

# Salesforce Integration Using PI: How to Perform Query and Other DML Operations Using the Enterprise WSDL



## Applies to:

SAP NetWeaver Process Integration 7.1

## Summary

A case study for integrating Salesforce.com for query and create webservice operation using PI 7.1

**Author:** Unnamalai Arunachalam

**Company:** Wipro Technologies

**Created on:** 8 July 2010

## Author Bio

Unnamalai Arunachalam is currently working in Wipro Technologies as SAP XI/PI consultant.

## Table of Contents

Introduction .....	3
Overview .....	3
Perform Query Operation on a SFDC Account Object .....	3
Download Enterprise WSDL from SFDC .....	3
Modify Query Object in the Enterprise WSDL .....	3
Repository Objects .....	6
Design Objects .....	11
Configure an Outbound Proxy or RFC Call .....	13
Perform Create operation of an SFDC Account Object .....	14
Download Enterprise WSDL from SFDC .....	14
Modify Create Object in the Enterprise WSDL .....	14
Repository Objects .....	15
Design Objects .....	21
Configure an Outbound Proxy or RFC Call .....	22
Related Content .....	23
Disclaimer and Liability Notice .....	24

## Introduction

In my current project we are using PI as the Enterprise wide service bus. Our client is evaluating the use of salesforce.com as the CRM tool. As part of this evaluation exercise, client wanted us to do a prototype interface to integrate their core ERP (SAP ECC) with Salesforce.com (SFDC).

I would like to thank Bhavesh Kantilal, Saravana Kuppusamy, Santosh Kumar Vellingiri, Harsh Chawla, Prasanna Vittal and Pragya Sharma for their valuable inputs and helping me in developing this prototype successfully.

## Overview

For this POC we developed a scenario to integrate the ECC system with salesforce.com via PI. We implemented two scenarios as part of this POC,

1. Perform a query operation on an SFDC Account object
2. Perform Create operation of an SFDC Account object.

We opted for the web services approach to perform the above POC. SFDC provides an enterprise WSDL, which can be downloaded and imported to Repository.

### Steps to be followed for performing a DML operation:

1. Download Enterprise WSDL from SFDC.
2. Modify the DML operation object in the Enterprise WSDL according to the webservice performed.
3. Create Repository objects.
4. Create Directory objects.
5. Configure either an outbound proxy or RFC call.

## Perform Query Operation on a SFDC Account Object

### Download Enterprise WSDL from SFDC

In the first place we need to have an account registered in salesforce.com (free developer account) to download the WSDL. Login in to SFDC with user credentials and download the WSDL from API. Also refer to this [Article](#) on “**Salesforce.com Integration Using SAP PI: A Case Study**” for the detail description on how to create an account in SFDC and call a simple webservice ‘getServerTimestamp’.

### Modify Query Object in the Enterprise WSDL

Query operation executes a query against the specified “Account” object and returns data as response. Account Data type has multiple elements or fields of which we can query ‘n’ number of fields according to our requirement.

Screenshot of Account Data type with elements from the enterprise WSDL.

```
- <complexType name="Account">
- <complexContent>
- <extension base="ens:sObject">
- <sequence>
  <element name="AccountContactRoles" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="AccountNumber" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="AccountPartnersFrom" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="AccountPartnersTo" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Active__c" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="ActivityHistories" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="AnnualRevenue" nillable="true" minOccurs="0" type="xsd:double" />
  <element name="Assets" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Attachments" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="BillingCity" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="BillingCountry" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="BillingPostalCode" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="BillingState" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="BillingStreet" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="Cases" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Contacts" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Contracts" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="CreatedBy" nillable="true" minOccurs="0" type="ens:User" />
  <element name="CreatedById" nillable="true" minOccurs="0" type="tns:ID" />
  <element name="CreatedDate" nillable="true" minOccurs="0" type="xsd:dateTime" />
  <element name="CustomerPriority__c" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="Description" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="Events" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Fax" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="Histories" nillable="true" minOccurs="0" type="tns:QueryResult" />
  <element name="Industry" nillable="true" minOccurs="0" type="xsd:string" />
  <element name="IsDeleted" nillable="true" minOccurs="0" type="xsd:boolean" />
  <element name="LastActivityDate" nillable="true" minOccurs="0" type="xsd:date" />
  <element name="LastModifiedBy" nillable="true" minOccurs="0" type="ens:User" />
  <element name="LastModifiedById" nillable="true" minOccurs="0" type="tns:ID" />

