Consuming SAP Enterprise Services in a Microsoft Office

InfoPath Form

Applies to:

- SAP Enterprise SOA
- Microsoft Office InfoPath 2007
- Technical consultants, developers

Summary

This article demonstrates interoperability features of SAP Enterprise SOA in its integration with Microsoft Office tools. It introduces a sample InfoPath form template that consumes SAP Enterprise Services to retrieve information from back-end systems.

Authors: Tarnoruder Anne, SAP NetWeaver Product Management, SAP Labs Israel

Dudu Benabou, Microsoft Consulting Services, Microsoft Israel

Company: SAP

Created on: February 2007

Authors Bio

Anne Tarnoruder is a part of the SAP NetWeaver Product Management team in SAP Labs, Israel, responsible for knowledge transfer in the areas of PDK for .NET and Visual Composer SDK. Prior to joining SAP in 2004, she held a number of positions in software development, team management, system architecture and technical communications in various high-tech companies. Anne holds an M.S. degree in Applied Mathematics.

Dudu Benabou is a solution architect working as part of MCS (Microsoft Consulting Services) at Microsoft Israel for the Microsoft - SAP alliance, serving as a contact for Microsoft product groups in SAP Labs Israel and representing Microsoft Global ISV group.
# Table of Contents

Introduction ......................................................................................................................................3

About Microsoft Office InfoPath 2007 ...........................................................................................3

About SAP Enterprise SOA and Enterprise Services .................................................................3

Prerequisites ..................................................................................................................................3

Application ....................................................................................................................................4

Implementation .................................................................................................................................4

   Designing the Expense Report Form ..........................................................................................4

   Creating the Service Class .........................................................................................................8

   Finalizing and Distributing the Form Template .......................................................................11

Testing the Form ............................................................................................................................13

Copyright .......................................................................................................................................15
Introduction

This article demonstrates interoperability features of SAP Enterprise SOA (Service Oriented Architecture) in its integration with Microsoft Office tools. It introduces a sample Microsoft Office InfoPath 2007 form template that consumes SAP Enterprise Services to implement a business process.

About Microsoft Office InfoPath 2007

Microsoft Office InfoPath 2007 is an information-gathering program included in the 2007 release of the Microsoft Office system. With it, you can create and deploy electronic forms to gather information efficiently and reliably. You can also use the InfoPath Forms Services capabilities in Microsoft Office SharePoint Server 2007 to extend your business processes beyond your corporate firewall, delivering forms as Microsoft Office Outlook e-mail messages, Web browser forms, or forms for mobile devices.

About SAP Enterprise SOA and Enterprise Services

SAP Enterprise SOA is a blueprint for an adaptable, flexible, and open IT architecture for developing service-based, enterprise-scale business solutions. With SAP NetWeaver as a technical foundation, enterprise SOA moves IT architectures to higher levels of adaptability by elevating Web services to an enterprise level.

An enterprise service is typically a series of Web services combined with business logic that can be accessed and used repeatedly to support a particular business process. Aggregating Web services into business-level enterprise services provides a more meaningful foundation for the task of automating enterprise-scale business scenarios.

SAP Enterprise Services Repository offers a rich collection of services in various business domains, such as ERP, CRM or Financial Services. These services are published in the UDDI-compliant SAP Enterprise Services Registry and are available via the standard Web services interface.

Prerequisites

- Microsoft Office InfoPath 2007 (included with MS Office 2007 Ultimate, Professional Plus and Enterprise editions)
- Access to SAP enterprise services
Application

The InfoPath Expense Report form is used by employees to report travel expenses. To facilitate this business process, the form retrieves the static employee-related information, such as contact and organizational data from the back-end systems, and automatically populates the relevant fields.

The form template consists of two components representing the separate presentation and business layers: an interactive form and a class that implements the business logic by accessing enterprise services via their standard Web service interface. This class invokes service interfaces exposed by Web service proxies, automatically created by the development environment.

