November 2007

Utilizing State Oriented Communication for Web Services Based on Business APIs (BAPI) with Microsoft Visual Studio 2005

Summary

Some applications require stateful oriented communication. If these function modules are published as Web services state oriented communication is supported for the server side using HTTP sessions that can be used by .NET based Web services clients to achieve consistency.
Instead of stateful communication SAP recommends to use SAP Enterprise Services because they are using a stateless model and ensure consistency of the called backend application.
This scenario however is applicable in a situation where no SAP Enterprise Services can be used and where for example an existing stateful RFC communication is to be replaced by a Web service based communication calling the same function modules.

Applies to

- SAP NetWeaver Application Server ABAP 6.40 SP21 or higher[1]
- Microsoft Visual Studio 2005

Contact

For feedback or questions you can contact the Collaboration Technology Support Center via the .NET Technologies forum in the .NET interoperability area of SDN. Please check the .NET interoperability area in SDN for any updates or further information.

Authors Bio

André Fischer works at **SAP AG** in the Strategic Alliance Microsoft Team. He is also a member of the CTSC (Collaboration Technology Support Center) that addresses various kinds of interoperability topics regarding SAP and Microsoft solutions. André has specialized in single sign-on, SAP Microsoft Active Directory integration, SAP Exchange Infrastructure BizTalk integration and knowledge management Microsoft Windows integration.

Jürgen Daiberl works at **Mircosoft Corp.** as Technical Evangelist at the Developer & Platform Evangelist Team in Redmond. Prior to his role in Redmond he worked at the CTSC (Collaboration Technical Support Center) in Walldorf / Germany, a joint staffed team between Microsoft and SAP. In his current role he is responsible for the Interoperability between Microsoft .NET and SAP NetWeaver on the Application level.

[1] Please see SAP Note 1050075 - Adaptation of the cookie setting for stateful communication

Contents

Introduction

In some cases it is necessary having a session / state oriented communication between applications. This can be because of aggregation of data for later or for handling multiple transactions in one Web Service call. An example could be the external BAPI COMMIT required by some BAPIs in contrast to SAP Enterprise Services that are using a stateless programming model.

SAP Enterprise Services using stateless communication

SAP Enterprise Services are based on a harmonized object model and are subject to strict governance. Every SAP Enterprise Service (in contrast to a RFC or a Web service based on a RFC) is a atomic transaction that leaves the database of the business application in a consistent state. Therefore it is not the responsibility of the consumer to ensure data consistency on the backend called.
This is not the case if a stateful model is used. Here it is the responsibility of the consumer to make sure that the calls to the backend are made in a consistent way. In principal the developer is even able to perform several calls to different BAPIs and RFCs thereby creating a LUW that spans the complete backend or even several backend systems. The developer therefore has to know the programming model used in the backend system(s) which makes an implementation more complicated and error prone.

Therefore SAP has decided to go for a stateless programming model in its eSOA-strategy. If SAP Enterprise Services are used the burden of responsibility to ensure data consistency is not left to the developer any more even if the Enterpise Service iself calls different systems in the backend.

In the long term SAP therefore *strongly recommends* only to use stateless Enterprise Services as interfaces to communicate with SAP NetWeaver rather than calling simple web services based on BAPIs.

Stateful BAPI calls using Web Services as an intermediate solution

If however it is necessary to deal with existing BAPI/RFC interfaces that are published as Web services this Collaboration Brief will show how to setup a session / state oriented communication with the SAP system. For demonstration purposes the Remote Function Module SRT_TESTS_FB_SUM is chosen. In addition you can find an example how to use session / state oriented communication in order to handle external BAPI COMMITs.

Remote Function Module using stateful communication

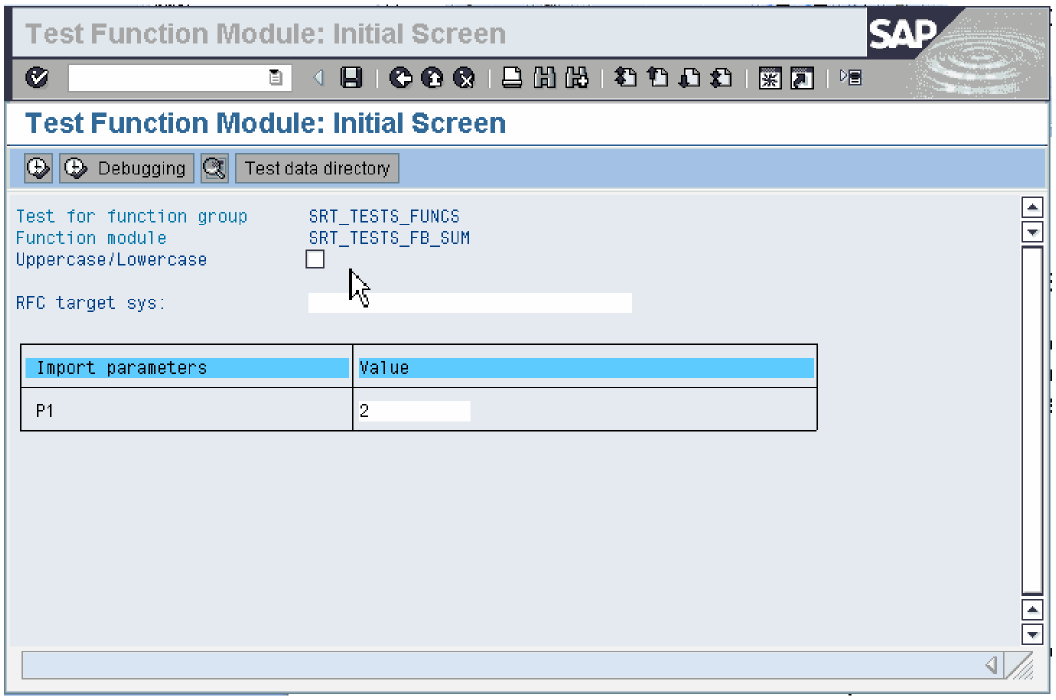
The Remote Function Module SRT_TESTS_FB_SUM accepts an input parameter P1 of type I and will return parameter RESULT of type I that contains the sum of P1 and the value of the global variable GL_SUM. The value of GL_SUM is initially zero.
If you call this RFC again from the same session, the variable GL_SUM will remember the entry from the previous call and will add the value of P1 to the value of RESULT that

has been returned the last time. Technically, this is achieved by using a global variable GL_SUM for the storage.

How-To Section

Testing the function module

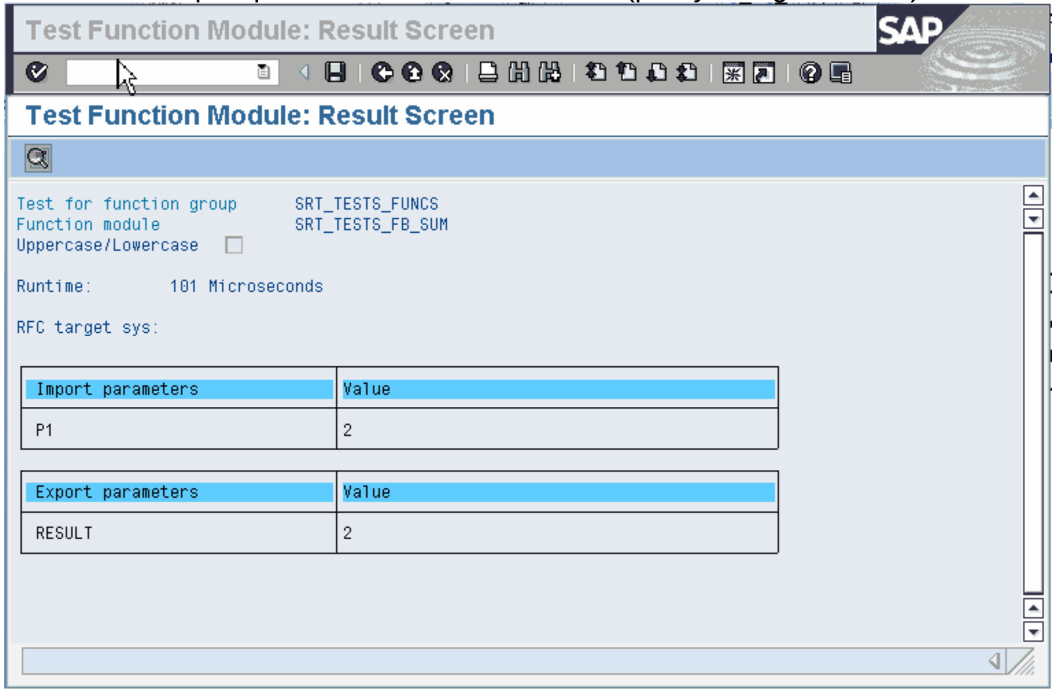
1. Test the Function Module (F8) and enter an integer value for the P1 parameter. The number 2 is used in the example.


Test Function Module: Initial Screen
Test Function Module: Initial Screen
Debugging
Test data directory

Test for function group	SRT_TESTS_FUNCS
Function module	SRT_TESTS_FB_SUM
Uppercase/Lowercase	☐
RFC target sys:	

Import parameters	Value
P1	2

2. You get an export parameter – RESULT that contains the sum of P1 and the value of the export parameter from the last call. (pretty straightforward).