```

For example, we take elements NAME, OWNERID, WEBSITE, and PHONE from the ACCOUNT data type to perform a query operation in SFDC. The WSDL will have a query action and query response as shown below.

```
- <element name="query">
- <complexType>
- <sequence>
  <element name="queryString" type="xsd:string" />
</sequence>
</complexType>
</element>
- <element name="queryResponse">
- <complexType>
- <sequence>
  <element name="result" type="tns:QueryResult" />
</sequence>
</complexType>
</element>

```

The QueryResult is of data type 'sObject'. By default all the action would point to data type sObject in this WSDL. To query Account related details, we change the data type of QueryResult to "Account".

```
- <complexType name="QueryResult">
  - <sequence>
    <element name="done" type="xsd:boolean" />
    <element name="queryLocator" type="tns:QueryLocator" nillable="true" />
    <element name="records" type="ens:sObject" nillable="true" minOccurs="0" maxOccurs="unbounded" />
    <element name="size" type="xsd:int" />
  </sequence>
</complexType>
```

Open the WSDL in text editor and change the data type of QueryResult from "sObject" to "Account".

```
- <complexType name="QueryResult">
  - <sequence>
    <element name="done" type="xsd:boolean" />
    <element name="queryLocator" type="tns:QueryLocator" nillable="true" />
    <element name="records" type="ens:Account" nillable="true" minOccurs="0" maxOccurs="unbounded" />
    <element name="size" type="xsd:int" />
  </sequence>
</complexType>
```

## Repository Objects

Import the Query WSDL in to repository as an external definition ED\_QueryWSDL.

The screenshot shows the 'Display External Definition' window for 'ED\_QueryWSDL'. The status is 'Being Processed'. The Name is 'ED\_QueryWSDL'. The File is 'SFDCWSDL\_queryAccount.wsdl'. The Source is empty. The Category is 'wsdl' and 'Messages'. The search bar contains 'name="query"'. The main pane displays the XML schema for the query operation.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:element xmlns:ens="urn:subject.enterprise.soap.sforce.com" name="query">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="queryString" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element xmlns:ens="urn:subject.enterprise.soap.sforce.com" name="queryResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="result" type="QueryResult" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element xmlns:ens="urn:subject.enterprise.soap.sforce.com" name="queryAll">
  <xsd:complexType>
    <xsd:sequence>
```

Create outbound synchronous interface which contains QueryResponse and QueryRequest as input and output messages respectively, chose from the external definition ED\_QueryWSDL.

**OI\_Query: Edit Message Interface** Status: Active

Name: OI\_Query

Namespace: [REDACTED]

Software Component Version: [REDACTED]

Description: [REDACTED]

**Attributes**

Category:  Inbound  Outbound  Abstract

Mode:  Synchronous  Asynchronous

**Message Types**

Message Type	Type Name of External Message *	Namespace of External Message *
Output Message	queryRequest	urn:enterprise.soap.sforce.com
Input Message	queryResponse	urn:enterprise.soap.sforce.com
Fault Message Types	Type Name *	Namespace *

Similarly create an inbound synchronous interface with query request and query response as input and output messages respectively from the external definition ED\_QueryWSDL.

Create Message Mapping with the query string as input and also dynamically pass the server URL in a UDF.

Below is the Screenshot of the message mapping which shows that the query string is supplied as a constant to the target message.

The screenshot displays the 'MM\_QueryResponse: Edit Message Mapping' window. The 'Design' tab is active, showing a mapping between an external message 'queryRequest' and a target message 'queryRequest'. The 'Tree' view on the left shows the source message structure with 'query' (1..1 occurrences) containing 'queryString' (1..1 occurrences, type 'xsd:string'). The 'Tree' view on the right shows the target message structure with 'query' (1..1 occurrences) containing 'queryString' (1..1 occurrences, type 'xsd:string'). A mapping arrow connects the source 'queryString' to the target 'queryString'. The 'Functions' list at the bottom includes 'Text', 'substring', 'concat', 'equalsS', 'indexOf', 'lastIndexOf', 'compare', and 'replaceSt'.

Tree	Occurrences	Type	Details	Description
[ ] query	1..1			
[ ] queryString	1..1	xsd:string		

Constant [Select Name, ownerId, website, Phone from Account]



We see that the server URL remains the same for all the operations performed in the WSDL. So we decided to go with the UDF, to set the Server URL dynamically at the header using TServerLocation parameter.

