April 2006

Building Microsoft .NET Applications using Visual Studio 2005 and SAP NetWeaver Web Services

Author

Birgit Steiner, Architect, Softlab GmbH, birgit.steiner@softlab.de

Co-Editors

Tilo Böttcher, SAP Program Manager CTSC, Global SAP Alliance, Microsoft, tiloboet@microsoft.com
Jürgen Daiberl, SAP Program Manager CTSC, Global SAP Alliance, Microsoft, jdaiberl@microsoft.com

Summary

This paper provides information about how to use the Web Services SAP NetWeaver Application Server offers with Visual Studio 2005. The paper contains a detailed walk-through of how to configure SAP NetWeaver for offering Web Services including screenshots and the necessary SAP transactions and a detailed explanation of how to write a .NET application using the Web Services from SAP. In addition the author describes a possible approach for tracing and debugging.

Applies to

- Microsoft Visual Studio .NET 2005
- Microsoft .NET Framework 2.0
- SAP NetWeaver 04
- SAP NetWeaver Application Server
- SAP WebAS 6.20 ï 6.40

Keywords

SAP NetWeaver, Web Services, Visual Studio 2005, SAP Integration

Level of difficulty

Technical consultants, Solution Architects, Developers

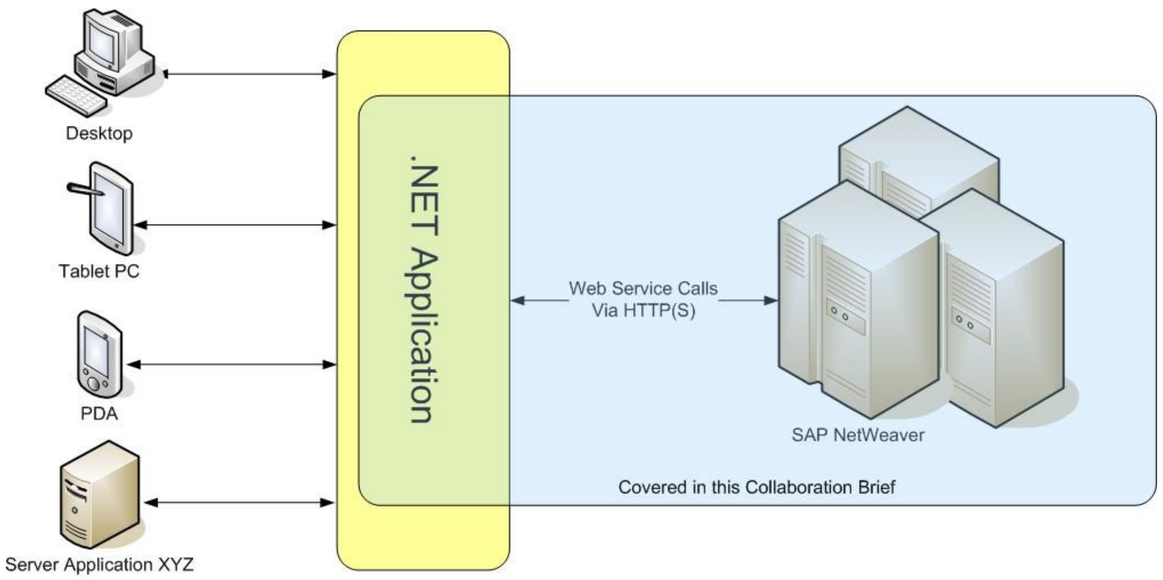
Contact

This document is provided to you by the Oxford Computer Group Ltd. and the Collaboration Technology Support Center Microsoft, a joint team from SAP and Microsoft that drives interoperability.
For feedback or questions you can contact the Softlab GmbH at info@softlab.de the CTSC at ctsc@sap.com or ctsc@microsoft.com. Please check the .NET interoperability area in the SAP Developer Network (http://sdn.sap.com) and at the Microsoft-SAP Alliance web site (http://www.microsoft-sap.com) for any updates or further information.

Contents

Introduction

Using the SAP NetWeaver Application Server Web Services from .NET code is as easy as using any other Web Service written in Java, .NET or any other programming language. So, why to write a whitepaper?
It is easy if everything is set up properly. So the first part will show you what you need to set up a Web Service in SAP and how to create the .NET Web Service client.
It is easy if everything works but as experience shows it often doesn't. That's why we will show common issues and how you can troubleshoot errors.
And last but not least we will walk you through the creation of a SAP Web Service using an existing BAPI[1] and a .NET Web Service client step-by-step.



This paper won't cover the basic concepts and standards of Web Services, how to write a Web Service, neither in .NET nor in SAP, how to call Web Services from SAP, and won't discuss architectural topics like "contract-first", messaging patterns, UDDI, etc.

The Basics

In this whitepaper we use the latest software versions of Microsoft and SAP. Both have a sound support of Web Service technology and standards - Microsoft with its products .NET Framework 2.0 and Visual Studio 2005 and SAP with the NetWeaver Application Server 6.40.

Microsoft .NET supports Web Services since the advent of the .NET Framework 1.0. Since Visual Studio .NET the IDE comes with built in support for Web Service. On the service side we have ASP.NET Web Service project template. On the client side we have the Add Web Reference wizard that creates the Web Service proxy classes from a WSDL. Generating Web Service and Web Service client classes from a WSDL can also

[1] BAPI is a Business Application Programming Interface to existing object orientated methods, used for synchronous communication, implemented as Remote Function Call

be achieved by the tool wsdl.exe that comes with Visual Studio. In the core product the standards SOAP 1.1 and 1.2, WSDL, UDDI and WS-I BP 1.0 are integrated. With WSE (Web Service Enhancements) Microsoft provides a tool kit that supports and eases the implementation of the WS-* standards support, for instance WS-Security, in Web Services.

The SAP NetWeaver Application Server 6.40 comes with an infrastructure for Web Services that implements the basic Web Services standards like SOAP 1.1, WSDL, UDDI and WS-I BP 1.0. From the WS-* standards it currently supports WS-Security only. On top of this infrastructure the Web Service Toolset offers wizard driven configuration of Web Service interfaces, WSDs, security and technical communication features. It also supports publishing of Web Services in UDDI registries.

In both technologies there are frameworks that hide the raw SOAP message from the developer, i.e. the developer can write a BAPI in SAP or a method in .NET as usual without having knowledge of the format of a SOAP message. In the clients you can even call a Web Service method like any other method of an object. The environments (SAP NetWeaver Application Server and .NET Framework) handle the transformation from ñnormalò code into SOAP messages for the developer. This is done in the underlying infrastructure by serializing the call into the SOAP message and by deserializing incoming calls for instance into an instance of a .NET object. This applies for both directions, i.e. SOAP requests and SOAP responses. In .NET this works not only with ñgoodò messages, also exceptions thrown in the code are transformed into SOAP faults. In SAP error handling looks a little bit different, since there is a return structure that holds error messages for ñlogicalò errors, BAPIRET or BAPIRET2, these are returned to the caller as any other SOAP response. Only errors in the infrastructure are returned as SOAP faults, e.g. if a Web Service does not support the called method.
Before any system can communicate with your Web Service you have to provide the interface description in form of a WSDL. This is also created automatically by both technologies.