Test Function Module: Result Screen

Test for function group	SRT_TESTS_FUNCS
Function module	SRT_TESTS_FB_SUM
Uppercase/Lowercase	☐

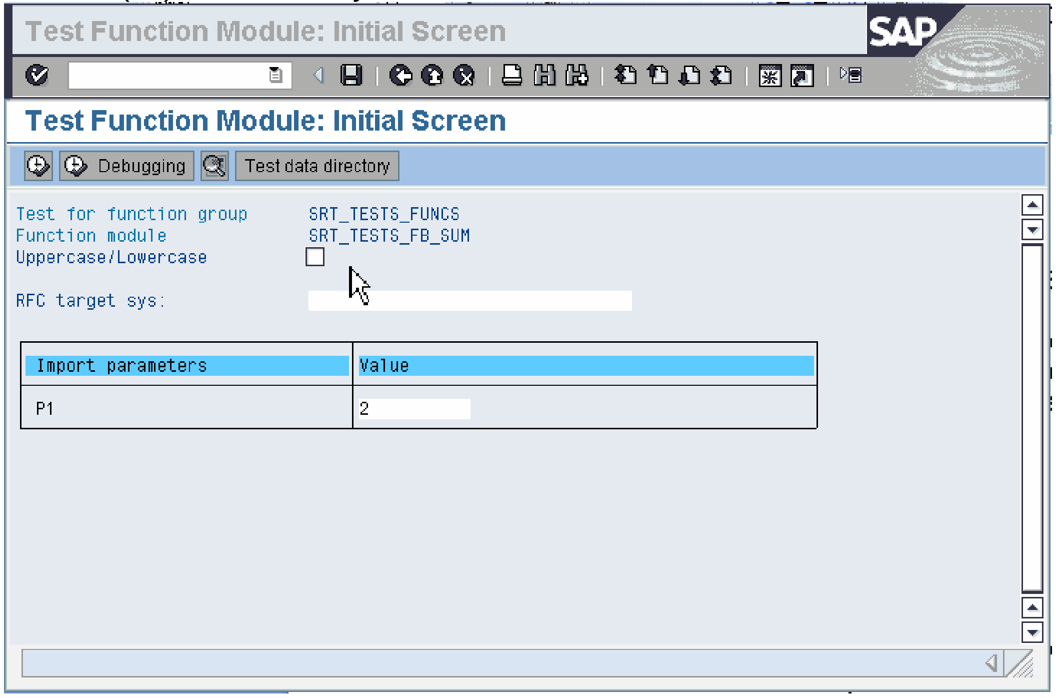
Runtime: 101 Microseconds

RFC target sys:

Import parameters	Value
P1	2

Export parameters	Value
RESULT	2

3. Hit Back (F3) twice to take you back to the Test Function Module: Initial Screen.

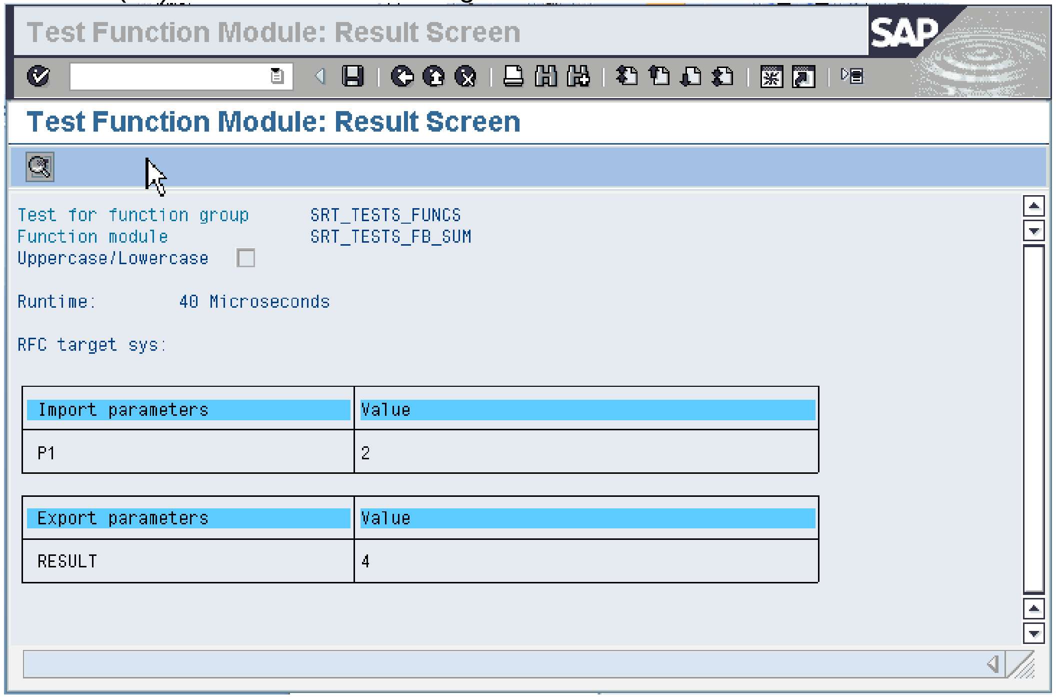


Test Function Module: Initial Screen

Test for function group	SRT_TESTS_FUNCS
Function module	SRT_TESTS_FB_SUM
Uppercase/Lowercase	☐
RFC target sys:	

Import parameters	Value
P1	2

4. Execute (F8) the function module again.



Test Function Module: Result Screen

```
Test for function group      SRT_TESTS_FUNCS
Function module              SRT_TESTS_FB_SUM
Uppercase/Lowercase    ☐

Runtime:         40 Microseconds

RFC target sys:
```

Import parameters	Value
P1	2

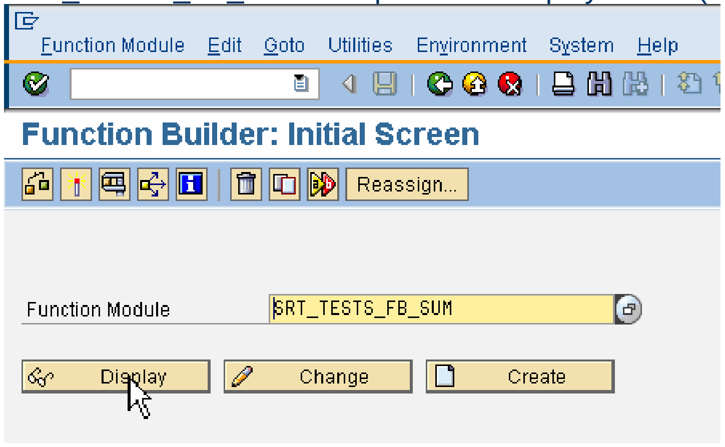
Export parameters	Value
RESULT	4

5. You should now the value 4 as a result. The reason is that we are using a global variable of this function group in this function module that will grow by P1.

If you leave the test function screens (F3) the value of global variable entries are then gone and you will start again with the initial value 0.

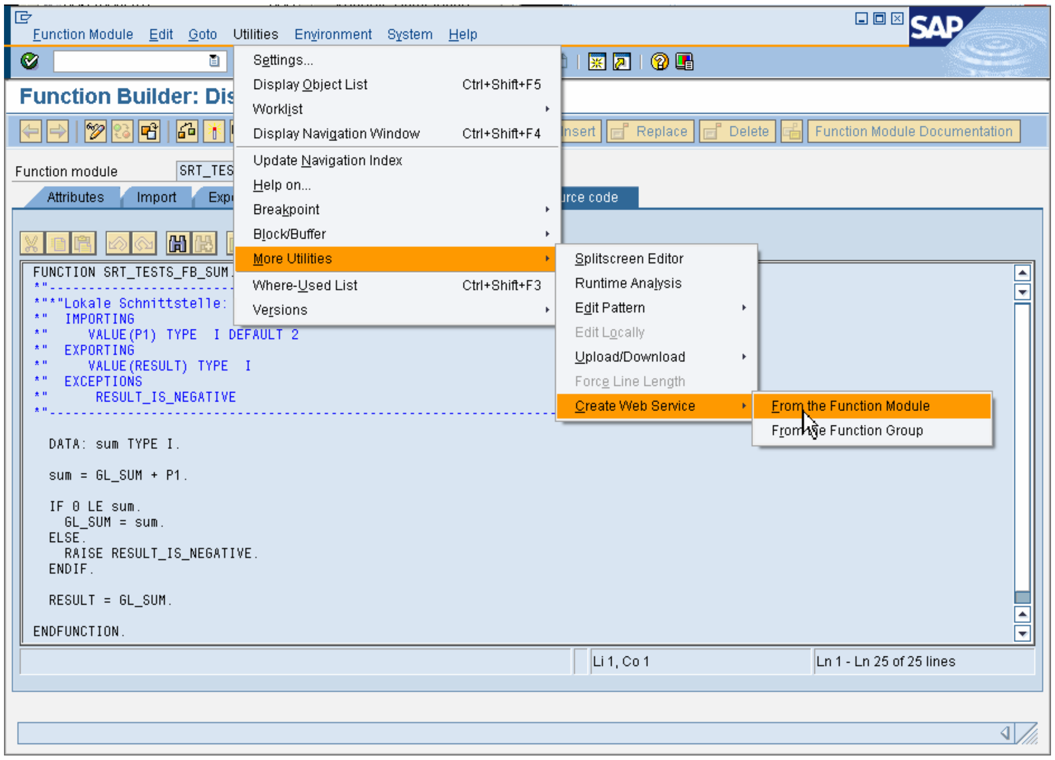
Create a web service from the function module SRT_TESTS_FB_SUM using the Web Service Creation Wizard