```
public String setTargetURL(Container container){
    DynamicConfiguration conf = (DynamicConfiguration)
    container.getTransformationParameters().get(StreamTransformationConstants.DYNAMIC_CONFIGURATION);

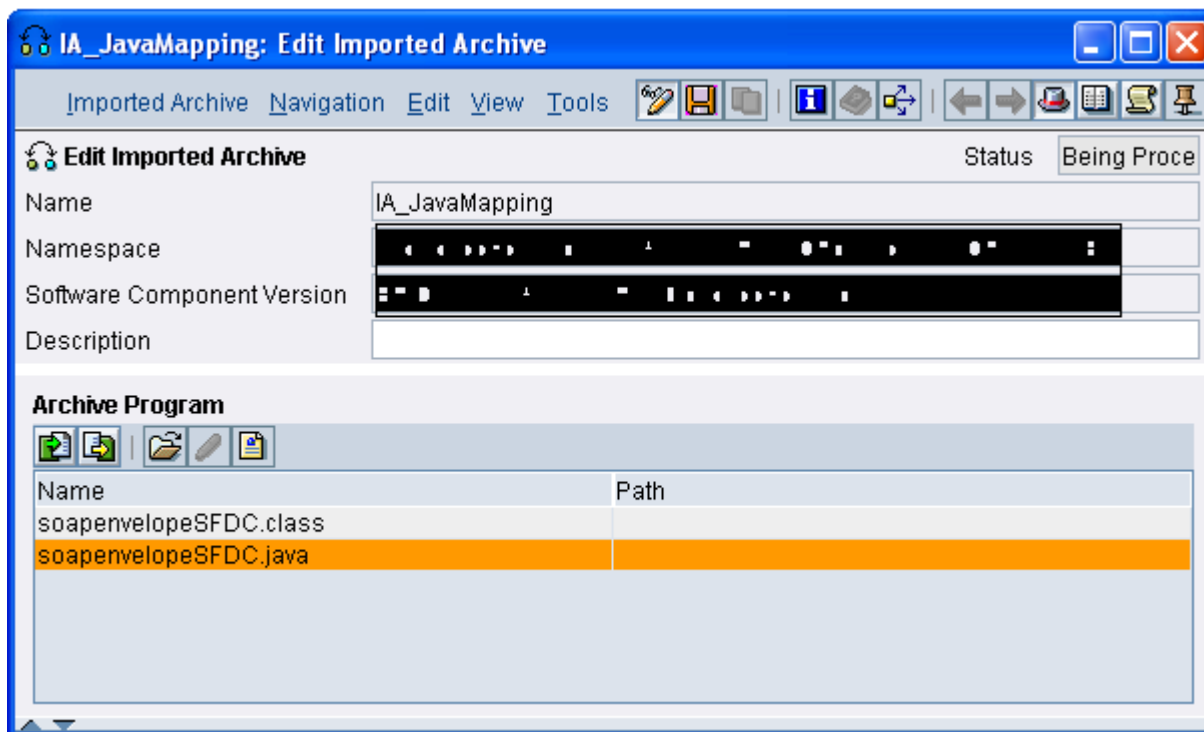
    DynamicConfigurationKey key2 =
    DynamicConfigurationKey.create("http://sap.com/xi/XI/System/SOAP", "TServerLocation");

    conf.put(key2, "https://ap1-
    api.salesforce.com/services/Soap/c/18.0/00D900000000ISZc");