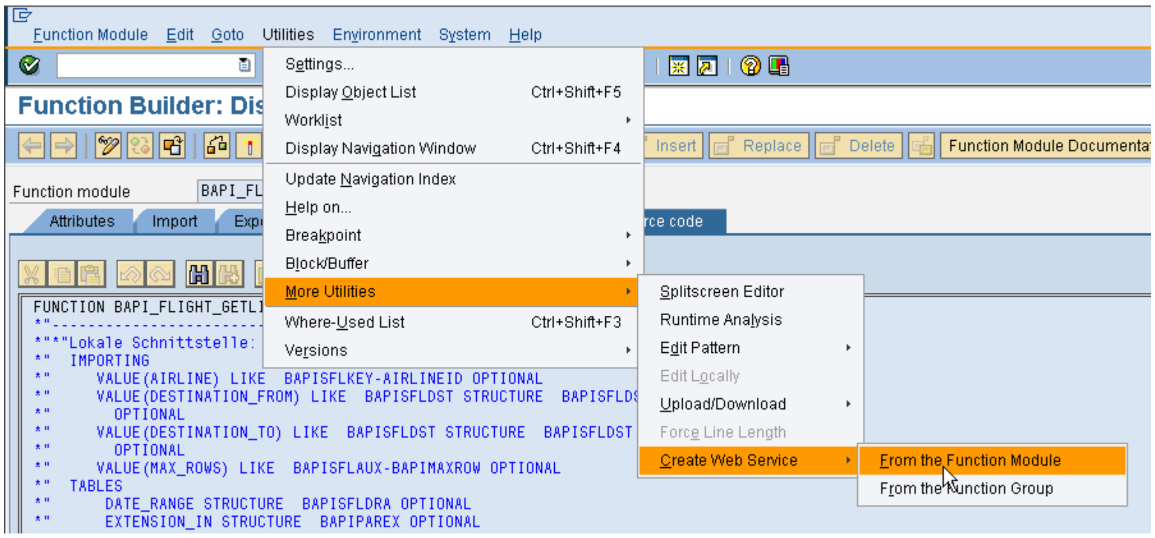
Make It Work

What do we want to achieve? We want to call a SAP Web Service from .NET. Technically speaking this means there must be a SAP Web Service and then we have to build a .NET client that is able to communicate with this Web Service.

Set up SAP Web Service

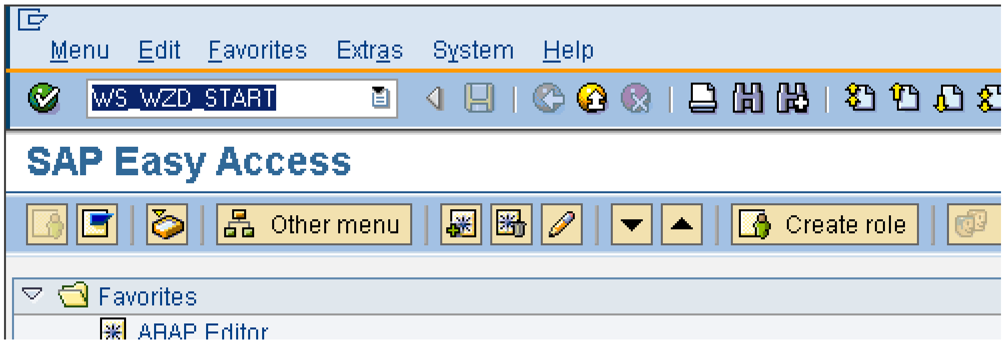
To create a Web Service in SAP you do not have to write ABAP [2]or Java code, you can just configure an existing BAPI or RFC[3].
For this you must start the Web Service Creation Wizard. There are three different entry points to the wizard. You can select a BAPI or a remote function module in transaction SE37 or SE80 and start the Web Service Creation Wizard for it.

[2] ABAP (Advanced Business Application Programming) is the object-oriented programming language for SAP
[3] RFC (Remote Function Call) is a application program interface for SAP applications and is used for synchronous and asynchronous communication

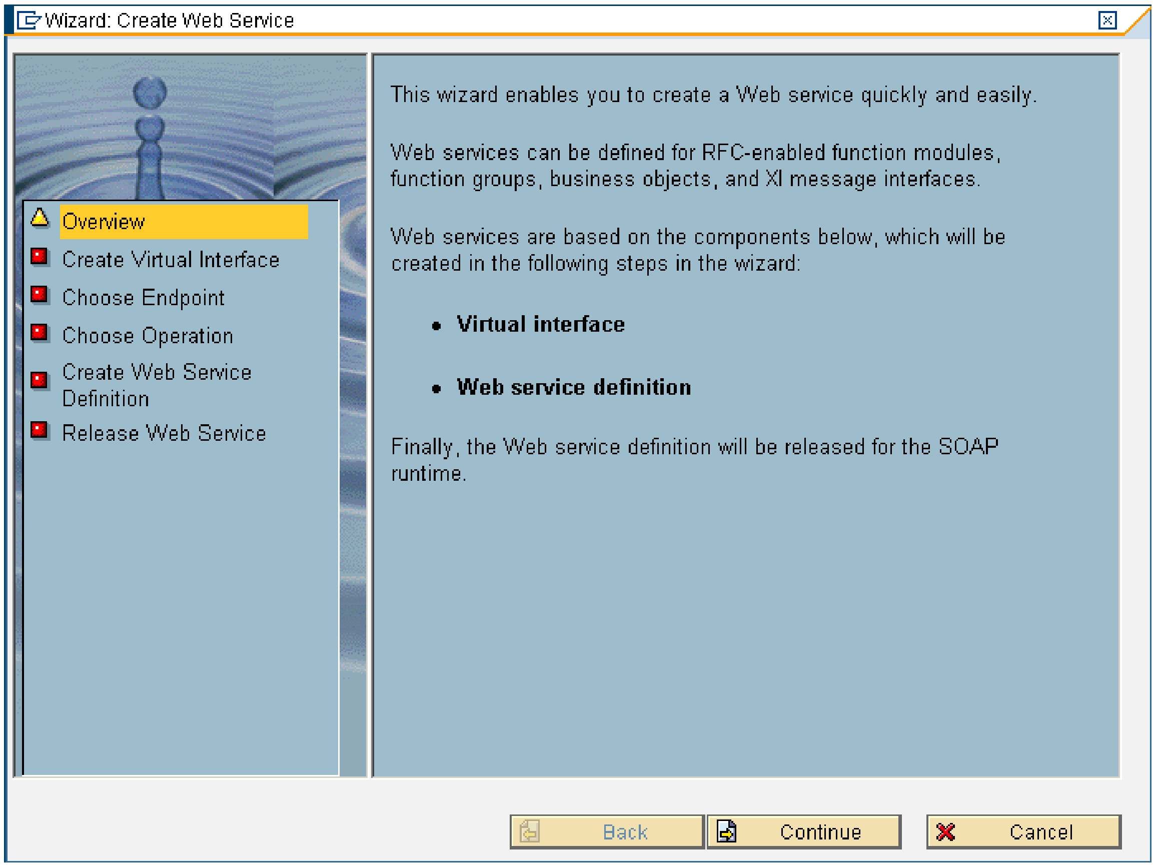


```
FUNCTION BAPI_FLIGHT_GETLI
*"----------------------
*"*"Lokale Schnittstelle:
*"  IMPORTING
*"     VALUE(AIRLINE) LIKE  BAPISFLKEY-AIRLINEID OPTIONAL
*"     VALUE(DESTINATION_FROM) LIKE  BAPISFLDST STRUCTURE  BAPISFLDS
*"       OPTIONAL
*"     VALUE(DESTINATION_TO) LIKE  BAPISFLDST STRUCTURE  BAPISFLDST
*"       OPTIONAL
*"     VALUE(MAX_ROWS) LIKE  BAPISFLAUX-BAPIMAXROW OPTIONAL
*"  TABLES
*"      DATE_RANGE STRUCTURE  BAPISFLDRA OPTIONAL
*"      EXTENSION_IN STRUCTURE  BAPIPAREX OPTIONAL
```