1. Start transaction SE37 and enter the name of the function module SRT_TESTS_FB_SUM and press the Display button (F7).



Function Module SRT_TESTS_FB_SUM

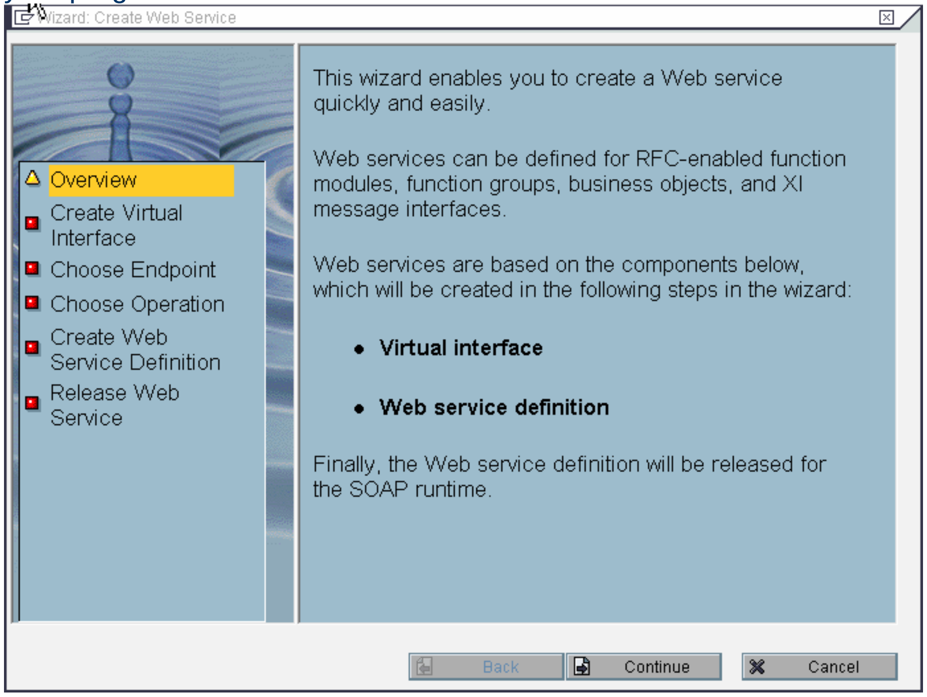
2. Begin the Web service creation by choosing Utilities -> More Utilities -> Create Web Service -> From the Function Module



Function module SRT_TES

```
FUNCTION SRT_TESTS_FB_SUM.
*"----------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(P1) TYPE  I DEFAULT 2
*"  EXPORTING
*"     VALUE(RESULT) TYPE  I
*"  EXCEPTIONS
*"      RESULT_IS_NEGATIVE
*"----------------------------------------------------

  DATA: sum TYPE I.

  sum = GL_SUM + P1.

  IF 0 LE sum.
    GL_SUM = sum.
  ELSE.
    RAISE RESULT_IS_NEGATIVE.
  ENDIF.

  RESULT = GL_SUM.

ENDFUNCTION.
```

3. The Web Service Wizard screen appears. The list on the left-hand side shows your progress in the wizard. Choose ‘Continue’.



This wizard enables you to create a Web service quickly and easily.

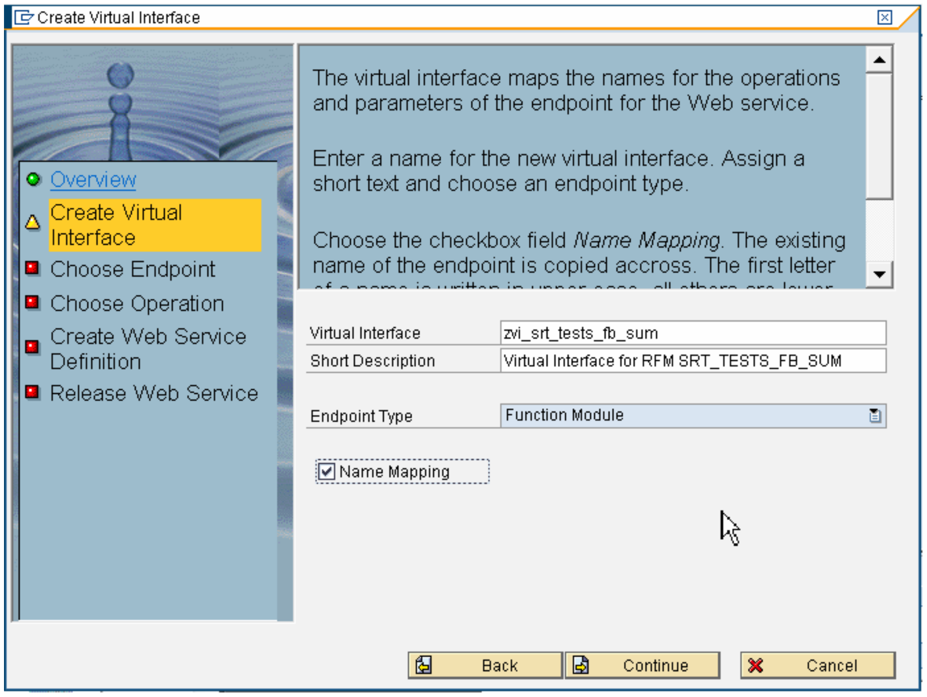
Web services can be defined for RFC-enabled function modules, function groups, business objects, and XI message interfaces.

Web services are based on the components below, which will be created in the following steps in the wizard:

- **Virtual interface**
- **Web service definition**

Finally, the Web service definition will be released for the SOAP runtime.

4. Enter ‘zvi_srt_tests_fb_sum’ as name for the Virtual Interface (VI). Enter a short description, such as ‘Virtual Interface for RFM SRT_TESTS_FB_SUM’. Check the checkbox ‘Name Mapping’. The ‘Endpoint Type’ is ‘Function Module’ and choose ‘Continue’.



The virtual interface maps the names for the operations and parameters of the endpoint for the Web service.

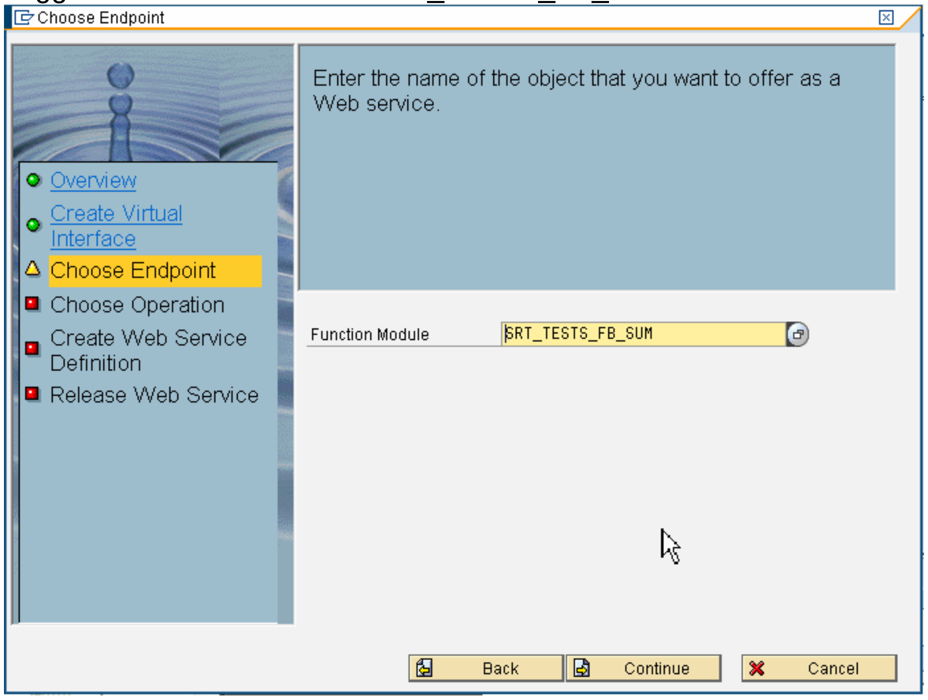
Enter a name for the new virtual interface. Assign a short text and choose an endpoint type.

Choose the checkbox field *Name Mapping*. The existing name of the endpoint is copied accross. The first letter

Virtual Interface	zvi_srt_tests_fb_sum
Short Description	Virtual Interface for RFM SRT_TESTS_FB_SUM
Endpoint Type	Function Module

☑ Name Mapping

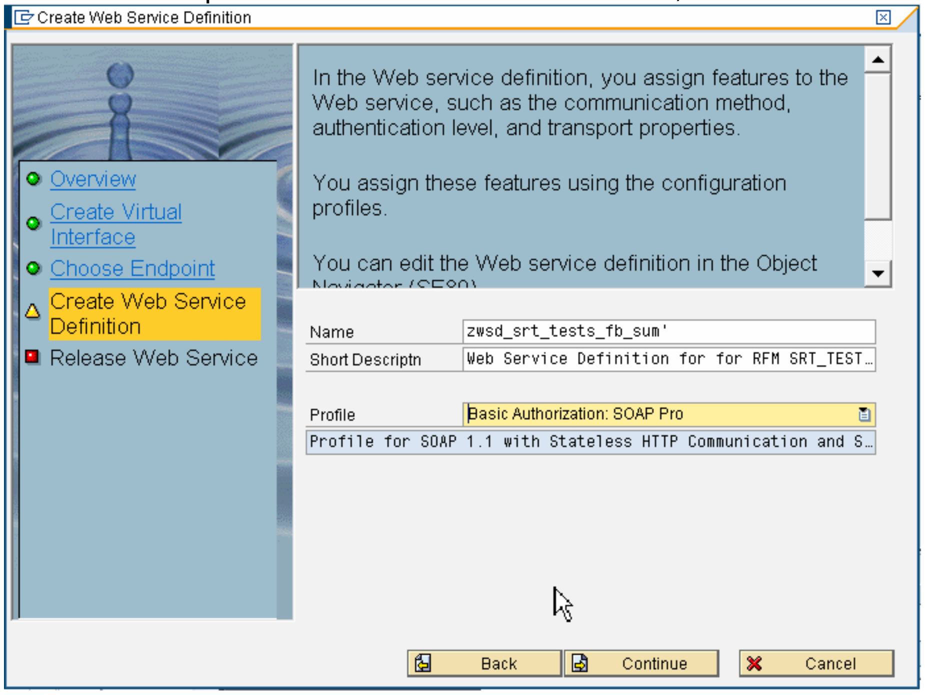
5. On the next screen you specify the object that is to be exposed as a Web service in accordance with the endpoint type that you selected before. Accept the suggested function module ‘SRT_TESTS_FB_SUM’ and choose ‘Continue’



Enter the name of the object that you want to offer as a Web service.

