

SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & 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.

Applies To:

Interactive Forms based on Adobe software, Web Dynpro (WD), SAP NetWeaver Developer Studio (NWDS)

Summary

The purpose of this article is to share my experience with Interactive Forms based on Adobe software and to show you how to build an Adobe PDF form that invokes a Web service (WS). In this article, I describe how to create a form that consumes a Web service exposed by the SAP Web Application Server (WAS). I will also explain how to merge 3 WSDL files generated by the WAS into one file.

By: Gilles Atlan

Company: SAP, NPI

Date: 23 Jul 2005

Table of Contents

Scenario 1: Invoke WS from Adobe Acrobat Professional	3
Scenario 2: Invoke WS from a Web Dynpro application	3
Scenario 3: Invoke WS from Adobe Reader via Web Dynpro (design time)	3
The Web service invoked	4
Create a Web Dynpro project	5
Load a WSDL	6
Auto-generate fields.....	8
Redesign the UI	9
Build, Run, Test	9
Downloading the WSDLs:.....	11
4 Steps for merging the files:	11

Introduction

An interactive PDF form can invoke a Web service. This form that consumes a WS can be run as part of a Web Dynpro application but also by using the stand-alone Adobe Reader (available free of charge) under certain conditions. First, I will explain the different scenarios we have for invoking a WS, afterwards I will show you how to implement one of them.

Possible Scenarios

Scenario 1: Invoke WS from Adobe Acrobat Professional

You can create a PDF form using the stand-alone Adobe LiveCycle Designer (without Web Dynpro integration). You can create a Web service data connection to the form, but at runtime you will not be able to access WS through the form if you use the free Adobe Reader. You need Adobe Acrobat Professional to use the form properly. Acrobat requires a license fee.

Scenario 2: Invoke WS from a Web Dynpro application

You can build a Web Dynpro application that contains an interactive PDF form, which invokes a WS for passing and acquiring data. This PDF form is created in NWDS within a Web Dynpro project.

Scenario 3: Invoke WS from Adobe Reader via Web Dynpro (design time)

In order to have a PDF form interact with any Web service (exposed on the Internet, or by WAS, XI etc.) from Adobe Reader, you need to render this form on the SAP WAS. Once the file has been rendered as part of a Web Dynpro application, it can be saved locally (in PDF format) and distributed to anybody anywhere. This file can be opened in Adobe Reader, and can have a submit button that can invoke a WS. I will explain below how to build such a form.

Working with such a scenario, you can use a PDF form as a business process entry point (see picture below). The form can pass data to XI and start a business process.

The process flow is as follows:

1. The PDF form is created as part of a Web Dynpro application, then saved locally.
2. The PDF passes the data by invoking a WS exposed by the WAS.
3. The WS performs some business logic (EJB implementation) and passes the PDF data to XI by invoking a WS exposed by XI.