Another way is to launch the wizard directly with the transaction code WS_WZD_START.



The wizard guides you through the steps of creating a virtual interface, choosing the endpoint and operation, creating the Web Service definition and releasing the Web Service.



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.

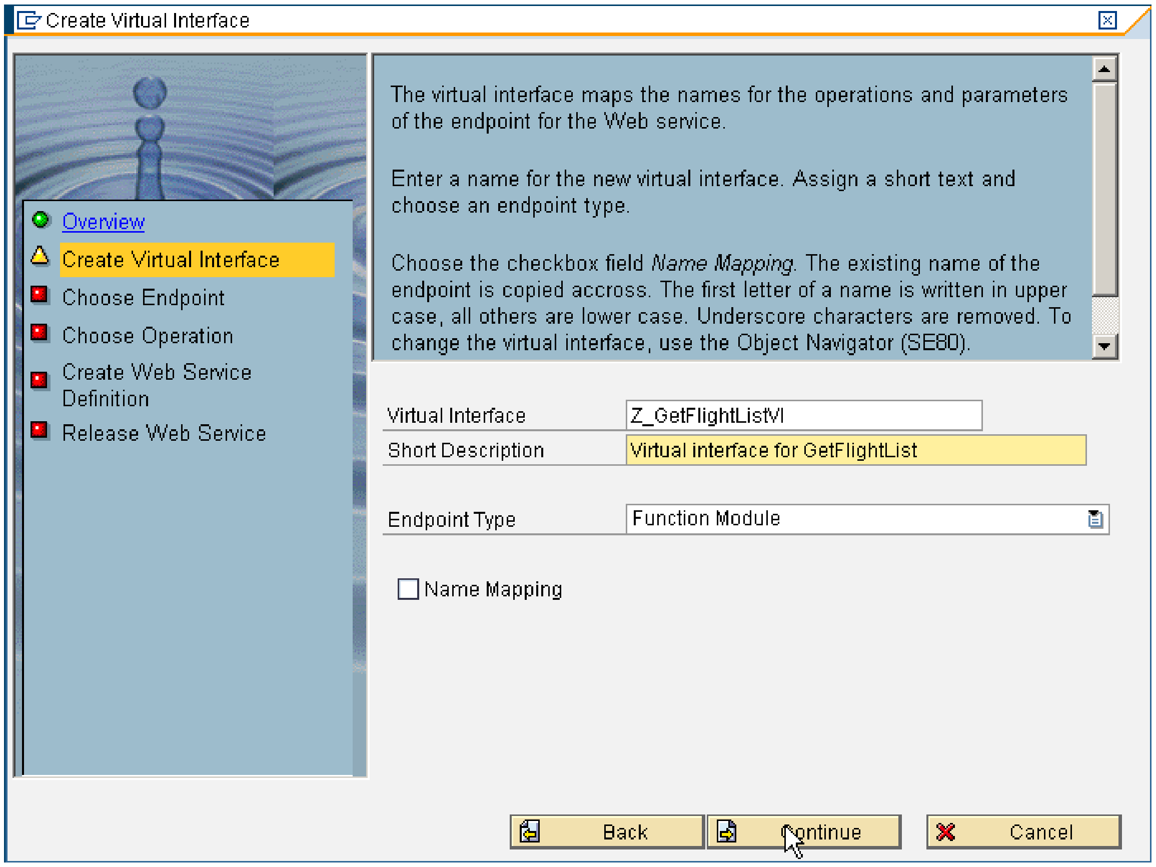
Web services are based on the components below, which will be created in the following steps in the wizard:

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

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

Before you start you should check if you have a package for storing your Web Service. You can check this in the ABAP Development Workbench (transaction code SE80) in the Repository Browser. How you can check and create a package is described in Check for Package.

Stepping through the Web Service Wizard is quite straight forward. For your first Web Service we would suggest that you stick to the default settings and do not configure name mapping or WS-Security for instance. You, especially .NET developers, have to pay attention to use a Virtual Interface name and a Web Service Definition name from the same namespace as your package, i.e. in our case both have to start with Z.


Create Virtual Interface

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

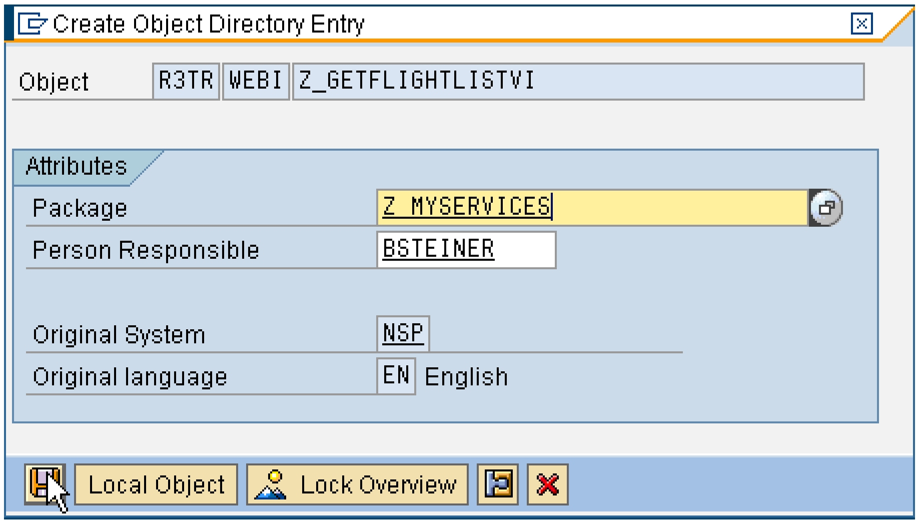
Enter a name for the new virtual interface. Assign a short text and choose an endpoint type.

