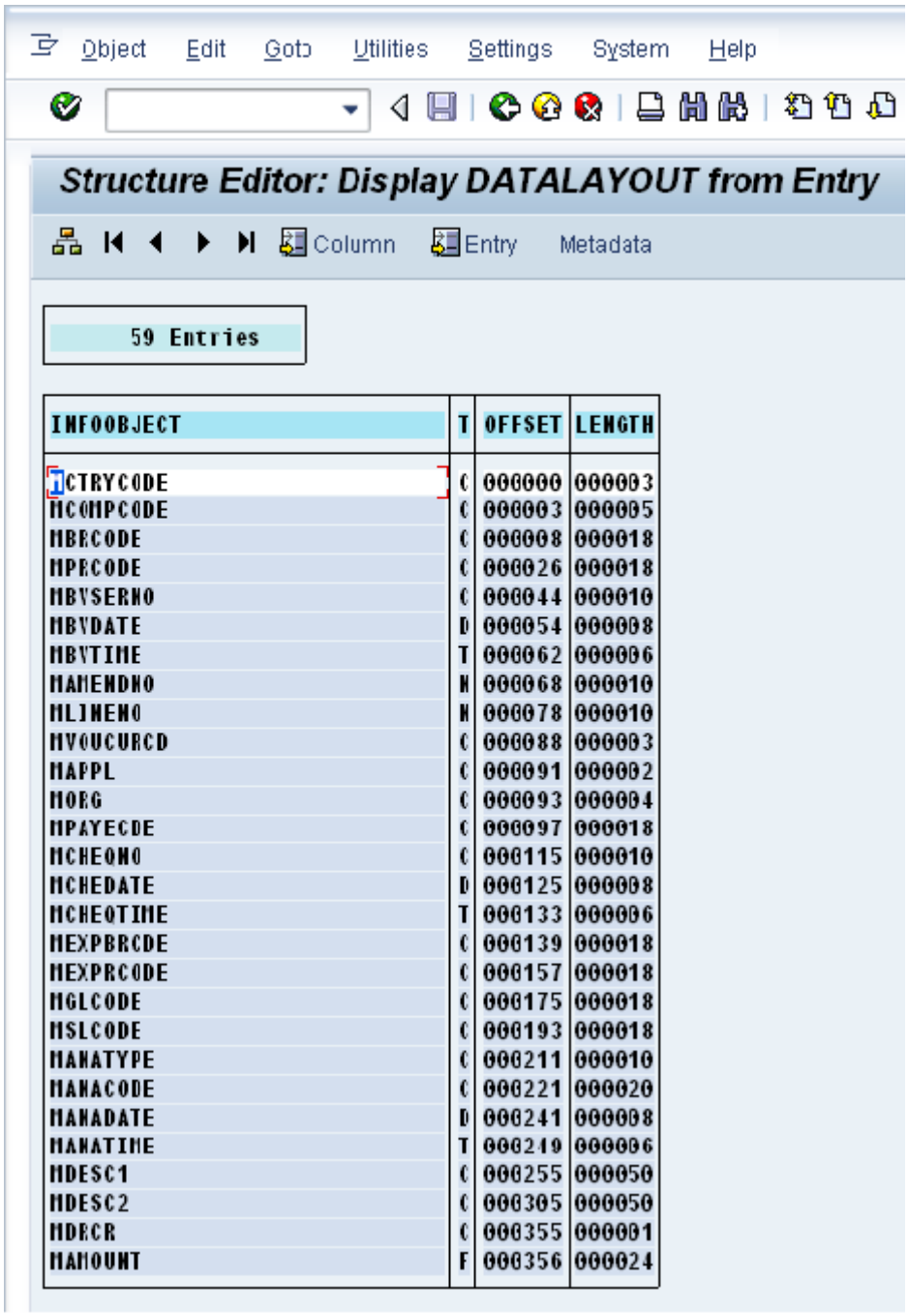
Invoke the above BAPI, it returns the output as IRFCTABLE an RFC function output. From the data output the results are computed in the required. The sample data manipulation is shown in the sample code in the next step below.