Function Module SRT_TESTS_FB_SUM

6. Now create the Web Service Definition that references the VI you just created. Enter ‘zwsd_srt_tests_fb_sum’ as name of the definition and a short description such as ‘Web Service Definition for for RFM SRT_TESTS_FB_SUM’. From the list of available profiles choose ‘Basic Authorization’, and then choose ‘Continue’.



In the Web service definition, you assign features to the Web service, such as the communication method, authentication level, and transport properties.

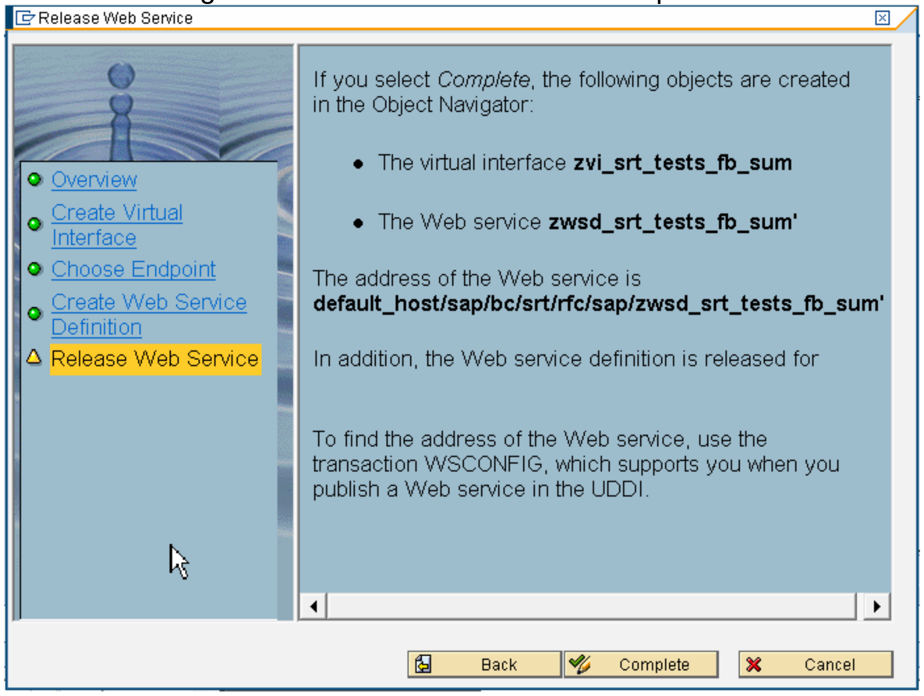
You assign these features using the configuration profiles.

You can edit the Web service definition in the Object

Name zwsd_srt_tests_fb_sum'
Short Descriptn Web Service Definition for for RFM SRT_TEST…

Profile Basic Authorization: SOAP Pro
Profile for SOAP 1.1 with Stateless HTTP Communication and S…

7. Finally, you release the Web service definition for the SOAP runtime. The VI , WSD and configuration are created. Choose ‘Complete’.



If you select *Complete*, the following objects are created in the Object Navigator:

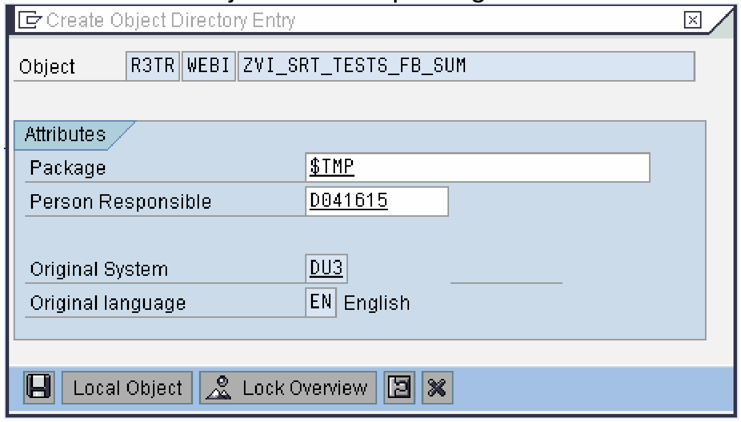
- The virtual interface **zvi_srt_tests_fb_sum**
- The Web service **zwsd_srt_tests_fb_sum'**

The address of the Web service is **default_host/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_sum'**

In addition, the Web service definition is released for

To find the address of the Web service, use the transaction WSCONFIG, which supports you when you publish a Web service in the UDDI.

8. Choose Local Object. For the package choose $TMP.



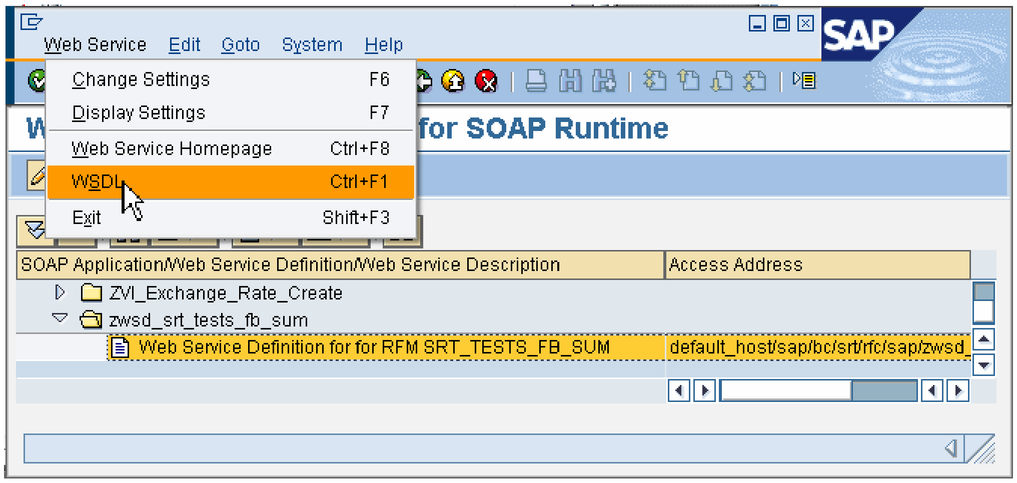
Object	R3TR	WEBI	ZVI_SRT_TESTS_FB_SUM
Package	$TMP		
Person Responsible	D041615		
Original System	DU3		
Original language	EN	English	

9. Press ‘Save’.

Get the WSDL from the web service

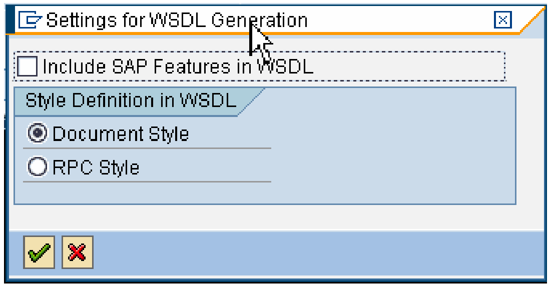
1. Start transaction WSADMIN. (Web Service Administrator for SOAP Runtime) and select the newly created Web Service ‘zwsd_srt_tests_fb_sum’.

2. Expand the node zwsd_srt_tests_fb_sum and select the Web Service Definition. From the menue choose Web Service -> WSDL.



SOAP Application/Web Service Definition/Web Service Description	Access Address
ZVI_Exchange_Rate_Create	
zwsd_srt_tests_fb_sum	
Web Service Definition for for RFM SRT_TESTS_FB_SUM	default_host/sap/bc/srt/rfc/sap/zwsd_

3. The ‘Settings for WSDL Generation’ popup tells you that there are two styles of WSDL supported: ‘Document Style’ and ‘RPC Style’. Choose ‘Document Style’ and then ‘OK’


Settings for WSDL Generation

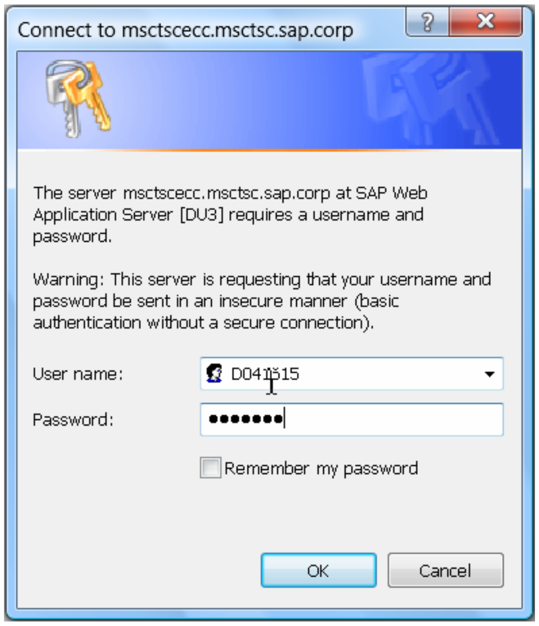
☐ Include SAP Features in WSDL

Style Definition in WSDL

[✓ Document Style]

RPC Style

4. You are prompted to enter the credentials to log on to the SAP NetWeaver Application Server.



Connect to msctscecc.msctsc.sap.corp

The server msctscecc.msctsc.sap.corp at SAP Web Application Server [DU3] requires a username and password.