Choose the checkbox field *Name Mapping*. The existing name of the endpoint is copied accross. The first letter of a name is written in upper case, all others are lower case. Underscore characters are removed. To change the virtual interface, use the Object Navigator (SE80).

Virtual Interface	Z_GetFlightListVI
Short Description	Virtual interface for GetFlightList
Endpoint Type	Function Module

☐ Name Mapping

When you finish the wizard SAP creates all objects and their corresponding transport requests for you. That is the point where you need a package for your Web Services.


Create Object Directory Entry

Object	R3TR	WEBI	Z_GETFLIGHTLISTVI

Attributes

Package	Z_MYSERVICES
Person Responsible	BSTEINER
Original System	NSP
Original language	EN English

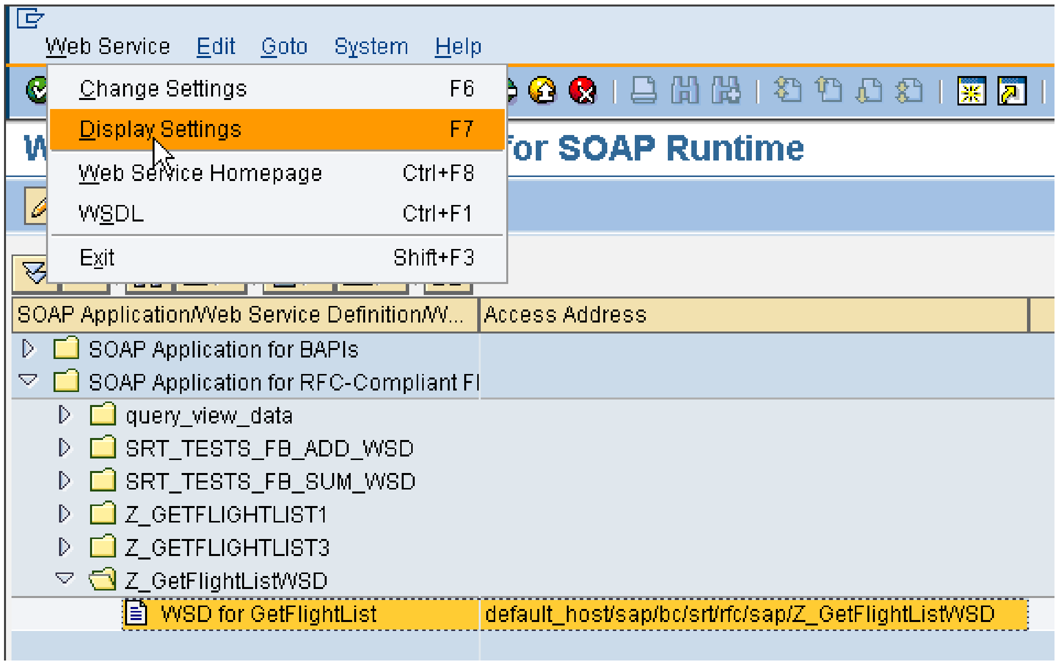
Instead of using the Web Service Creation Wizard you could have created the Virtual Interface and the Web Service Definition manually using the transaction SE80.

So the Web Service was created, but where is it? There are multiple transactions to view the Web Service:

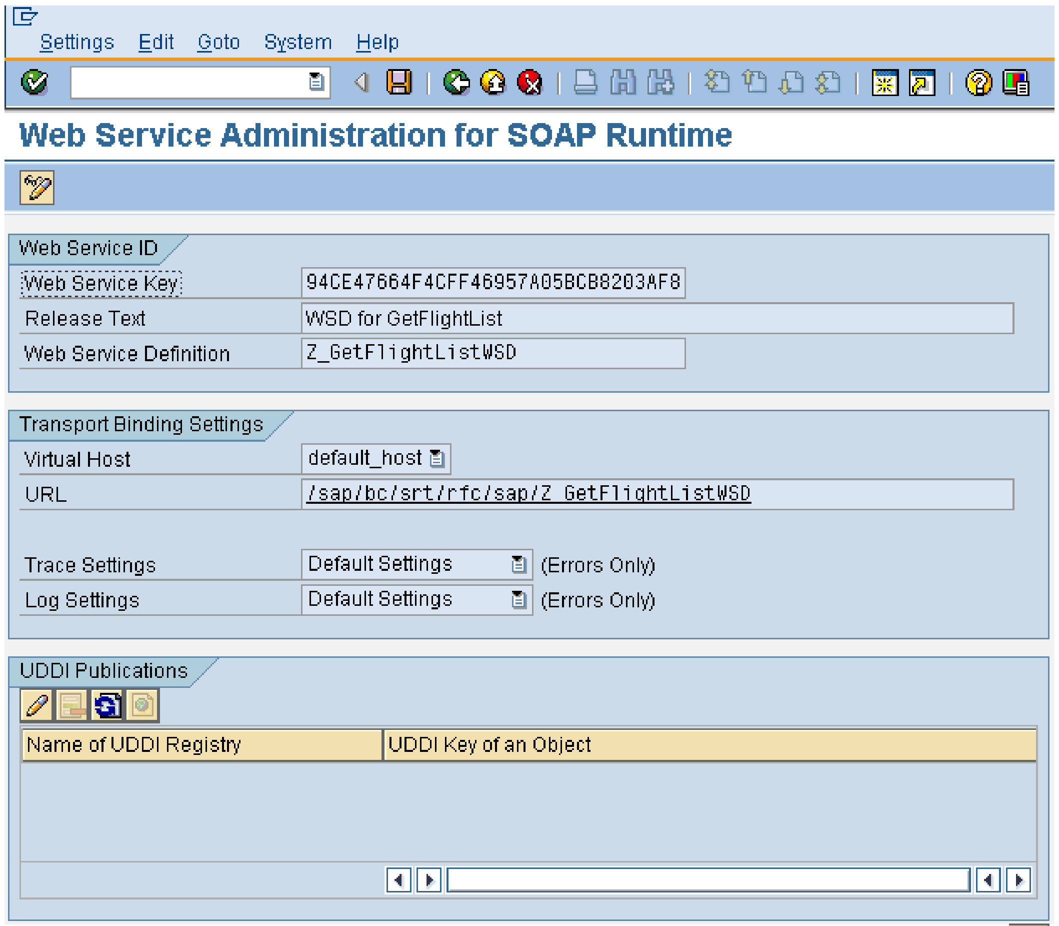
¶ SE80
beneath your package you find the settings for the Web Service Definition and the Virtual Interface. Here you can for instance configure security settings for the definition or change the Web Serviceֳs parameter names in the interface.

¶ WSCONFIG
here you can release your Web Service definition for the SOAP runtime, configure the address or a security profile for your Web Service. It also offers a consistency check for your Web Service configuration.