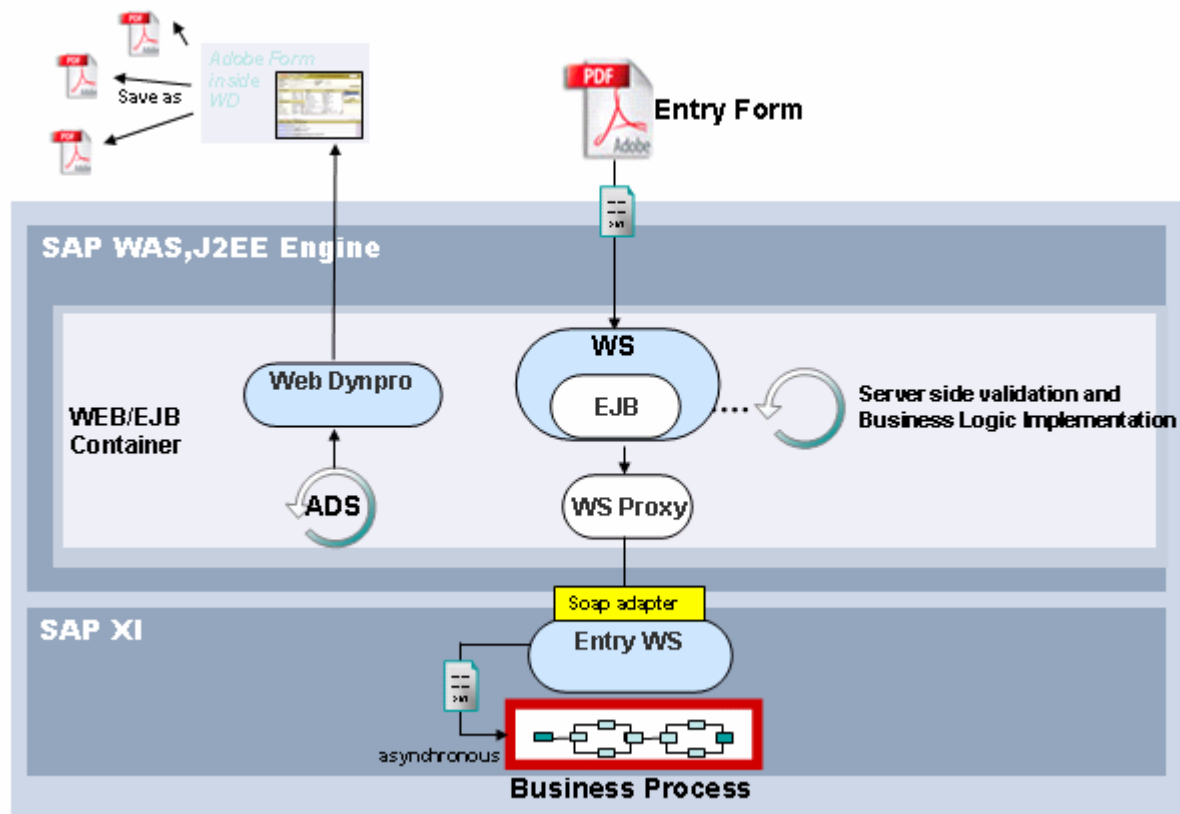


Figure 1: PDF form as a business process entry point

How to bind an interactive form to a Web service

The Web service invoked

We will create a form for passing customer details to an EJB through a WS. This “how-to” won’t describe how to build an EJB and expose it as a WS.

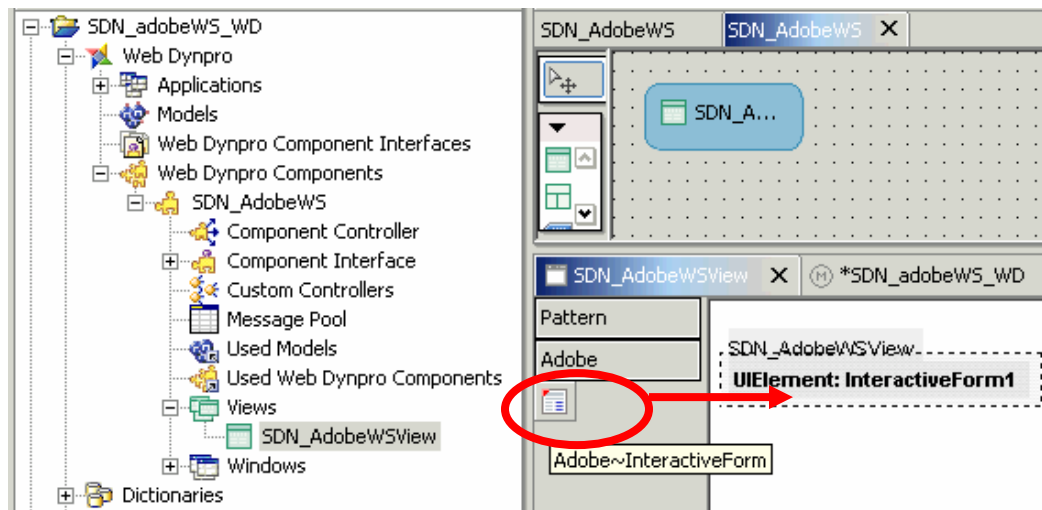
The interactive PDF form will invoke this simple EJB remote method exposed by the WS:

```
public String registerCustomer(String firstname, String lastname,  
                               String address, int phone)  
{  
    // TODO : Implement your business logic  
    return "Customer " + firstname + " " + lastname + " Registered";  
}
```

