SAP Web Services and Microsoft Office: How to Create Simple Composite Applications

Applies to:

SAP NetWeaver 2004s, SAP ECC6.0, Microsoft Office 2003 with Web Services Toolkit 2.0

Summary

With web services, it is now possible to create composite applications that integrate SAP functionality in Microsoft Office front-ends in a simple and straightforward matter. This article will show how to build small applications using Excel and VB for applications.

Author(s): Marcus Schiffer

Company: Degussa AG

Created on: 23 August 2006

Author Bio

Marcus Schiffer works with the Corporate IT department of Degussa AG. He is involved in developing SOA strategies.

Table of Contents

Integrating Microsoft and SAP

The integration of SAP and Microsoft Office has been a topic in IT departments for a long time. Typically、data is retrieved from SAP to be manipulated and enriched in Excel tables. The data is eventually sent back to SAP to create or update business documents. Numerous examples can be found in larger companies like the famous creation of customer orders, confirmation of working times in the project system or posting contact reports from Word to a CRM system. Today, Project DUET even offers ready-to-use products for SAP and Microsoft integration.

The technology for most of these applications has been available for a long time. In Microsoft Visual Basic (VB) and Visual Basic, for applications references to SAP-delivered libraries (e.g. the SAP remote function call (RFC) library or in former times the BAPI OCX) can be included. From within the VB for applications code in Microsoft Office these libraries were invoked and connections to SAP could be made. Due to the propietory nature of this technology, application development based on these methods was always an expert's task, requiring much experience in SAP as well as in VB.

With enterprise Service-Oriented Architecture (SOA), however, there is an attractive alternative for interfacing SAP and Microsoft Office that depends on easy-to-use tools and industry standards (cf. Ref. 1). In the following demonstration scenario, a simple example is given that can easily be adopted for any other scenario.

Preparing the Demo Scenario

To demonstrate the web service-based interfacing between SAP and Microsoft Office, we want to build a small example using an SAP ECC 6.0 system and Excel Version 2003.

In Excel, a set of data will be created offline and sent to SAP for posting after reconnecting to the company network. The SAP document type "Sales Activity" was chosen with the respective BAPI for creation of activities (BAPI_BPCONTACT_CREATEFROMDATA).

In all of the following, the attention is focused on the handling of the complex table types in VB for applications. Most BAPI functions use table parameters that need to be filled and read by the application proxy. The use of simple types is more or less straightforward.

EXCEL and the Web Services Toolkit

First, the Microsoft Excel environment has to be prepared. For use with Office 2003, Microsoft delivers the "Web Services Toolkit 2.01" for download at:

http://www.microsoft.com/downloads/details.aspx?FamilyID=fa36018a-e1cf-48a3-9b35-169d819ecf18&DisplayLang=en

After the installation, the Web Services plug-in can be found in the Excel VB editor under "Extras->Web Service References".

Excel is now ready to import WSDL files for the automatic creation of classes and data types to call the corresponding web services.

Creating the Web Service in SAP

So the next step is to generate a web service from the SAP BAPI module and to expose it as WSDL file. See also (Ref. 2).

Before we proceed, it is necessary to mention that in Visual Basic, the word "TYPE" is restricted. So when generating a WSDL with restricted words in the type definitions, the Web Services Toolkit will not work properly. Unfortunately, the typical BAPI interface includes the BAPIRET2 structure with a field called TYPE. One way to overcome the restriction is to use the type mapping in the web service virtual interface as described in the previously mentioned tutorial. We choose as an alternative to use a copied version of the BAPIRET2 structure with a renamed TYPE field.

To prepare the interface the BAPI is encapsulated with the BAPI_TRANSACTION_COMMITT in a new function module in the customer name space (Z_ BAPI_BPCONTACT_CREATEFROMDATA). The interface of the function that will serve as basis for the Web service consists of the following parameters:

```
*"  IMPORTING
*"     VALUE(SENDER) TYPE  BAPI_SENDER
*"     VALUE(TESTRUN) TYPE  STRING
*"  TABLES
*"      GENERALDATA STRUCTURE  BAPI_BUS1037_VBKAKOM_CR
*"      BUSINESSPARTNER STRUCTURE  BAPI_VBKA_VBPA2KOM OPTIONAL
*"      RETURN STRUCTURE  ZBAPIRET2 OPTIONAL
```

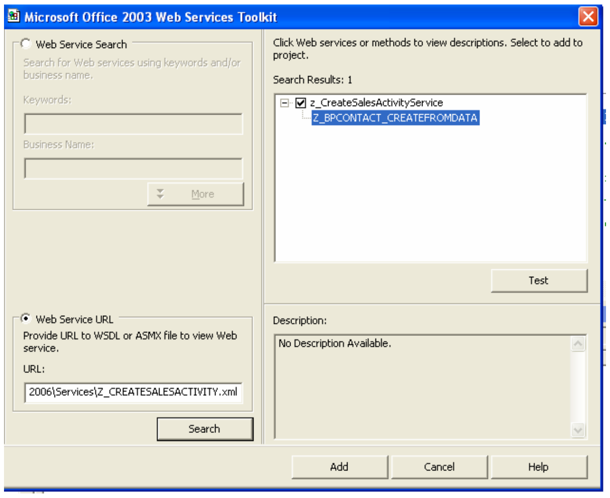
Note that a few interface parameters have been removed to keep it simple. From the menu “Utilities->Create Web Service”, it is now easy to expose the new function module as a web service. Simply following the wizard, naming the service and checking the “release service for runtime” box is enough to find the service as an active component in the transaction WSADMIN.