    return " ";
}
```

Import the Java mapping. This Java Code will do a Soap lookup to get the session ID. Also build the soap envelope manually because the query webservice expects few other attributes to be set explicitly. Please refer this [Java code](#) for the SOAP Lookup and to build the Envelope.

Screenshot of Java mapping imported in to repository:



Create an operation mapping with both message mapping and java mapping in subsequent order in the mapping program, as shown below,

**IM\_QueryResponse: Display Interface Mapping**

Interface Mapping Navigation Edit View

Status Active

Name IM\_QueryResponse

Namespace

Software Component Version

Description

Design Test

**Source Interface \***

Name	Namesp...	Software...	Occurre...
OL_Query			1

**Target Interface \***

Name	Namesp...	Software...	Occurre...
IL_QueryRes			1

Read Interfaces

Request Response

**Source Message**

queryRequest

**Mapping Program**

Type	Name	Namespace
Message Mapping	MM_QueryResponse	
Java Class	SoapEnvelopeForSFDC	

**Target Message**

queryReques

## Design Objects

Create SOAP receiver communication channel for query response.

The screenshot displays the configuration for a communication channel in SAP NetWeaver Administrator. The window title is "CC\_QueryRes: Display Communication Channel". The status is "Active".

**Display Communication Channel**

Communication Channel: CC\_QueryRes  
 Party:   
 Service: BS\_Sample  
 Description:   
 Status: Active

**Parameters** | Identifiers | Module

Adapter Type \* SOAP | http://sap.com/xi/XI/System | SAP BASIS 6.40  
 Sender |  Receiver  
 Transport Protocol \* HTTP  
 Message Protocol \* SOAP 1.1  
 Adapter Engine \* Integration Server

**Connection Parameters**

Target URL \* https://www.salesforce.com/services/Soap/c/18.0  
 Configure User Authentication  
 Configure Certificate Authentication  
 Configure Proxy

**Adapter-Specific Message Attributes**

Use Adapter-Specific Message Attributes  
 Variable Transport Binding  
 Variable Header (XHeaderName1)   
 Variable Header (XHeaderName2)   
 Do Not Use SOAP Envelope  
 Keep Headers  
 Keep Attachments  
 Use Encoded Headers  
 Use Query String  
 SOAP Action queryResponse

Please check the option “Do not use SOAP Envelope” in the receiver communication Channel for query response.

Provide the SOAP Action as “queryResponse” in the Conversion parameters of the Communication channel.

**Conversion Parameters**

Do Not Use SOAP Envelope  
 Keep Headers  
 Keep Attachments  
 Use Encoded Headers  
 Use Query String  
 SOAP Action queryResponse

The incoming SOAP Message payload will be of content type “application/xml” this is due to that we checked the option “Do Not use SOAP Envelope”. In order to maintain the content type as “Text/xml” we use the adapter module “MessageTransformBean”.

The screenshot displays the 'Display Communication Channel' configuration window for 'CC\_QueryRes'. The status is 'Active'. The configuration includes the following details:

- Communication Channel: CC\_QueryRes
- Party: (empty)
- Service: BS\_Sample
- Description: (empty)

The 'Module' tab is selected, showing the 'Processing Sequence' and 'Module Configuration' sections.

**Processing Sequence**

Number	Module Name	Type	Module Key
1	localejbs/AF_Modules/MessageTransfo...	Local Enterprise Bean	key
2	sap.com/com.sap.aii.af.soapadapter/XI...	Local Enterprise Bean	soap

**Module Configuration**

Module Key	Parameter Name	Parameter Value
key	Transform.ContentType	text/xml

Create Receiver determination and Interface determination.

Create Receiver agreement.

## Configure an Outbound Proxy or RFC Call

We can either configure a proxy or RFC to trigger the scenario. In our case we configured an outbound proxy to trigger. Once the Proxy is triggered the soap envelope and Header is added to the Query Request. We can also test the scenario from Runtime Workbench tool.

Screen shot from SXMB\_MONI Request message. The query Request message is embedded with the SOAP Envelope and Header, which is manually built in the Java Mapping.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com">
- <soapenv:Header>
- <urn:SessionHeader>
  <urn:sessionId>00D90000000ISZc!
  ARUAQDA_mbl7Bh3v8LxL9qSCM04MFq0SgR2zZQmo5yvjv4nrHqc7bBX7PANEopzB0KSRBBS0A2yxSWGvPVx8xOtp0eSqn_VsM</urn:sessionId>
</urn:SessionHeader>
</soapenv:Header>
- <soapenv:Body>
- <ns0:query xmlns:ns0="urn:enterprise.soap.sforce.com">
  <ns0:queryString>Select Name, ownerId, website, Phone from Account</ns0:queryString>
  </ns0:query>
</soapenv:Body>
</soapenv:Envelope>
```