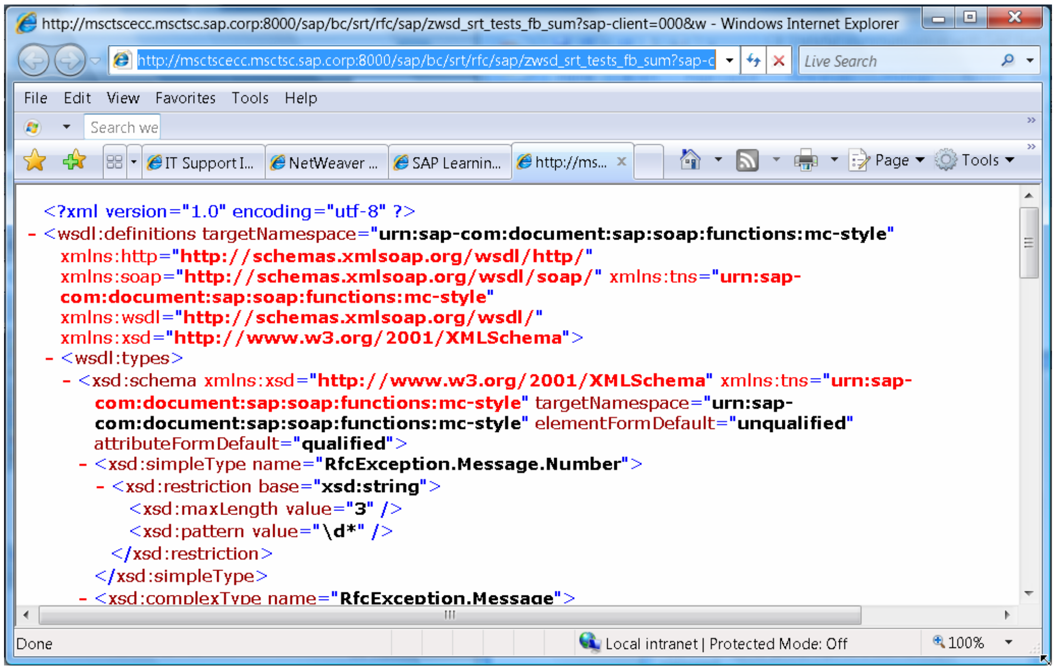
Warning: This server is requesting that your username and password be sent in an insecure manner (basic authentication without a secure connection).

User name: D041515

Password: •••••••

☐ Remember my password

5. The browser opens the URL that contains the WSDL of the Web service, in this case http://msctscecc.msctsc.sap.corp:8000/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_sum?sap-client=000&wsdl=1.1



http://msctscecc.msctsc.sap.corp:8000/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_sum?sap-client=000&w - Windows Internet Explorer

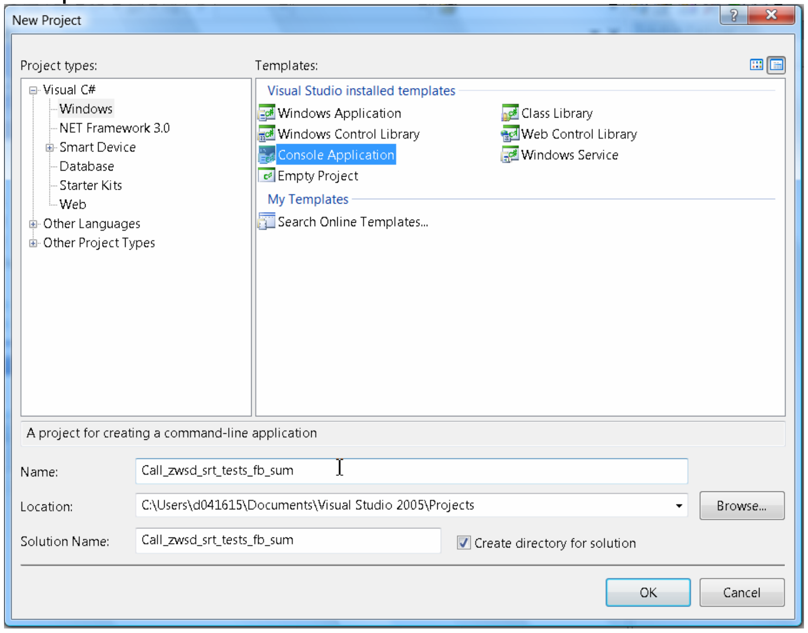
http://msctscecc.msctsc.sap.corp:8000/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_sum?sap-c