In order to avoid “deserialisation error” messages, it may also be worthwhile to change all type definitions to upper case. This can be done by unchecking and rechecking the “exposed” flag in the Web Service interface. Now all field names are in upper case.

From the transaction WSADMIN, a WSDL file can be generated. Just indicate the service and choose the WSDL button from the menu. Save a copy of the WSDL file to your local computer choosing the .wsdl or .xml extension.

In the last step, a user name and password can be provided in the service interface in transaction SICF to allow easy access to the service (cf. Ref. 2).

Building the EXCEL application



Search for Web services using keywords and/or business name.

Provide URL to WSDL or ASMX file to view Web service.

URL: 2006\Services\Z_CREATESALESACTIVITY.xml

Click Web services or methods to view descriptions. Select to add to project.

Search Results: 1

z_CreateSalesActivityService
Z_BPCONTACT_CREATEFROMDATA

No Description Available.

Now everything is prepared for the EXCEL application.

From the VB editor, choose “Web Service Reference” and mark the “Web Service URL” checkbox. Type the full address of the WSDL file (e.g. C:\Z_CREATESALESACTIVITY.xml). A search result should appear.

If the WSDL file is correctly spelled and no search result appears, the WSDL file may be incorrect. One reason may be that in SAP, the input and output types are different (e.g. upper case for input fields, lower case for output fields).

Check the service and press the “add” button. The toolkit will parse the file and create type definitions and classes in the VB editor.

Wizard results

After the upload of the WSDL file, the toolkit has added new classes of different types to the EXCEL project. Some classes represent the table structures. In the example, the interface consists of the tables and other simple import structures leading to the structure classes (for the businesspartner tables e.g.:`struc_ BAPIBUS1037VBKAKOMCR).`

Two other classes provide the objects for working with the web service. The method for calling the service in the example is in a class called

```
clsws_zcreatesalesactivity
```

and here the method for calling the service is implemented:

```
 Public Sub wsm_Z_BPCONTACT_CREATEFROMDATA(ByRef ar_BUSINESSPARTNER As
Variant, ByRef ar_GENERALDATA As Variant, ByRef ar_RETURN As Variant,
ByVal obj_SENDER As struct_BAPISENDER, ByVal str_TESTRUN As String)
```

Note that all table parameters are of type “variant”, simple structures are of respective type “struc_xxx” and simple variables are of type “string”.

In case of individual other applications, the classes and type definitions need to be checked: All tables and structure parameters from the function interface should appear as class modules named struc_xxx, including the type definitions for all fields. If not, check for restricted words in the data definitions of the WSDL file.

One thing needs further attention. In the SAP-generated WSDL file, SAP fields of type “date” may be represented as

```
<xsd:simpleType name="date">
<xsd:restriction base="xsd:string">
  <xsd:maxLength value="10" />
    <xsd:pattern value="\d\d\d\d-\d\d-\d\d" />
```

```
      </xsd:restriction>
      </xsd:simpleType>
```

....

```
        <xsd:element name="FROM_DATE" type="tns:date" />
```

These fields are converted by the Web Services Toolkit in Excel according to

```
            Public FROM_DATE As date
```

To avoid “deserialisation failed” errors in the Web Service call, all “date” fields must be redefined to “string”.

```
            Public FROM_DATE As String
```

Calling the Service

To call the web service, an object needs to be created:

```
      Dim SalesActivityWS As New clsws_zCreateSalesActivityS
```

Then all input tables need to be filled. This can be achieved for all tables as follows. First declare the table variables as “Variant”. For example, for the businesspartners:

```
      Dim ar_BUSINESSPARTNER as varinat
```

Additionally, a variable of type struc_`BAPIBUS1037VBKAKOMCR` is needed to fill the input data. To account for more than one entry, the variable is defined as array.

```
      Dim _StrucBusinesspartner(2) as struc_BAPIBUS1037VBKAKOMCR
```

Now the structure _StrucBusinesspartner can be filled from a corresponding EXCEL table or by use of VB input forms according to the needs. Finally, the input variables are transferred:

```
      ar_BUSINESSPARTNER = _StrucBusinesspartner
```

`And the service is called.`

```
      Call SalesActivityWS. wsm_Z_BPCONTACT_CREATEFROMDATA ( ar_BUSINESSPARTNER,
      ar_GENERALDATA, ar_RETURN, obj_SENDER, str_TESTRUN)
```

After the execution of the service, all ar_RETURN fields can be evaluated.

Possible Errors

- If the “deserialisation error” occurs, check again for remaining fields of type “date” or lower case field names.
- Another error might read “too many open connections”. This means the service could not log on to SAP. Check the user name and password provided in transaction SICF.

Conclusion

With the Web Service Toolkit for Microsoft Office 2003, it is easy to create simple applications using VB for applications. If a few basic rules are acknowledged, the toolkit provides an easy to use environment for accessing SAP functionality through web services for retrieving and posting data.

Related Content

- Standards in Web Service Development
- Web Service Settings in SAP NetWeaver 2004s

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.