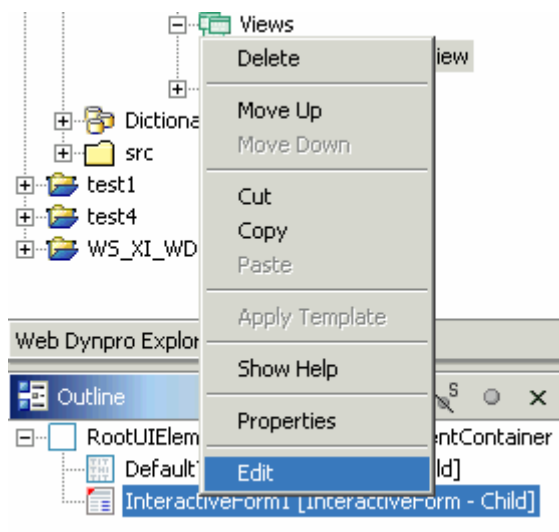
Create a Web Dynpro project

To be able to create an interactive PDF form in SAP NetWeaver Developer Studio, you need to first create a Web Dynpro project. In the Web Dynpro view, you insert an interactive form by dragging and dropping the InteractiveForm UI element onto the view.

See the InteractiveForm UI element in this screenshot:



The Adobe LiveCycle Designer is launched within the Developer Studio, and allows you to design your form. To launch the Designer, go to the "Outline" tab, select the InteractiveForm UI element, right-click it, and select "Edit" (see the following screenshot):



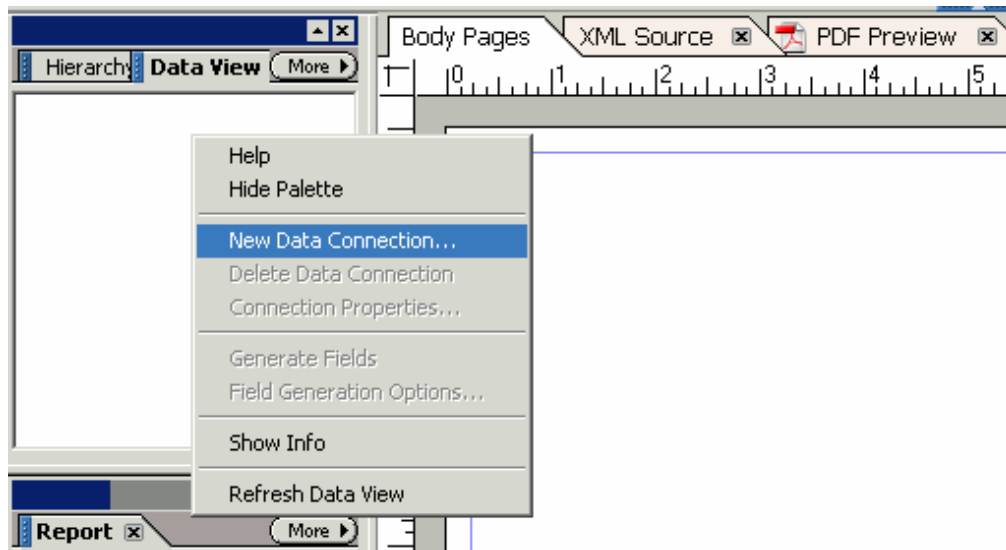
Note: The first launch of Adobe LiveCycle Designer sometimes requires a short wait. (The Designer that appears in NWDS is practically identical to the stand-alone version that you could launch from C:\Program Files\Adobe\Designer<6 or 7>\FormDesigner.exe. Remember, though, that if you use the

stand-alone Designer without the Web Dynpro integration for creating a form that calls a WS, the end user will not be able to call the WS from Adobe Reader (see scenario 1).

Load a WSDL

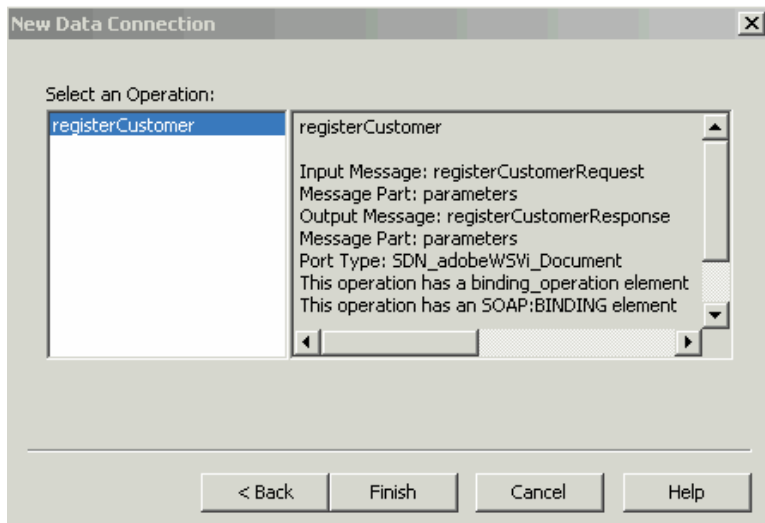
In order to be able to consume the WS from a PDF form, we need the WSDL of this WS. The WSDL needs to be in one file. Usually, when you build a WS with the NWDS, the NWDS generates 3 WSDL files. You have to merge the 3 WSDL files into one in order to be able to load it into the Adobe LiveCycle Designer. (Appendix 1 below gives a detailed explanation on how to merge the 3 WSDL files into one.)