¶ WSADMIN
In this transaction you can view and change the logging and tracing settings of the Web Service.



SOAP Application/Web Service Definition/W...	Access Address
SOAP Application for BAPIs	
SOAP Application for RFC-Compliant Fl	
query_view_data	
SRT_TESTS_FB_ADD_WSD	
SRT_TESTS_FB_SUM_WSD	
Z_GETFLIGHTLIST1	
Z_GETFLIGHTLIST3	
Z_GetFlightListWSD	
WSD for GetFlightList	default_host/sap/bc/srt/rfc/sap/Z_GetFlightListWSD



Web Service Administration for SOAP Runtime

Web Service ID

Web Service Key	94CE47664F4CFF46957A05BCB8203AF8
Release Text	WSD for GetFlightList
Web Service Definition	Z_GetFlightListWSD

Transport Binding Settings

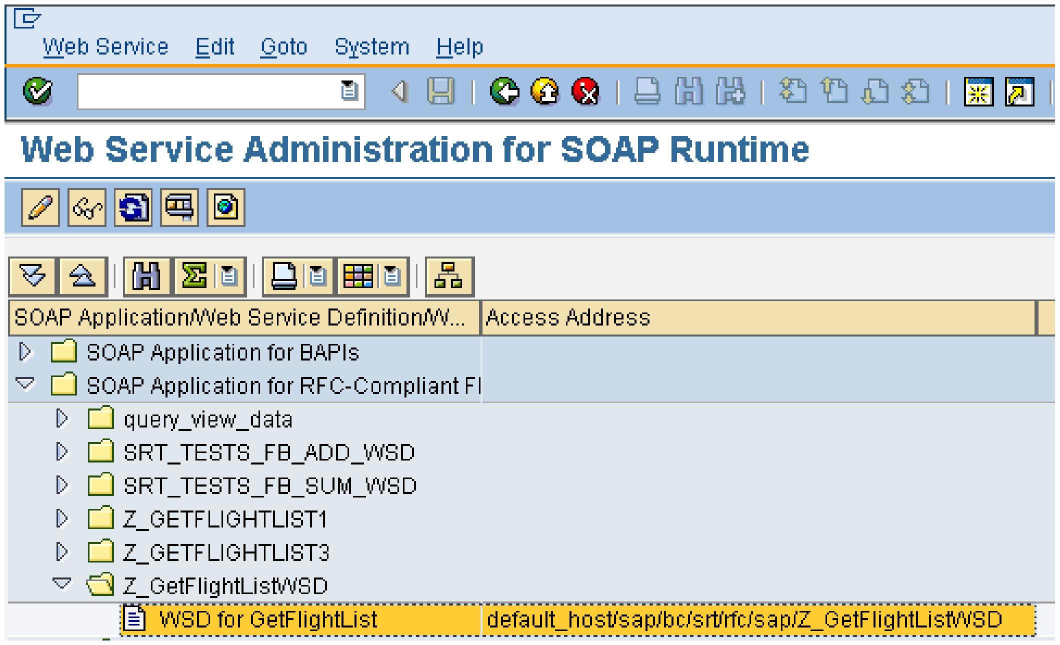
Virtual Host	default_host
URL	/sap/bc/srt/rfc/sap/Z_GetFlightListWSD
Trace Settings	Default Settings (Errors Only)
Log Settings	Default Settings (Errors Only)

UDDI Publications

Name of UDDI Registry	UDDI Key of an Object

Info: In SAP you can switch between display and change mode with the icon .

From within the WSADMIN transaction you can also test the Web Service or view the Web Serviceȇs WSDL.
Testing can be started either via the Web Service Homepage entry in the Web Service menu or with the icon .



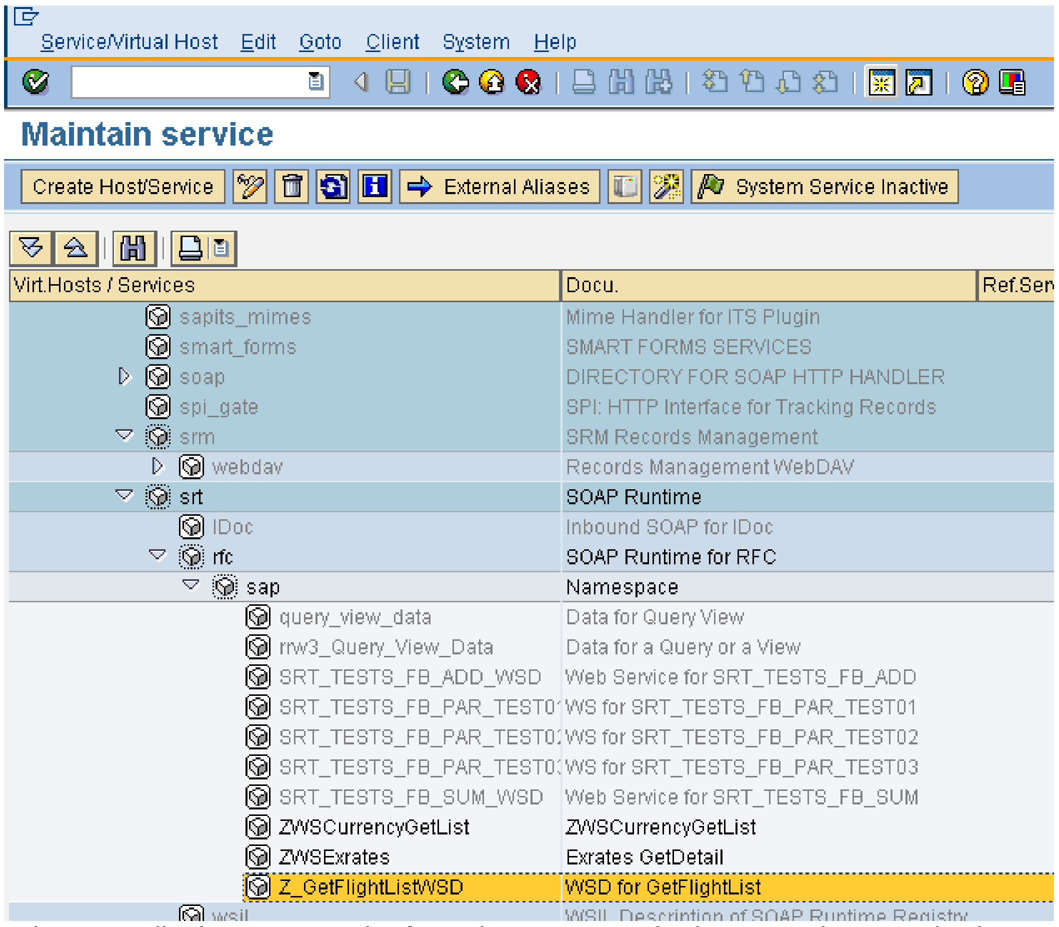
Web Service Administration for SOAP Runtime

SOAP Application/Web Service Definition/W...	Access Address
SOAP Application for BAPIs	
SOAP Application for RFC-Compliant FI	
query_view_data	
SRT_TESTS_FB_ADD_WSD	
SRT_TESTS_FB_SUM_WSD	
Z_GETFLIGHTLIST1	
Z_GETFLIGHTLIST3	
Z_GetFlightListWSD	
WSD for GetFlightList	default_host/sap/bc/srt/rfc/sap/Z_GetFlightListWSD

It can happen that you get the following error message Settings for J2EE Server do not exist. For our scenario this is not a problem, since the Web Service will work nonetheless.