The form template is packaged for distribution either by email, or by deployment to a network location or to SharePoint services, where users can access it.

The following diagram illustrates the architecture of the Expense Report form template application.

![Diagram](image_url)

Implementation

The following section takes you through the steps required to implement and distribute the Expense Report form template. For the purposes of this implementation, we will select and customize an existing sample form template.

Designing the Expense Report Form

3. Rearrange the layout, fields and their labels so that the form template looks as follows:
4. Map the fields in the form to the XML schema in the Data Source panel. To open the panel, select View → Data Sources....
5. Attach an event handler to the UserID field by right-clicking it and selecting Programming → Changed Event. A Microsoft Visual Studio Tools for Applications window opens with the form code-behind module. The event handler declaration is added automatically:

```csharp
public void InternalStartup()
{
    EventManager.XmlEvents["/my:expenseReport/my:employee/my:userID"].Changed +=
    new XmlChangedEventHandler(sapName_Changed);
}
```

6. Create a simple login form by right-clicking the SAP Expense Report project in the Project Explorer and selecting Add New Item → Windows Form. Name the form Login. Drag and drop controls onto the form so that it looks as follows:
7. In the event handler, add code to show the login form, receive user credentials and call the `FillFormWithData` function to retrieve the data from the wrapper component and populate the form.

```csharp
public void sapName_Changed(object sender, XmlEventArgs e)
{
    // show login form
    Login loginDialog = new Login();
    loginDialog.UserName = e.NewValue;
    loginDialog.ShowDialog();
    if (loginDialog.FormCancel == true)
        return;
    string userName = loginDialog.UserName;
    string password = loginDialog.Password;
    loginDialog.Dispose();
    // Fill the form with data from SAP back-end system
    FillFormWithData(userName, password);
}
```
Creating the Service Class

The Service class encapsulates business logic provided by the enterprise services, in this case retrieval of employee and organizational data from the underlying back-end system. The class exposes the properties and methods used by the form code.

1. Open the SAP Expense Report project by selecting Tools → Programming → Microsoft Visual Studio Tools for Applications from the main menu.
2. Right-click the project in the Project Explorer and select Add Web reference… In the dialog box that opens, provide a URL to the service and the login credentials.
3. Add a reference to each of the following enterprise services:

<table>
<thead>
<tr>
<th>Enterprise Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WorkAgreementSimpleByElementsQuer yResponse_In</td>
<td>Reads an employee's work agreement IDs according to certain selection criteria. The output of the service is used as an input parameter for other services.</td>
</tr>
<tr>
<td>EmployeeNameByEmployeeQueryResponse_In</td>
<td>Returns employee names, form of address and other related parameters.</td>
</tr>
<tr>
<td>EmployeeCommunicationDataByEmployeeQueryResponse_InService</td>
<td>Returns employee contact information, such as phone numbers, email and fax.</td>
</tr>
<tr>
<td>OrganisationalCentreEmployeeSimpleByEmployeeQueryResponse_InService</td>
<td>Returns employee organizational center information, such as organizational unit and its location.</td>
</tr>
<tr>
<td>OrganisationalCentreCostCentreSimpleByOrganisationalCentreQueryResponse_InService</td>
<td>Returns employee cost center information.</td>
</tr>
<tr>
<td>ReportingLineManagerSimpleByEmployeeQueryResponse_InService</td>
<td>Returns an employee's manager name and contact information.</td>
</tr>
</tbody>
</table>

Local Web service proxies are created automatically when a Web reference is added.