Query Response from SFDC which contains the details of the various accounts. Screenshot from SXMB\_MONI Response message.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns="urn:enterprise.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:sf="urn:subject.enterprise.soap.sforce.com">
- <soapenv:Body>
- <queryResponse>
- <result>
  <done>true</done>
  <queryLocator xsi:nil="true" />
- <records xsi:type="sf:Account">
  <sf:Id xsi:nil="true" />
  <sf:Name>ABC_SFDCTest</sf:Name>
  <sf:OwnerId>00590000000h8N6AAI</sf:OwnerId>
  <sf:Phone>1234567890</sf:Phone>
</records>
- <records xsi:type="sf:Account">
  <sf:Id xsi:nil="true" />
  <sf:Name>ABC_SFDCTest</sf:Name>
  <sf:OwnerId>00590000000h8N6AAI</sf:OwnerId>
  <sf:Phone>1234567890</sf:Phone>
</records>
- <records xsi:type="sf:Account">
  <sf:Id xsi:nil="true" />
  <sf:Name>ABC_SFDCTest</sf:Name>
  <sf:OwnerId>00590000000h8N6AAI</sf:OwnerId>
  <sf:Phone>1234567890</sf:Phone>
</records>
- <records xsi:type="sf:Account">
  <sf:Id xsi:nil="true" />
  <sf:Name>SFDCTest_04</sf:Name>
  <sf:OwnerId>00590000000h8N6AAI</sf:OwnerId>
  <sf:Phone>9731217520</sf:Phone>
```

## Perform Create operation of an SFDC Account Object

### Download Enterprise WSDL from SFDC

We need to have an account registered in salesforce.com (free developer account) to download the WSDL. Login in to SFDC with user credentials and download the WSDL from API. Also refer to this [Article](#) on “Salesforce.com Integration Using SAP PI: A Case Study” for the detail description on how to create an account in SFDC and download the WSDL.

### Modify Create Object in the Enterprise WSDL

The WSDL will have a create operation and the corresponding create response as a webservice. The Create object is pointing to data type “sObject”. Now our requirement is to create an Account object. So we change the data type of create operation.

```
- <element name="create">
  - <complexType>
    - <sequence>
      <element name="sObjects" type="ens:sObject" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
- <element name="createResponse">
  - <complexType>
    - <sequence>
      <element name="result" type="tns:SaveResult" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

Open the WSDL in the Text editor and change the data type to “Account”.

```
- <element name="create">
  - <complexType>
    - <sequence>
      <element name="sObjects" type="ens:Account" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
- <element name="createResponse">
  - <complexType>
    - <sequence>
      <element name="result" type="tns:SaveResult" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
</element>
```

## Repository Objects

Import the Create WSDL in to repository as external definition ED\_Create.

**ED\_Create: Display External Definition**

External Definition    Navigation    Edit    View    Tools

Save    Status: Active

Name: ED\_Create

Namespace: [REDACTED]

Software Component Version: [REDACTED]

Description: [REDACTED]

Category: wsdl    Messages: From All Available Message Definitions

File \*: SFDCWSDL\_Create.wsdl

Source: [REDACTED]

Imported Document    Messages    **WSDL**    External References

Search: [REDACTED]