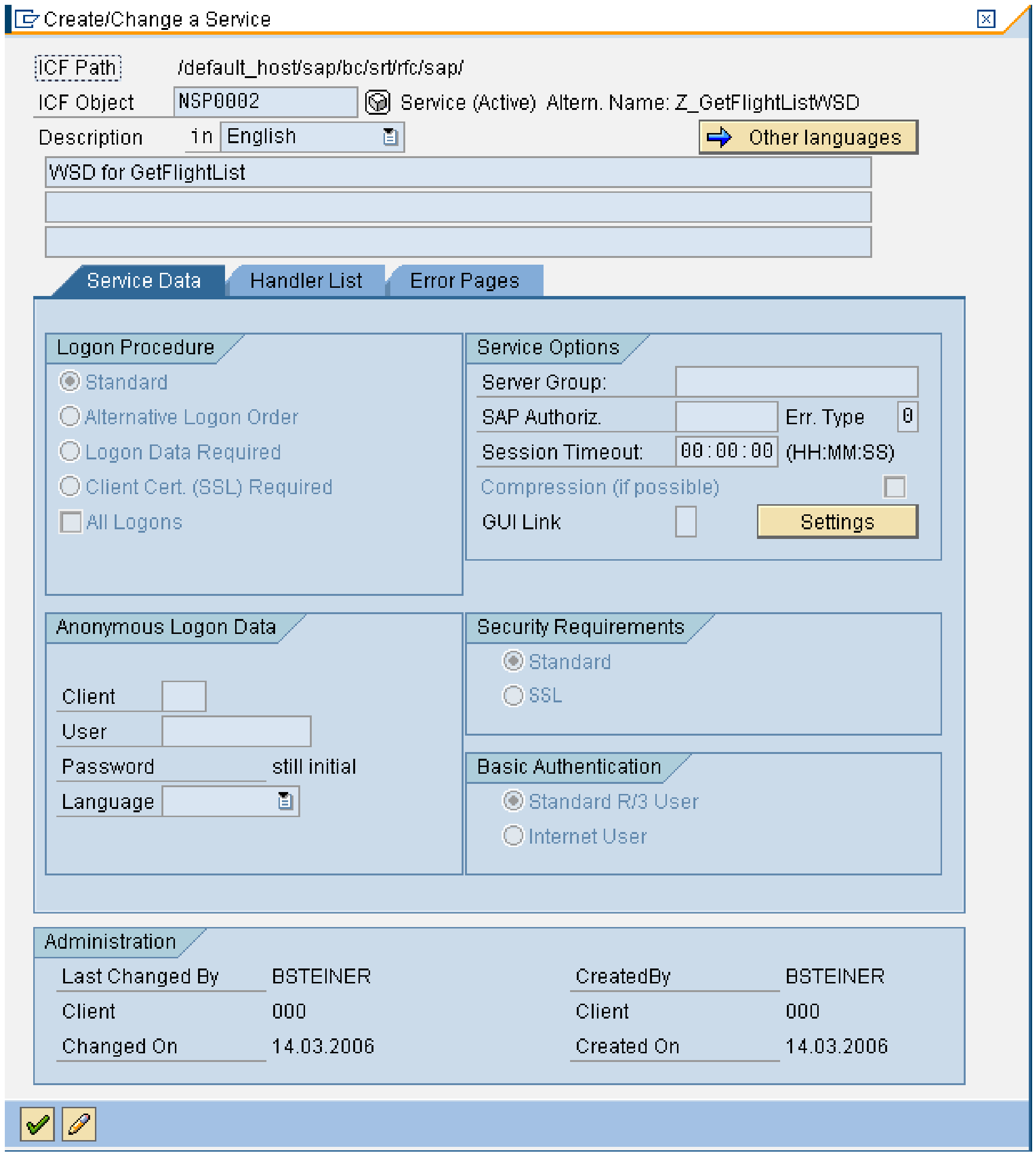
Apart from testing you can also display the WSDL either with the menu or with the icon.

¶ SICF
In the transaction SICF you can display and change the Web server settings for your Web Service. You will find your Web Service beneath default_host/sap/bc/srt/rfc/sap.



Virt.Hosts / Services	Docu.	Ref.Ser
sapits_mimes	Mime Handler for ITS Plugin	
smart_forms	SMART FORMS SERVICES	
soap	DIRECTORY FOR SOAP HTTP HANDLER	
spi_gate	SPI: HTTP Interface for Tracking Records	
srm	SRM Records Management	
webdav	Records Management WebDAV	
srt	SOAP Runtime	
IDoc	Inbound SOAP for IDoc	
rfc	SOAP Runtime for RFC	
sap	Namespace	
query_view_data	Data for Query View	
rrw3_Query_View_Data	Data for a Query or a View	
SRT_TESTS_FB_ADD_WSD	Web Service for SRT_TESTS_FB_ADD	
SRT_TESTS_FB_PAR_TEST0	WS for SRT_TESTS_FB_PAR_TEST01	
SRT_TESTS_FB_PAR_TEST0	WS for SRT_TESTS_FB_PAR_TEST02	
SRT_TESTS_FB_PAR_TEST0	WS for SRT_TESTS_FB_PAR_TEST03	
SRT_TESTS_FB_SUM_WSD	Web Service for SRT_TESTS_FB_SUM	
ZWSCurrencyGetList	ZWSCurrencyGetList	
ZWSExrates	Exrates GetDetail	
Z_GetFlightListWSD	WSD for GetFlightList	
wsil	WSIL Description of SOAP Runtime Registry	

When you display your serviceȇs settings you see for instance that standard authentication and Basic Authentication with Standard R/3 User are selected. For the client calling the Web Service this means that Basic Authentication is used and the SAP credentials are sent over the wire in clear text. This is one of the easiest security configurations, but SAP supports also SSO, X.509 tickets or WS-Security for handling authentication.