Currently SAP supports only the Document style of WSDL, and not the Microsoft-specific RPC style, which causes a problem when consuming SAP services in the Microsoft environment. In this example, you need to apply a workaround that involves a minor correction of locally-stored WSDL files:

a. In the Project Explorer toolbar, click the Show all files icon and expand the Web reference folder.
b. For each Web reference in the project, perform the following steps:
c. Open the WSDL file, replace all instances of `<wsdl:part name="parameters"` with `<wsdl:part name="parameter"` (remove the s in parameters) and save the file.d. Right-click the Reference.map file and choose Run Custom Tool to invoke a utility that generates code for XML Web services. This recreates the local proxy.

4. Right-click the project in the Project Explorer and select Add → Class. Add a class module.
5. In the class, implement service calls to retrieve the necessary information. For example:

```csharp
public string EmployeeName()
{
    if (!workAgreementExists)
        return "NA - no work agreement";
    ECC_EmployeeNameByEmployeeQRService empNameService = new ECC_EmployeeNameByEmployeeQRService();
    empNameService.Credentials = _credentials;
    empNameService.Proxy = _webProxy;
    EmployeeNameByEmployeeQueryMessage empNameQueryMessage = new EmployeeNameByEmployeeQueryMessage();
    empNameQueryMessage.EmployeeNameSelectionByEmployee = new EmployeeNameByEmployeeQueryMessageEmployeeNameSelectionByEmployee();
    empNameQueryMessage.EmployeeNameSelectionByEmployee.WorkAgreement_ID = new WorkAgreementID();
    // Call the service...
    // Get employee name...
    return empNameQueryMessage.EmployeeNameSelectionByEmployee.EmployeeName;
}
```
empNameQueryMessage.EmployeeNameSelectionByEmployee.WorkAgreement_ID.Value = _workAgreementId;
EmployeeNameByEmployeeResponseMessage empNameResponse = empNameService.EmployeeNameByEmployeeQueryResponse_In(empNameQueryMessage);
string empName = empNameResponse.Employee.Name.DeviatingFullName;
if (empName == null) {
    empName = "NA";
    return empName;
}

6. Go back to the form code-behind module and implement the `FillFormWithData` function to retrieve the data from the wrapper component and populate the fields of the form:

```csharp
private void FillFormWithData(string userName, string password) {
    // initializing service
    System.Net.NetworkCredential credential = new System.Net.NetworkCredential(userName, password);
    service = new ESCall(credential, proxy);
    workAgreement = service.CreateWorkAgreement();

    // Call service to retrieve employee name and set the field value
    string serviceEmpName = service.EmployeeName();

    XPathNodeIterator empName = MainDataSource.CreateNavigator().Select("/my:expenseReport/my:employee/my:name", this.NamespaceManager);
    empName.MoveNext();
    empName.Current.SetValue(serviceEmpName);
    //fill other fields…
}
```

7. When done, build the project.
Finalizing and Distributing the Form Template

Configure the submit options of the form to send an email to the employee’s manager. In the Submit Button Properties dialog box, set the email destination to the managerEmailAddress field.

To enable the form to run in InfoPath, set the security options and digitally sign the content.

1. Go to Tools → Form Options and select Security and Trust.
2. Choose the Full Trust option and the Sign this form checkbox.
3. Click Select Certificate, select a certificate and click OK.

4. In the InfoPath design window, select View → Design Tasks. In the Design Tasks panel that opens, click Publish Form Template.
5. In the Publishing Wizard that opens, select To a network location. Provide the form name and the locations for its publication and access. Alternatively, you can select other publication options.
Publishing Wizard

Where do you want to publish the form template?

- To a SharePoint server with or without InfoPath Forms Services
- To a list of e-mail recipients
- To a network location
- As an installable form template (requires Microsoft Visual Studio)

< Back  Next >
**Testing the Form**

1. Open the form.
2. Type the employee ID in the user ID field, and move the cursor from the field. In the login box that opens, type your credentials and click **OK**.

With these credentials, the form calls enterprise services to bring the employee information and populate the relevant fields.
3. Fill in the missing data, save the form and click Submit. The form is sent by email to the manager. For more details, examine the accompanying code sample, available for download along with this article.