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="urn:sap-
    com:document:sap:soap:functions:mc-style"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="urn:sap-
        com:document:sap:soap:functions:mc-style" targetNamespace="urn:sap-
        com:document:sap:soap:functions:mc-style" elementFormDefault="unqualified"
        attributeFormDefault="qualified">
      - <xsd:simpleType name="RfcException.Message.Number">
        - <xsd:restriction base="xsd:string">
            <xsd:maxLength value="3" />
            <xsd:pattern value="\d*" />
          </xsd:restriction>
        </xsd:simpleType>
      - <xsd:complexType name="RfcException.Message">
```

Done   Local intranet | Protected Mode: Off   100%

Creating a web service client using Visual Studio 2005

1. Start Visual Studio to create a console application. From the menue select File -> New -> Project.
2. In the ‘New Project’ window select ‘console application’. Choose Call_zwsd_srt_tests_fb_sum as the name and the solution name for this project and press OK.



A project for creating a command-line application

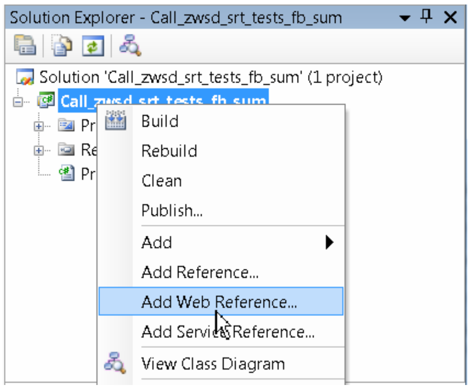
Name: Call_zwsd_srt_tests_fb_sum

Location: C:\Users\d041615\Documents\Visual Studio 2005\Projects

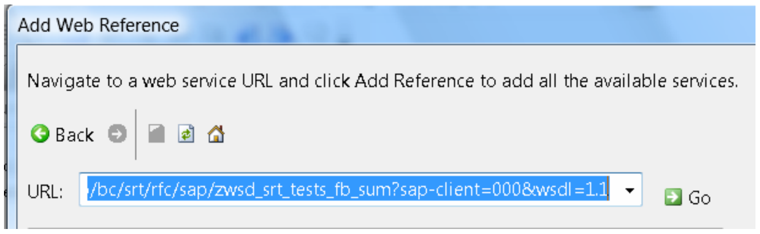
Solution Name: Call_zwsd_srt_tests_fb_sum

☑ Create directory for solution

3. In the Solution Explorer choose the project ‘Call_zwsd_srt_tests_fb_sum’ and press the right mouse button. In the context menue choose ‘Add Web Reference’.


Solution Explorer - Call_zwsd_srt_tests_fb_sum
Solution 'Call_zwsd_srt_tests_fb_sum' (1 project)
Call_zwsd_srt_tests_fb_sum
Build
Rebuild
Clean
Publish...
Add
Add Reference...
Add Web Reference...
Add Service Reference...
View Class Diagram

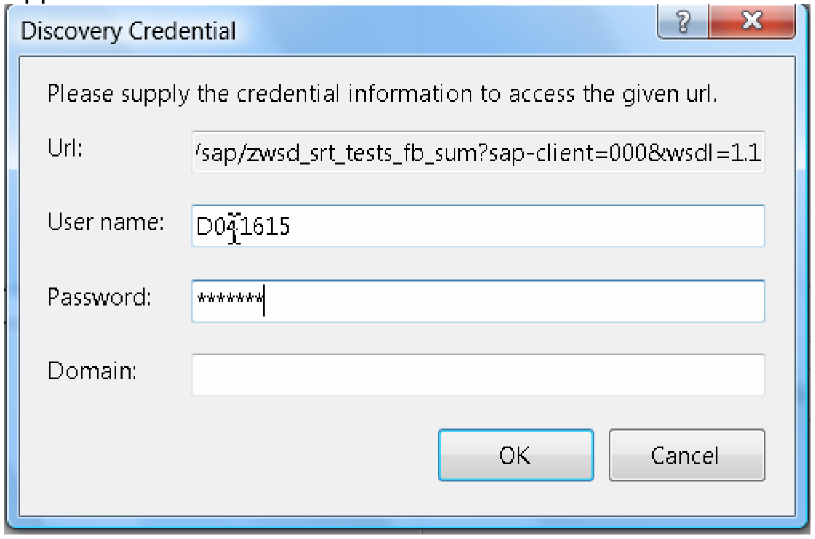
4. In the ‘Add Web Reference’ window enter the URL of the WSDL and press ‘Go’.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: /bc/srt/rfc/sap/zwsd_srt_tests_fb_sum?sap-client=000&wsdl=1.1

5. You are now prompted to enter your credentials for the SAP NetWeaver Application server.



Please supply the credential information to access the given url.

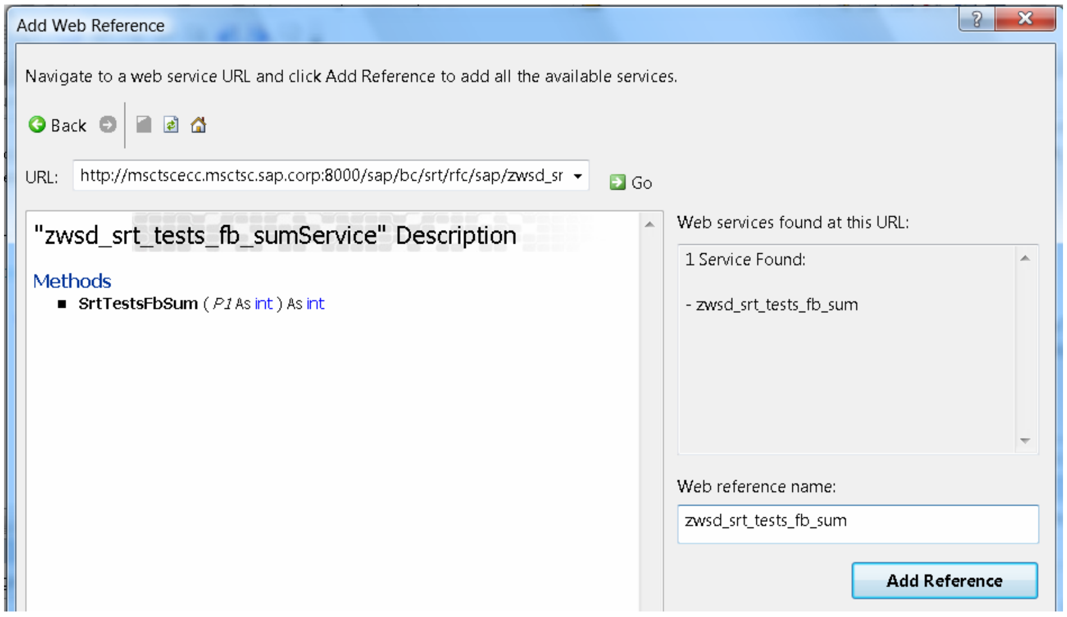
Url: 'sap/zwsd_srt_tests_fb_sum?sap-client=000&wsdl=1.1

User name: D041615

Password: *******

Domain:

6. The ‘Add Web Reference’ window now shows the methods of the web service. Change the Web reference name to zwsd_srt_tests_fb_sum and press the ‘Add Reference’ button.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://msctscecc.msctsc.sap.corp:8000/sap/bc/srt/rfc/sap/zwsd_sr

"zwsd_srt_tests_fb_sumService" Description

Methods

- SrtTestsFbSum (*P1* As int) As int

Web services found at this URL:

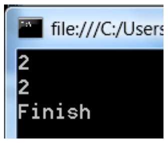
1 Service Found:

- zwsd_srt_tests_fb_sum

Web reference name: zwsd_srt_tests_fb_sum

7. Paste the coding of the class …

8. Start the console application in debug mode

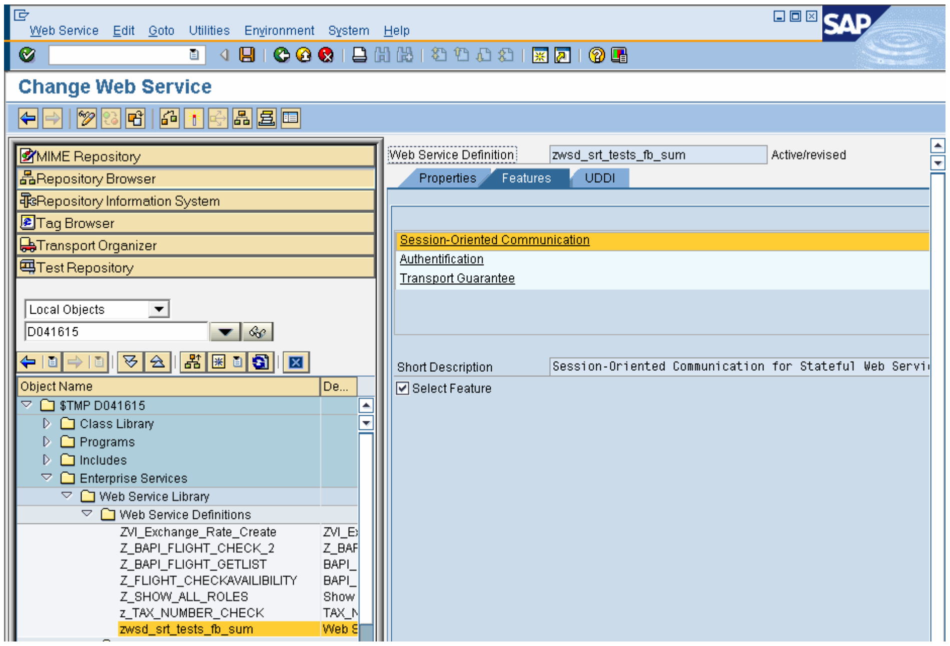