To load the WSDL into the PDF form, go to the “Data View” tab in the Designer, right-click on “More”, and select “New Data Connection”. See the following screenshot:

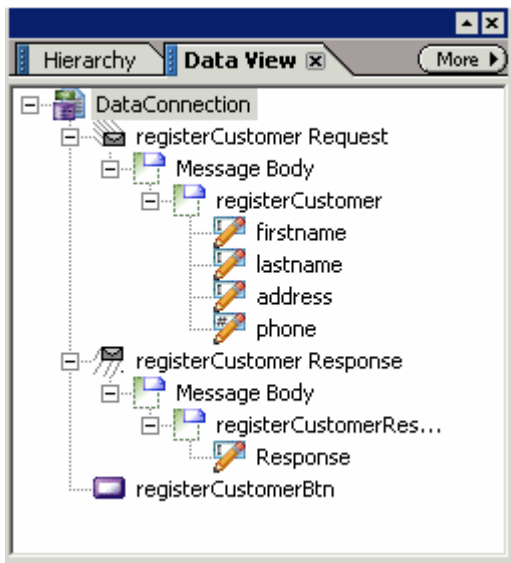


Then follow the wizard, select your WS WSDL file, and select your remote method invoked by the WS.

In our example, the WS will call a simple “registerCustomer” EJB remote method:



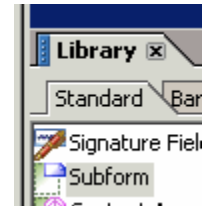
After you click on 'Finish', the “Data View” tab will display the parameters that are passed (Request) and returned (Response) by the WS. See below:



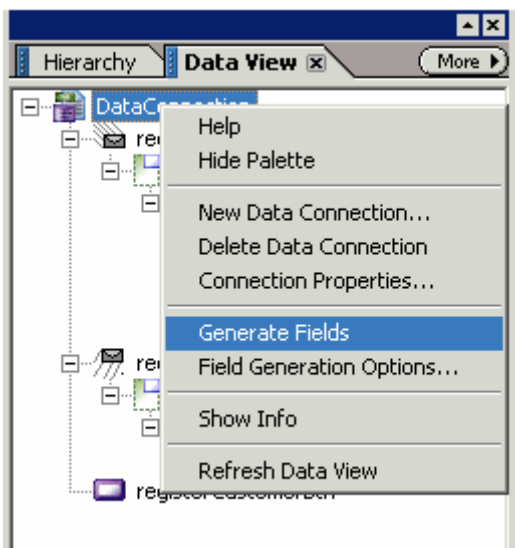
Auto-generate fields

Now that you have a data connection, you will need to create the UI that will be bound to the data. A nice Designer feature allows you to auto-generate these UI fields from the WSDL.

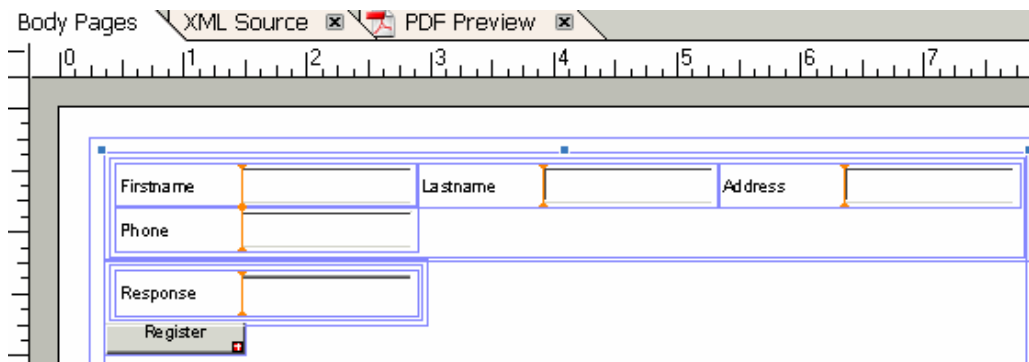
But first you will have to add a “Subform” UI component to your empty PDF sheet. Go to the Library tab and drag and drop the “subform” element to the “Body pages” view.