Create/Change a Service

ICF Path	/default_host/sap/bc/srt/rfc/sap/
ICF Object	NSP0002 Service (Active) Altern. Name: Z_GetFlightListWSD
Description in	English

WSD for GetFlightList

Service Data | Handler List | Error Pages

Logon Procedure
- [✓ Standard]
- Alternative Logon Order
- Logon Data Required
- Client Cert. (SSL) Required
- All Logons

Service Options

Server Group:	
SAP Authoriz.	Err. Type 0
Session Timeout:	00:00:00 (HH:MM:SS)
Compression (if possible)	
GUI Link	

Anonymous Logon Data

Client	
User	
Password	still initial
Language	

Security Requirements
- [✓ Standard]
- SSL

Basic Authentication
- [✓ Standard R/3 User]
- Internet User

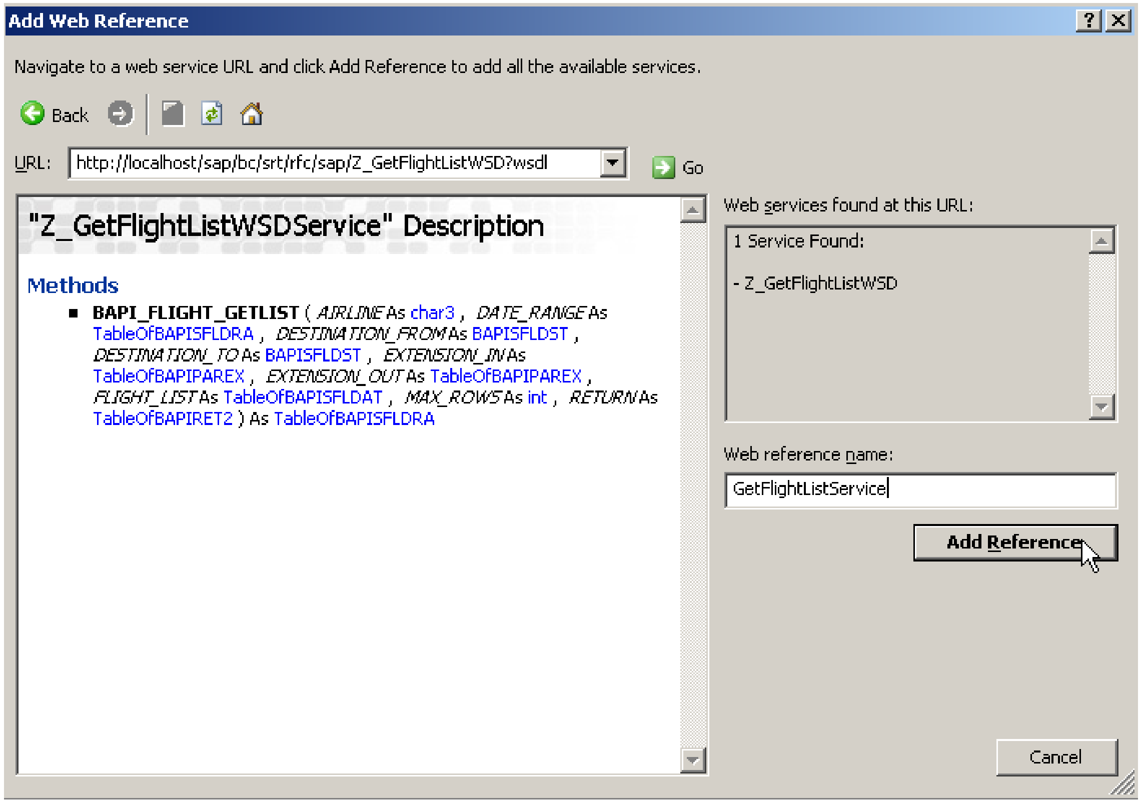
Administration

Last Changed By	BSTEINER	CreatedBy	BSTEINER
Client	000	Client	000
Changed On	14.03.2006	Created On	14.03.2006

To enable any client to call your Web Service you must provide the WSDL to the client programmer. You can do this either by downloading the WSDL from transaction WSADMIN and hand over the file or you go to the SICF transaction, navigate to your Web Service and select Test service. This opens the browser with the URL of your Web Service in the address bar, e.g. http://localhost/sap/bc/srt/rfc/sap/z_getflightlistwsd?sap-client=000. When you now replace sap-client=000 with wsdl and browse to this address (http://localhost/sap/bc/srt/rfc/sap/z_getflightlistwsd?wsdl) you can view the WSDL from remote via the browser. This way of accessing a Web Service's WSDL looks very familiar to .NET developers as you can access the WSDL of Web Service written in ASP.NET the same way.

Set up .NET Web Service Client

When you have the WSDL as a file or the address of the WSDL you can start creating the Web Service client in .NET.
First you must create a new project in Visual Studio 2005. We will use a Console Application for our samples in this whitepaper.
The next step is to make the SAP Web Service visible and callable in the project, as we do not want to create and parse the SOAP messages manually. In Visual Studio 2005 you can use the Add Web Reference wizard to create so called proxy classes for the Web Service.



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

URL: http://localhost/sap/bc/srt/rfc/sap/Z_GetFlightListWSD?wsdl

"Z_GetFlightListWSDService" Description

Methods

- BAPI_FLIGHT_GETLIST (*AIRLINE* As char3 , *DATE_RANGE* As TableOfBAPISFLDRA , *DESTINATION_FROM* As BAPISFLDST , *DESTINATION_TO* As BAPISFLDST , *EXTENSION_IN* As TableOfBAPIPAREX , *EXTENSION_OUT* As TableOfBAPIPAREX , *FLIGHT_LIST* As TableOfBAPISFLDAT , *MAX_ROWS* As int , *RETURN* As TableOfBAPIRET2) As TableOfBAPISFLDRA

Web services found at this URL:

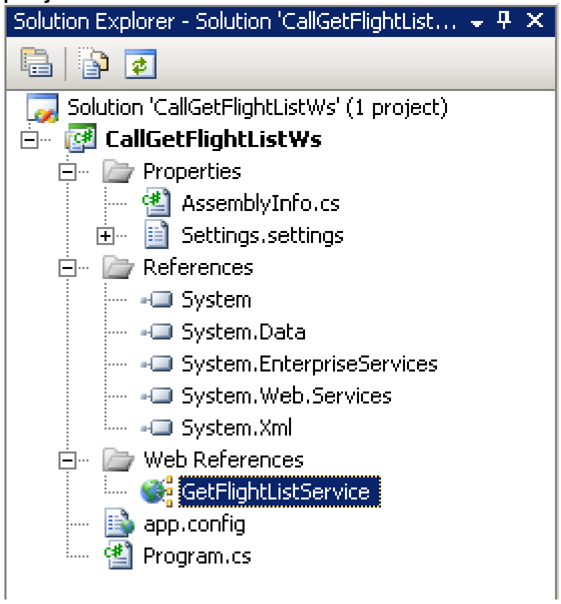
1 Service Found:

- Z_GetFlightListWSD

Web reference name: GetFlightListService

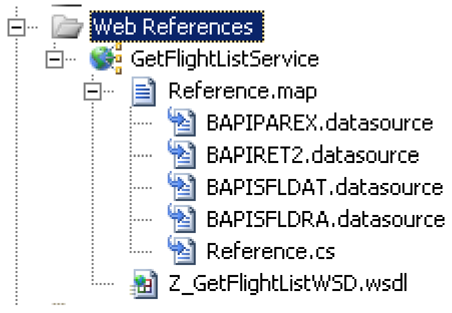
Here we used the address to the WSDL. As Basic Authentication is configured for the Web Service in SAP we had to enter a SAP user name and password to get access to the WSDL.