```
file:///C:/Users
2
2
Finish
```

The Results are the same! The sum does not grow. Why? Web Services are stateless.

Making the web service stateful

1. Open the Object Navigator (SE80), choose 'Local Objects' and display your own objects. Choose $TMP / <your user> -> Enterprise Services -> Web Service Library -> Web Services Definitions.
   a. EDIT 'zwsd_srt_tests_fb_sum'
   b. Select the Feature Session-Oriented Communication by clicking on the checkbox Select Feature
   c. Click on Save. Click on Check Click on the Activate button.
   d. You will see the status of the Web Service Definition becomes active

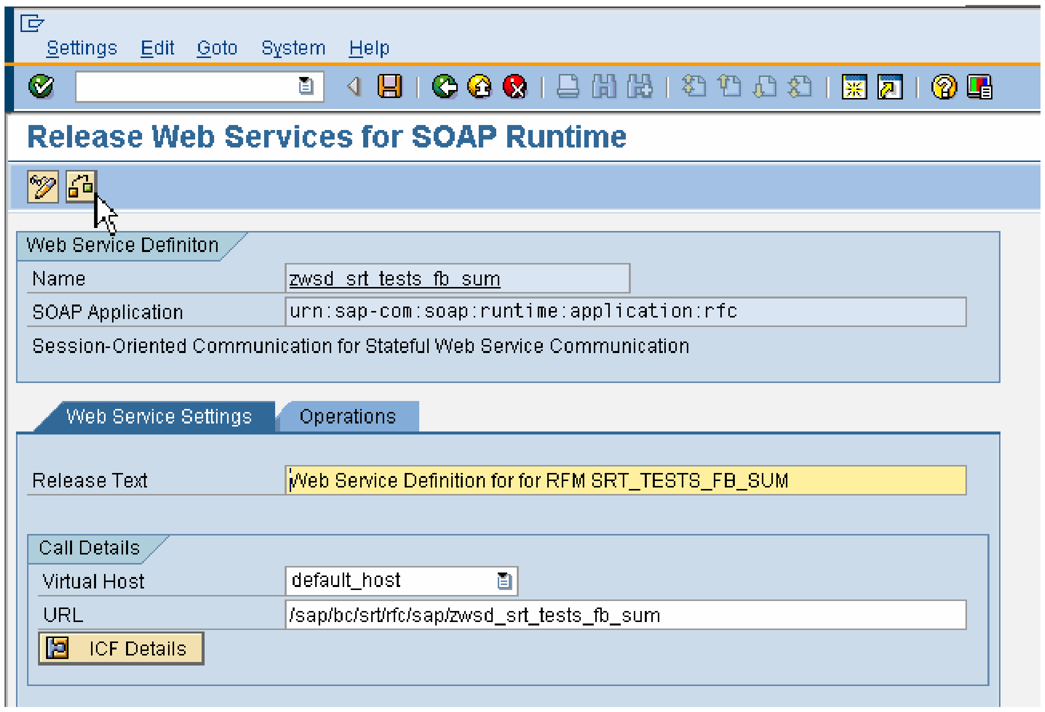


Web Service Definition: zwsd_srt_tests_fb_sum Active/revised

Short Description: Session-Oriented Communication for Stateful Web Servi

☑ Select Feature

2. Start transaction 'WSCONFIG' and enter the name of the Web service definition 'zwsd_srt_tests_fb_sum' and press enter. Mark your Web service ('ZWSD_SERIES_TESTXX'). Then choose 'Change'.



Release Web Services for SOAP Runtime

Web Service Definiton

Name	zwsd_srt_tests_fb_sum
SOAP Application	urn:sap-com:soap:runtime:application:rfc

Session-Oriented Communication for Stateful Web Service Communication

Web Service Settings | Operations

Release Text	Web Service Definition for for RFM SRT_TESTS_FB_SUM

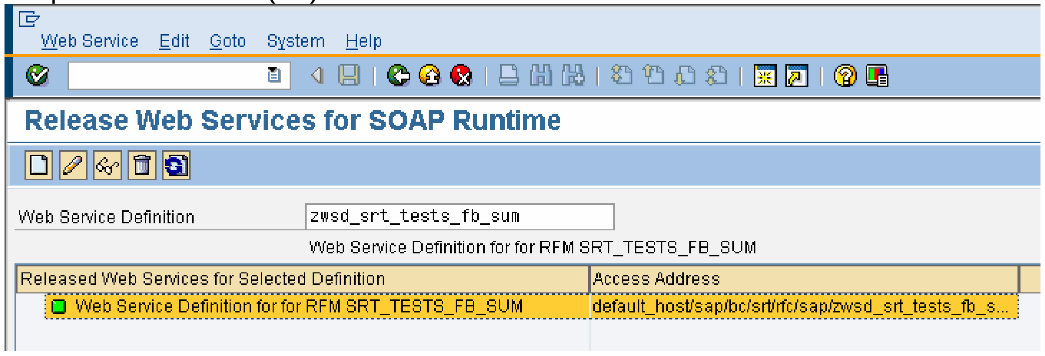
Call Details

Virtual Host	default_host
URL	/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_sum

ICF Details

3. Click Check (Ctrl+F2) You should see a message that Web Service configuration is consistent

4. Click Save (Ctrl+S) When prompted use the same Customizing Transport Request Click Back (F3)



Release Web Services for SOAP Runtime

Web Service Definition	zwsd_srt_tests_fb_sum

Web Service Definition for for RFM SRT_TESTS_FB_SUM

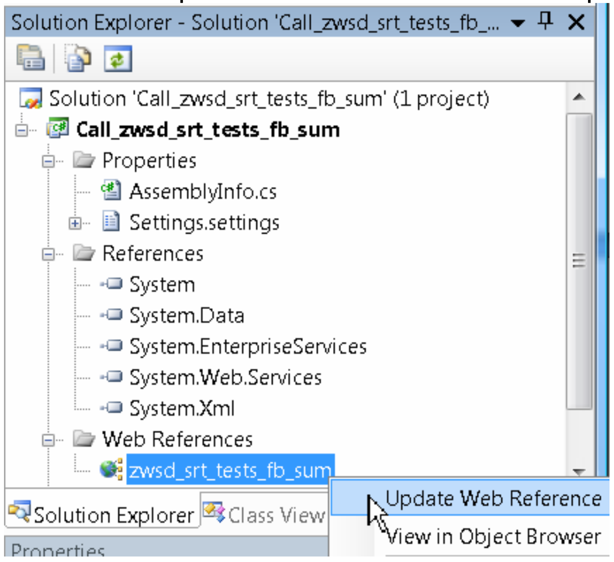
Released Web Services for Selected Definition	Access Address
🟩 Web Service Definition for for RFM SRT_TESTS_FB_SUM	default_host/sap/bc/srt/rfc/sap/zwsd_srt_tests_fb_s...

5. You should now have a Green Light

Retest the .NET web service client with the stateful web service

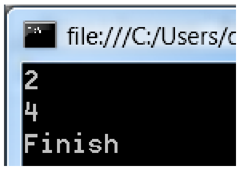
We will now retest the web service.

1. First we will update the web reference of our project.



2. Rebuild the application and start in debug mode.

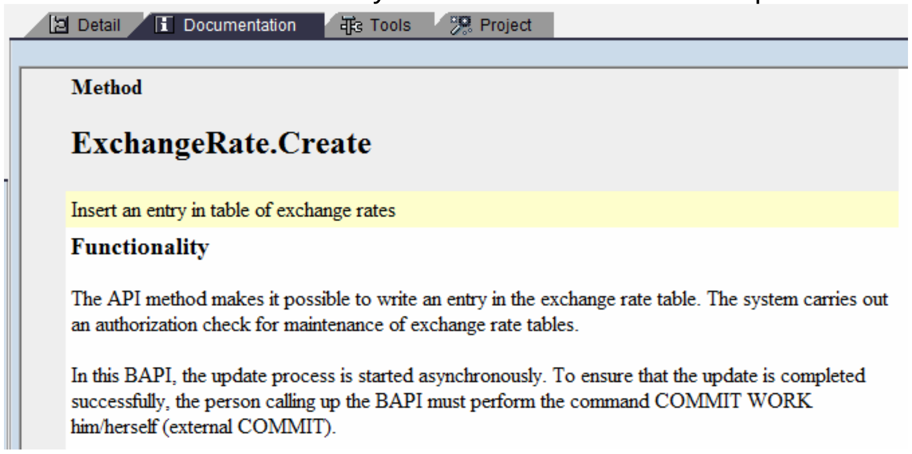
3. The result is now the same as we could observe it when testing the function module.