Then in the Data View tab, right-click, select “Generate Fields”. See screenshot below:



The result will be that a UI component will be created for each of the WS parameters. By default a Text field for data parameters and a button for the WS action has been created. See the screenshot below:

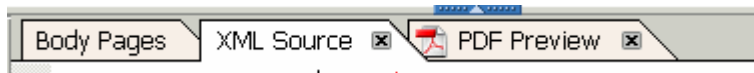


Redesign the UI

Of course the UI that has been auto-generated won't fit your UI requirements. You can redistribute, rename, reshape the UI elements or replace them by others. For example, you might want to replace the "Firstname" text field with a drop-down list. You will have to create a data binding connection between the drop-down UI component and the "firstname" WS parameter. There are different ways to do so, here is the way through some XML editing:

Drag & drop a "Drop-down List" UI element from the "Library" tab to the "Body Pages".

Select the UI text field that you want to replace, and switch to the "XML Source" view.



Your cursor will be located on the `<field name="firstname"...>` XML element. Below it, look for the `<connect../>` XML element.

```
<connect connection="DataConnection" usage="exportOnly"
ref="!connectionData.DataConnection.Body.registerCustomer.firstname"/>
```

Copy the above code to the "connect" element.

Switch back to the "Body Pages" view, Select your new "Drop-down List" UI, switch back to the "XML Source" view.

Paste the full `<connect....>` element just above the `</field>` element of the "Drop-down List" component. Now the selected value of this new drop-down list can be selected and sent by the WS.

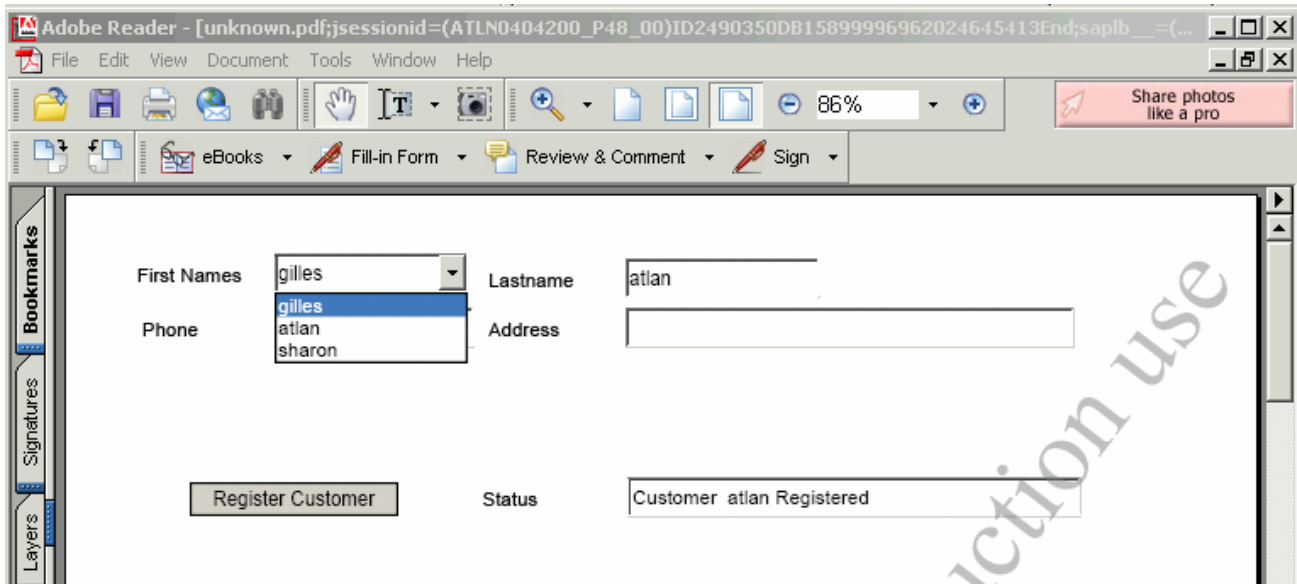
Now you can remove your "Firstname" Text Field or make it invisible.