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="result" type="DescribeTabSetResult" minOccurs="0" maxOccurs="unbounded" nillable="true" />
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element xmlns:ens="urn:subject.enterprise.soap.sforce.com" name="create">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="sObjects" type="ens:Account" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element xmlns:ens="urn:subject.enterprise.soap.sforce.com" name="createResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="result" type="SaveResult" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

```

Create outbound synchronous interface which contains CreateRequest and CreateResponse as output and input messages respectively, chose from the WSDL ED\_Create.

The screenshot shows the configuration interface for a Message Interface. The title bar reads "OI\_CreateReq: Display Message Interface". The main window has a menu bar with "Message Interface", "Navigation", "Edit", "View", and "Tools". Below the menu bar, the interface is titled "Display Message Interface" with a status of "Active".

Fields for configuration include:

- Name: OI\_CreateReq
- Namespace: [Redacted]
- Software Component Version: [Redacted]
- Description: [Redacted]

Below these fields are tabs for "Definition", "Context Objects", and "WSDL". The "Definition" tab is active and shows the following settings:

**Attributes**

- Category:  Inbound,  Outbound,  Abstract
- Mode:  Synchronous,  Asynchronous

**Message Types**

Message Type	Type Name of External Message *	Namespace of External Message *
Output Message	createRequest	urn:enterprise.soap.sforce.com
Input Message	createResponse	urn:enterprise.soap.sforce.com
Fault Message Types	[Empty]	[Empty]



Similarly create an inbound synchronous interface with query request and query response as input and output messages respectively from the external definition ED\_Create.

The screenshot shows the 'Display Message Interface' configuration window for 'II\_CreateRes'. The window title is 'II\_CreateRes: Display Message Interface'. The status is 'Active'. The configuration is as follows:

Field	Value
Name	II_CreateRes
Namespace	
Software Component Version	
Description	

The 'Definition' tab is selected. Under 'Attributes', the 'Inbound' radio button is selected, and the 'Synchronous' radio button is selected. Under 'Message Types', the configuration is as follows:

Message Type	Type Name of External Message *	Namespace of External Message *
Input Message	createRequest	urn:enterprise.soap.sforce.com
Output Message	createResponse	urn:enterprise.soap.sforce.com
Fault Message Types		

A message mapping which contains the entire account details. The server URL is set dynamically at the Header as it remains constant for all the operation. Create an UDF and set the Server URL to the parameter TServerLocation.

The screenshot displays the 'MM\_Create: Edit Message Mapping' window. The 'Design' tab is active, showing the message mapping structure. The 'Tree' view on the left shows the source message structure, and the 'Tree' view on the right shows the target message structure. The 'AccountNumber' element in the target tree is highlighted. The mapping diagram shows a constant value '02456460' being mapped to the 'ns1:Accou...' element. The 'Functions' list at the bottom includes 'Text', 'substring', 'concat', 'equalsS', 'indexOf', 'lastIndexOf', 'compare', and 'replaceSI'.

Import the Java mapping. Java Code will do a Soap lookup to get the session ID and build the soap envelope manually because the create webservice expects few other attributes to be set explicitly. Please refer the link for [Java code](#).

Create an operation mapping with both message mapping and java mapping in subsequent order in mapping program. Find below screenshot,

The screenshot shows the SAP PI Interface Mapping tool for the operation **IM\_create**. The tool is in the **Design** tab. The **Source Interface** is **OI\_CreateReq** and the **Target Interface** is **II\_CreateRes**. The **Mapping Program** is configured with the following settings:

Type	Name	Namespace
Message Mapping	MM_Create	
Java Class	SoapUpdateMap	

The **Source Message** is **createRequest** and the **Target Message** is **createRequest**. The tool also shows the **Request** and **Response** tabs, and the **Read Interfaces** button.

## Changes in the Java Mapping

- Certain changes required in Java code in order to create an account. The namespace **"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"** is added along with the existing soap envelope. This indicates that elements and the data types used belong to the above namespace.
- The sObjects in the Soap body is modified. This is because the sObjects will have only two elements 'fieldstonull' and 'Id'. To overwrite sObjects with the fields of account we specify the data type as "Account" in the Soap body along with namespace.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:enterprise.soap.sforce.com"
xmlns:urn1="urn:object.enterprise.soap.sforce.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <urn:SessionHeader>

<urn:sessionId>00D9000000015Zc!ARUAQEpopIFBeUMbP_WFLItpkE9I3Gy4m.lxvuZmjKXj2PKIzb.Y
7W0Xrza2pw0o72bFtXT0l6zDXn1RH0Be5Umcb055av0</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:create>

      <urn:sObjects xsi:type="Account" xmlns="urn:enterprise.soap.sforce.com">
        <Name>ABC_TEST</Name>
      </urn:sObjects>
    </urn:create>
  </soapenv:Body>
</soapenv:Envelope>

```

## Design Objects

Create SOAP receiver communication channel for create response.