The below is the sample SAP data while calling the BAPI.



Note: Only partial output is shown in the above snapshot. The actual output may vary depending on the business requirement.

The sample data layout will look like as below



INFOOBJECT	T	OFFSET	LENGTH
0CTRYCODE	C	000000	000003
0COMPCODE	C	000003	000005
0BRCODE	C	000008	000018
0PRCODE	C	000026	000018
0BVSERNO	C	000044	000010
0BVDATE	D	000054	000008
0BVTIME	T	000062	000006
0AMENDNO	N	000068	000010
0LINENO	N	000078	000010
0VOUCURCD	C	000088	000003
0APPL	C	000091	000002
0ORG	C	000093	000004
0PAYECDE	C	000097	000018
0CHEQNO	C	000115	000010
0CHEDATE	D	000125	000008
0CHEQTIME	T	000133	000006
0EXPBRCDE	C	000139	000018
0EXPRCODE	C	000157	000018
0GLCODE	C	000175	000018
0SLCODE	C	000193	000018
0AMATYPE	C	000211	000010
0AMACODE	C	000221	000020
0AMADATE	D	000241	000008
0AMATIME	T	000249	000006
0DESC1	C	000255	000050
0DESC2	C	000305	000050
0DRCR	C	000355	000001
0AMOUNT	F	000356	000024

Sample Code

The below code illustrates how to establish the connection and then call the BAPI function to return the data layout.

This is an example of a code sample:

```
RfcDestination destination =      RfcDestinationManager.GetDestination(ABAP_AS_POOLED);
String returnCode;
IRfcFunction function = null;
try
{

function =   destination.Repository.CreateFunction("BAPI_ODSO_READ_DATA_UC");
function.SetValue("ODSOBJECT", "IBVOU_IN");
function.SetValue("SELECTALLINFOOBJECTS", 'A');
function.Invoke(destination1);
}

catch (Exception ex)
{
Console.WriteLine(ex.ToString());
return;
}
//Get the structure RETURN (common for BAPI functions)
IRfcStructure returnStructure = function.GetStructure("RETURN");
returnCode = returnStructure.GetString("TYPE");
if (returnCode.Equals("E") || returnCode.Equals("A"))
{
      throw new Exception(returnStructure.GetString("MESSAGE"));
}
      IRfcTable codes = function.GetTable("RESULTDATA");
      //Iterate over all rows in the table BAPI
System.Data.DataTable dtSAPDataTable = new System.Data.DataTable();
DataColumn col1 = new DataColumn("USABLE");
DataColumn col2 = new DataColumn("CONTINUATION");
DataColumn col3 = new DataColumn("DATA");
dtSAPDataTable.Columns.Add(col1);
dtSAPDataTable.Columns.Add(col2);
dtSAPDataTable.Columns.Add(col3);
for (int i = 0; i < codes.RowCount; i++)
{
     //set the current row - the row used for Get*/Set* operations
codes.CurrentIndex = i;
System.Data.DataRow  newDataRow = dtSAPDataTable.NewRow();
newDataRow["USABLE"] = codes.GetString("USABLE");
newDataRow["CONTINUATION"] = codes.GetString("CONTINUATION");
newDataRow["DATA"] = codes.GetString("DATA");
dtSAPDataTable.Rows.Add(newDataRow);
}
}
```

With the above code the output from the result layout is obtained. It is then compared with the data layout and the calculations are made on the manipulated data output. And then depending on our business requirements the output data table in Dot net is formulated.

The final output is given in the form of data table to any front end reporting system in Dot Net.

Related Content

http://www.sdn.sap.com/irj/sdn/dotnet

SAP RFC Server Programming

http://www.se80.co.uk/sapfms/b/bapi/bapi_odso_read_data_uc.htm

Blog: A Spotlight on the New .NET Connector 3.0

For more information, visit the EDW 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.