Note: Another way is by going through the UI object properties and setting the data connection there.

Build, Run, Test

Once you have designed the form, you save, compile, deploy and run the Web Dynpro application. It will display the PDF form inside a Web Dynpro application. Then you can "Save as" the form locally as a PDF file. Double-click on this PDF file, Adobe Reader will open, and you can test your WS.

In our example after clicking on the Register button, the WS returns “Customer atlan Registered”, see the result below :



The screenshot shows the Adobe Reader interface with a web form. The form contains the following fields and values:

Field	Value
First Names	gilles
Lastname	atlan
Phone	atlan
Address	
Status	Customer atlan Registered

A "Register Customer" button is visible below the form fields. A large diagonal watermark reading "action use" is overlaid on the right side of the form.

Appendix 1: Merge WSDL files into one

Downloading the WSDLs:

Download the 3 WSDL files from the WAS using the WS-navigator tool:

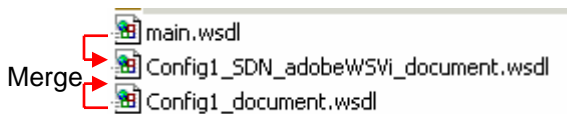
Go to <http://<was-host>:50000/wsnavigator/enterwsdl.html>

Select your WS

Click in the menu "WSDLs"

Download the default SAP WSDL

You will have 3 WSDL files in the downloaded zip files. In this example we will use the SDN_adobeWS Web Service that I created for this how-to, See below:



4 Steps for merging the files:

1. Copy/paste the `<wsdl:service>` XML element from the *main.wsdl* to the *Config1_SDN_adobeWSVi_document.wsdl*.
2. Copy the `<wsdl:binding>` XML element from the *Config1_document.wsdl* to the *Config1_SDN_adobeWSVi_document.wsdl*.
3. Add the SOAP namespace (`xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"`) from the *main.wsdl* to the *Config1_SDN_adobeWSVi_document.wsdl*.
4. Adjust the namespace reference by replacing "prt0" and "bns0" with "tns".

See below the merged WSDL file and the explanation of these 4 steps.

Add to the Config1_SDN_adobeWSVi_document.wsdl these xml elements :

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns0="urn:SDN_adobeWSVi"
targetNamespace="urn:SDN_adobeWSsd/SDN_adobeWSVi/document"
xmlns:tns="urn:SDN_adobeWSsd/SDN_adobeWSVi/document"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
<wsdl:types>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:SDN_adobeWSVi" xmlns:tns="urn:SDN_adobeWSVi"
elementFormDefault="qualified">
<xs:element name="registerCustomer">
<xs:complexType>
<xs:sequence>
<xs:element name="firstname" type="xs:string" nillable="true"/>
<xs:element name="lastname" type="xs:string" nillable="true"/>
<xs:element name="address" type="xs:string" nillable="true"/>
<xs:element name="phone" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="registerCustomerResponse">
<xs:complexType>
<xs:sequence>
<xs:element name="Response" type="xs:string" nillable="true"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
</wsdl:types>
<wsdl:message name="registerCustomerIn_doc">
<wsdl:part name="parameters" element="ns0:registerCustomer"/>
</wsdl:message>
<wsdl:message name="registerCustomerOut_doc">
<wsdl:part name="parameters" element="ns0:registerCustomerResponse"/>
</wsdl:message>
<wsdl:portType name="SDN_adobeWSVi_Document">
<wsdl:operation name="registerCustomer">
<wsdl:input message="tns:registerCustomerIn_doc"/>
<wsdl:output message="tns:registerCustomerOut_doc"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="Config1Binding" type="tns:SDN_adobeWSVi_Document">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<wsdl:operation name="registerCustomer">
<soap:operation soapAction="" />
<wsdl:input>
<soap:body use="literal" parts="parameters"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="SDN_adobeWS">
<wsdl:port name="Config1Port_Document" binding="tns:Config1Binding">
<soap:address
location="http://localhost:50000/SDN_adobeWS/Config1?style=document" />
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Step 3: Added from the Config1_document.wsdl

Step 4: Replace "prt0" and "bns0" with "tns"

Step 2: Added from the Config1_document.wsdl

Step 1: Added from the main.wsdl

Author Bio



Gilles Atlan is an SAP NetWeaver Solution Architect in the New Product Introduction (NPI) group. He mainly focuses on the J2EE technology in SAP NetWeaver.