```
2
4
Finish
```

Using Transaction Commit

As mentioned in the introduction session / state oriented communication can also be used for calling BAPIs via Web Service which require an external BAPI TRANSACTION COMMIT. An example of such a BAPI is BAPI_EXCHANGERATE_CREATE, if you take a look to the documentation you will see that this BAPI requires an external COMMIT.



Method

ExchangeRate.Create

Insert an entry in table of exchange rates

Functionality

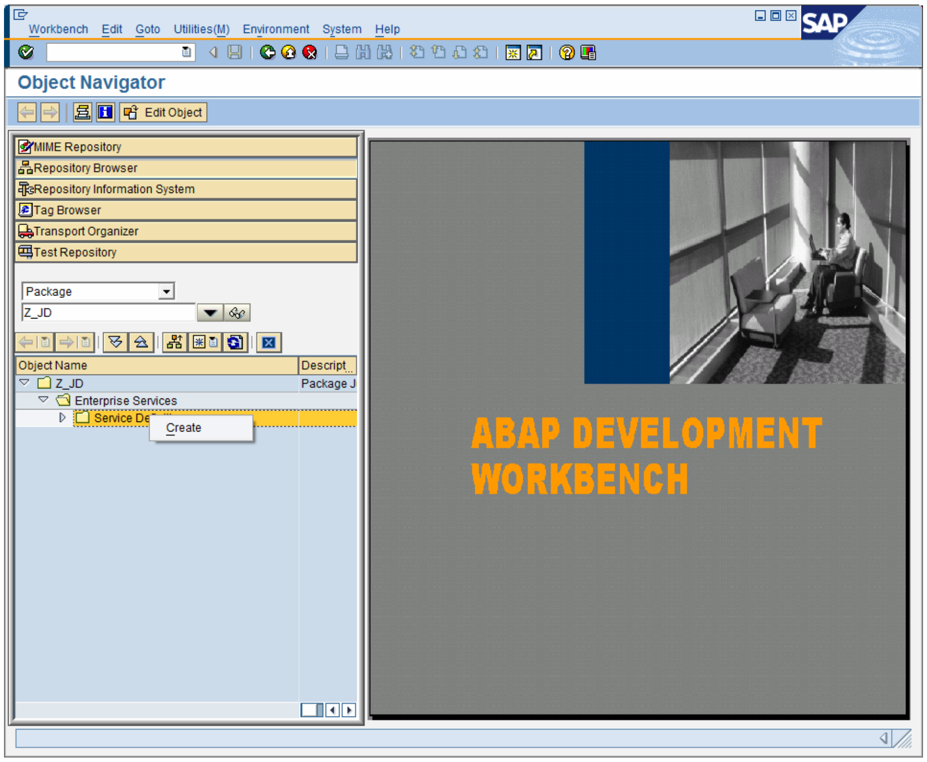
The API method makes it possible to write an entry in the exchange rate table. The system carries out an authorization check for maintenance of exchange rate tables.

In this BAPI, the update process is started asynchronously. To ensure that the update is completed successfully, the person calling up the BAPI must perform the command COMMIT WORK him/herself (external COMMIT).

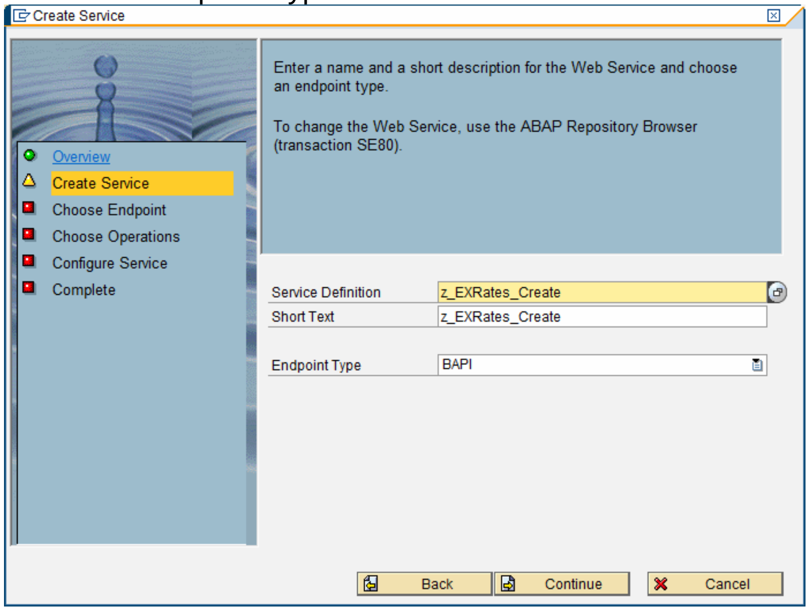
How-To Section External COMMIT

Create the Web Service

1. Start transaction SE80, go to Repository Browser, and open Enterprise Services. Right-click on Enterprise Services and choose *Create*.



2. The Web Service Creation Wizard starts, at the first screen click *Continue.* On the next screen specify a name for the Service Definition and a Short Text and choose as Endpoint Type BAPI.

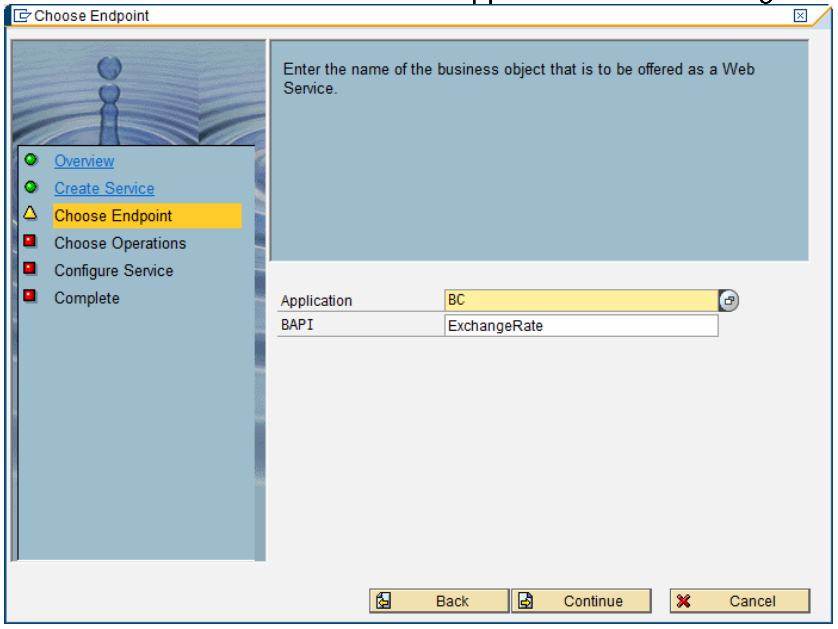


Enter a name and a short description for the Web Service and choose an endpoint type.

To change the Web Service, use the ABAP Repository Browser (transaction SE80).

Service Definition	z_EXRates_Create
Short Text	z_EXRates_Create
Endpoint Type	BAPI

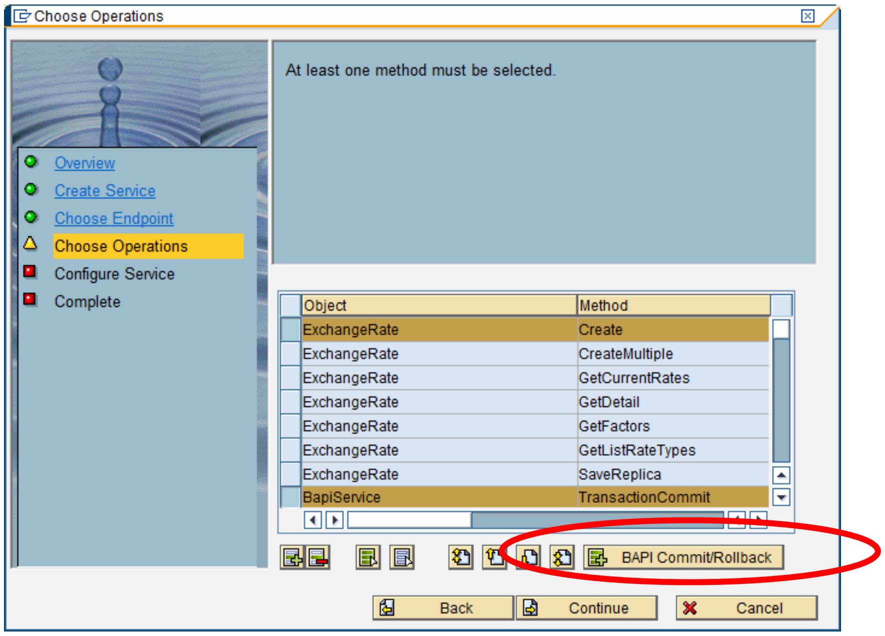
3. On the next screen choose BC as Application and ExchangeRate as BAPI



Enter the name of the business object that is to be offered as a Web Service.

Application	BC
BAPI	ExchangeRate

4. Mark the method Create in the list and add the BAPI Transaction Commit/Rollback



At least one method must be selected.

Object	Method
ExchangeRate	Create
ExchangeRate	CreateMultiple
ExchangeRate	GetCurrentRates
ExchangeRate	GetDetail
ExchangeRate	GetFactors
ExchangeRate	GetListRateTypes
ExchangeRate	SaveReplica
BapiService	TransactionCommit

5. Afterwards finish the Web Service Creation Wizard as described in the section Create a web service from the function module SRT_TESTS_FB_SUM using the Web Service Creation Wizard. For information on how to get the WSDL see Get the WSDL from the web service, for information on how to make the Web Service stateful see Making the web service stateful.

Develop a .NET application which handles external BAPI COMMITS

The development of the .NET application is the same as described in chapter Creating a web service client using Visual Studio 2005.
The following code snippet shows how to handle the BAPI_TRANSACTION_COMMIT in .NET, the complete code can be found in the Appendix.

```
CookieContainer cookie = new CookieContainer();
z_Create_ExchangeRatesService.z_Create_ExchangeRatesService _proxy
      = new z_Create_ExchangeRatesService.z_Create_ExchangeRatesService();
_proxy.Credentials =
      new NetworkCredential(Properties.Settings.Default.SAPUser,
                            Properties.Settings.Default.SAPPassword);
_proxy.CookieContainer = cookie;
_proxy.ExchangeRateCreate("X",
                          "000",
                          newData,
                          "X",
                          out strRateTypeOut,
                          out bapiReturn,
                          out strCurrTo);
_proxy.BapiServiceTransactionCommit("");
```

Appendix

Source code SRT_TESTS_FB_SUM

```
using System;
using System.Net;
using System.Collections.Generic;
using System.Text;
using System.Web.Services.Protocols;
using System.Web.Services;


namespace Call_zwsd_srt_tests_fb_sum
{
    class Program
    {
        static void Main(string[] args)
        {

            try
            {
                //Sample Project for calling WS in SAP and writing data

                int P1 = 2;
                int result = 0;

                CookieContainer cookie = new CookieContainer();

                zwsd_srt_tests_fb_sum.zwsd_srt_tests_fb_sumService _proxy1 = new
                Call_zwsd_srt_tests_fb_sum.zwsd_srt_tests_fb_sum.zwsd_srt_tests_
                fb_sumService();

                _proxy1.Credentials = new NetworkCredential("myuser",
                                                           "secret pwd");
                _proxy1.CookieContainer = cookie;
                result = _proxy1.SrtTestsFbSum(P1, true);
                Console.WriteLine(result);

                cookie = _proxy1.CookieContainer;
                zwsd_srt_tests_fb_sum.zwsd_srt_tests_fb_sumService _proxy2 =
                    new
                    Call_zwsd_srt_tests_fb_sum.zwsd_srt_tests_fb_sum.zwsd_srt_te
                    sts_fb_sumService();

                _proxy2.Credentials = new NetworkCredential("myuser",
                                                           "secret pwd");
                _proxy2.CookieContainer = cookie;
                result = _proxy2.SrtTestsFbSum(P1, true);
                Console.WriteLine(result);
                Console.WriteLine("Finish");
                string response = Console.ReadLine();
            }

            catch (SoapException exc)
            {
                Console.WriteLine("Exception thrown");
                throw exc;
                string response = Console.ReadLine();
            }
        }
    }
}
```

Source BAPI_TRANSACTION_COMMIT

```
using System;
using System.Net;
using System.Collections.Generic;
using System.Text;
using System.Web.Services.Protocols;
using System.Web.Services;

namespace SAP_WebService_Calls
{
    class CreateEXRates
    {
       static void Main(string[] args)
        {
            try
            {
                //Sample Project for calling WS in SAP and writing data
                z_Create_ExchangeRatesService.Bapiret2 bapiReturn =
                    new z_Create_ExchangeRatesService.Bapiret2();
                z_Create_ExchangeRatesService.Bapi10930 newData =
                    new z_Create_ExchangeRatesService.Bapi10930();
                newData.ExchRate = 1.112M;
                newData.ExchRateV = 0.0M;
                newData.FromCurr = "EUR";
                newData.FromFactor = 1;
                newData.FromFactorV = 0;
                newData.RateType = "M";
                newData.ToCurrncy = "USD";
                newData.ToFactor = 1;
                newData.ToFactorV = 0;
                newData.ValidFrom = "2007-10-01";
                string strRateTypeOut = "";
                string strCurrTo = "";
                CookieContainer cookie = new CookieContainer();

                z_Create_ExchangeRatesService.z_Create_ExchangeRatesService
                 _proxy = new
                 z_Create_ExchangeRatesService.z_Create_ExchangeRatesService();
                _proxy.Credentials =
                    new NetworkCredential(Properties.Settings.Default.SAPUser,
                                 Properties.Settings.Default.SAPPassword);
                _proxy.CookieContainer = cookie;
                _proxy.ExchangeRateCreate("X",
                                       "000",
                                       newData,
                                       "X",
                                       out strRateTypeOut,
                                       out bapiReturn,
                                       out strCurrTo);
                _proxy.BapiServiceTransactionCommit("");
                Console.WriteLine("Finish");
                string response = Console.ReadLine();
            }
            catch (SoapException exc)
            {
                throw exc;
            }
        }
    }
}
```