The screenshot shows the configuration for a communication channel named 'CC\_CreateRes'. The status is 'Active'. The communication channel is 'CC\_CreateRes', the party is empty, the service is 'BS\_Sample', and the description is empty. The 'Parameters' tab is selected, showing the following settings:

- Adapter Type: SOAP
- URL: http://sap.com/xi/XI/System
- System: SAP BASIS 6.40
- Sender:  Sender, Receiver:  Receiver
- Transport Protocol: HTTP
- Message Protocol: SOAP 1.1
- Adapter Engine: Integration Server

The 'Connection Parameters' section includes:

- Target URL: https://www.salesforce.com/services/Soap/c/18.0
- Configure User Authentication
- Configure Certificate Authentication
- Configure Proxy

The 'Adapter-Specific Message Attributes' section includes:

- Use Adapter-Specific Message Attributes
- Variable Transport Binding
- Variable Header (XHeaderName1):
- Variable Header (XHeaderName2):

Please check the option “Do not use SOAP Envelope” in the receiver communication Channel for query response.

Provide the SOAP Action as “create” in the conversion parameters of the Communication channel.

### Conversion Parameters

The 'Conversion Parameters' section includes the following settings:

- Do Not Use SOAP Envelope
- Keep Headers
- Keep Attachments
- Use Encoded Headers
- Use Query String
- SOAP Action: create

The incoming SOAP Message payload will be of content type “application/xml” this is due to that we checked the option “Do Not use SOAP Envelope”. In order to maintain the content type as “Text/xml” we use the adapter module “MessageTransformBean”.

## Configure an Outbound Proxy or RFC Call

We can either configure a proxy or RFC to trigger the scenario. In our case we configured an outbound proxy to trigger. Once the Proxy is triggered the soap envelope and Header is added to the Create Request

Screen shot from SXMB\_MONI Request message. Create Request message payload with the SOAP Envelope and header which is manually built in the Java mapping code.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com"
  xmlns:urn1="urn:subject.enterprise.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <soapenv:Header>
- <urn:SessionHeader>
  <urn:sessionId>00D900000000ISZc!
  ARUAQKauHpFnIGwe3WYB9mkwUjKQBcZ9hpYFh0A5tDT0vAl7zN7wuYHLiNqI82NxJgLOkPXHqQHm17qELigItU0iIS6Wu76X</urn:sessionId>
  </urn:SessionHeader>
</soapenv:Header>
- <soapenv:Body>
- <ns0:create xmlns:ns0="urn:enterprise.soap.sforce.com">
- <ns0:sObjects xsi:type="Account" xmlns="urn:enterprise.soap.sforce.com">
  <ns1:AccountNumber xmlns:ns1="urn:subject.enterprise.soap.sforce.com">02456460</ns1:AccountNumber>
  <ns1:BillingCountry xmlns:ns1="urn:subject.enterprise.soap.sforce.com">India</ns1:BillingCountry>
  <ns1:Name xmlns:ns1="urn:subject.enterprise.soap.sforce.com">ABC_SFDCTest</ns1:Name>
  <ns1:Phone xmlns:ns1="urn:subject.enterprise.soap.sforce.com">1234567890</ns1:Phone>
  </ns0:sObjects>
</ns0:create>
</soapenv:Body>
</soapenv:Envelope>
```

The response would return a Boolean value, true or false. Along with the Boolean value it would also return an ID for that particular account created in SFDC.

Screenshot from SXMB\_MONI Response message indicating that Account is Created in SFDC successfully.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns="urn:enterprise.soap.sforce.com">
- <soapenv:Body>
- <createResponse>
- <result>
  <id>001900000003f9rWAAQ</id>
  <success>true</success>
  </result>
</createResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Hope this would help you all in integrating salesforce.com with ECC through PI to perform a query and create DML operation.

## Related Content

Article by Prasanna:

<http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/50a76cfa-4966-2d10-aba7-da496d9b5bcf>

Wiki page to Add SOAP envelope and SOAP Lookup:

<http://wiki.sdn.sap.com/wiki/display/stage/SFDC+Integration+using+PI+7.1+-+How+to+add+SOAP+Envelope+in+Java+Mapping>

Call a webservice from UDF by Bhavesh

<http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/5001>

## **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.