Executing the wizard creates the proxy classes in the Web References sub folder in the project.



```
Solution Explorer - Solution 'CallGetFlightList...
Solution 'CallGetFlightListWs' (1 project)
CallGetFlightListWs
    Properties
        AssemblyInfo.cs
        Settings.settings
    References
        System
        System.Data
        System.EnterpriseServices
        System.Web.Services
        System.Xml
    Web References
        GetFlightListService
    app.config
    Program.cs
```

To view the proxy classes you must switch on "Show all files" in the Solution Explorer by clicking the icon . Now you can see that for each SAP data type a class was created from the WSDL.



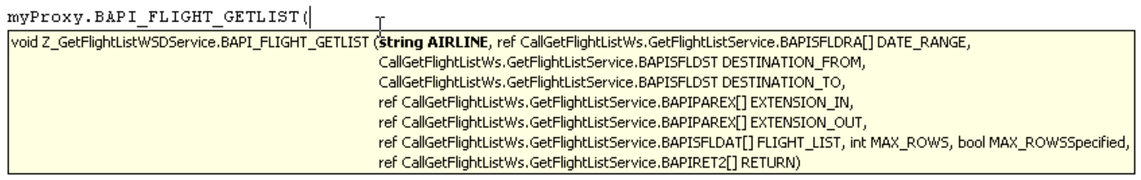
```
Web References
    GetFlightListService
        Reference.map
            BAPIPAREX.datasource
            BAPIRET2.datasource
            BAPISFLDAT.datasource
            BAPISFLDRA.datasource
            Reference.cs
        Z_GetFlightListWSD.wsdl
```

Another way to create the proxy classes is to use the wsdl.exe tool that comes with Visual Studio 2005.

Having the proxy classes created we have to create a proxy object and set the credentials for the Web Service call.

```
// create a proxy object
GetFlightListService.Z_GetFlightListService myProxy = new
GetFlightListService.Z_GetFlightListService();

// set SAP credentials
myProxy.Credentials = new System.Net.NetworkCredential("myuser",
"mypassword");
```

Up to this point calling a SAP Web Service was like calling any other Web Service. But when you look at the methods signature you see that there are a lot of parameters that are passed by reference.



```
myProxy.BAPI_FLIGHT_GETLIST(
void Z_GetFlightListWSDService.BAPI_FLIGHT_GETLIST (string AIRLINE, ref CallGetFlightListWs.GetFlightListService.BAPISFLDRA[] DATE_RANGE,
                                                    CallGetFlightListWs.GetFlightListService.BAPISFLDST DESTINATION_FROM,
                                                    CallGetFlightListWs.GetFlightListService.BAPISFLDST DESTINATION_TO,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIPAREX[] EXTENSION_IN,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIPAREX[] EXTENSION_OUT,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPISFLDAT[] FLIGHT_LIST, int MAX_ROWS, bool MAX_ROWSSpecified,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIRET2[] RETURN)
```

This is quite typical when calling SAP Web Service that base on BAPIs or remote function modules due to the SAP programming model.

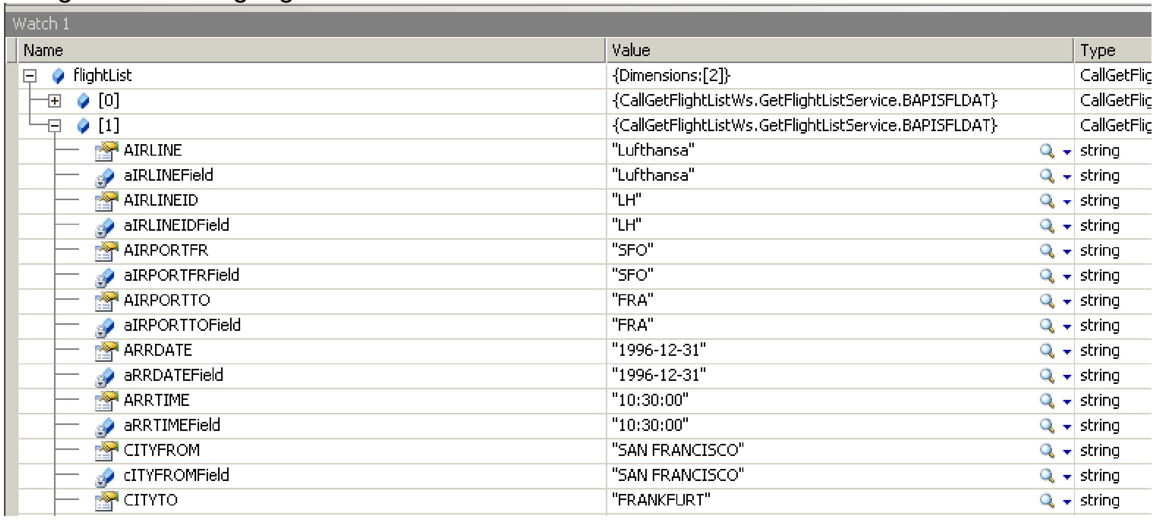
So for us this means we have to define and create each of those parameters and pass them by adding them to the method call. This can actually become very cumbersome. In one of our projects we had to do this for about 96 parameters.

```
// define parameters
GetFlightListService.BAPISFLDRA[] dateRange = new
                              GetFlightListService.BAPISFLDRA[0];
GetFlightListService.BAPISFLDST destinationFrom = new
                              GetFlightListService.BAPISFLDST();
GetFlightListService.BAPISFLDST destinationTo = new
                              GetFlightListService.BAPISFLDST();
GetFlightListService.BAPIPAREX[] extensionIn = new
                              GetFlightListService.BAPIPAREX[0];
GetFlightListService.BAPIPAREX[] extensionOut = new
                              GetFlightListService.BAPIPAREX[0];
GetFlightListService.BAPISFLDAT[] flightList = new
                              GetFlightListService.BAPISFLDAT[0];
GetFlightListService.BAPIRET2[] bapiReturn = new
                              GetFlightListService.BAPIRET2[0];

// initialize parameters
destinationFrom.AIRPORTID = "";
destinationFrom.CITY = "San Francisco";
destinationFrom.COUNTR = "";
destinationFrom.COUNTR_ISO = "";

destinationTo.AIRPORTID = "";
destinationTo.CITY = "Frankfurt";
destinationTo.COUNTR = "";
destinationTo.COUNTR_ISO = "";

// call SAP Web Service
myProxy.BAPI_FLIGHT_GETLIST("LH",
      ref dateRange,
      destinationFrom,
      destinationTo,
      ref extensionIn,
      ref extensionOut,
      ref flightList,
      20,
      true,
      ref bapiReturn);
```

Do It

Now that we have set up both ends of our scenario we can call the SAP Web Service from .NET.

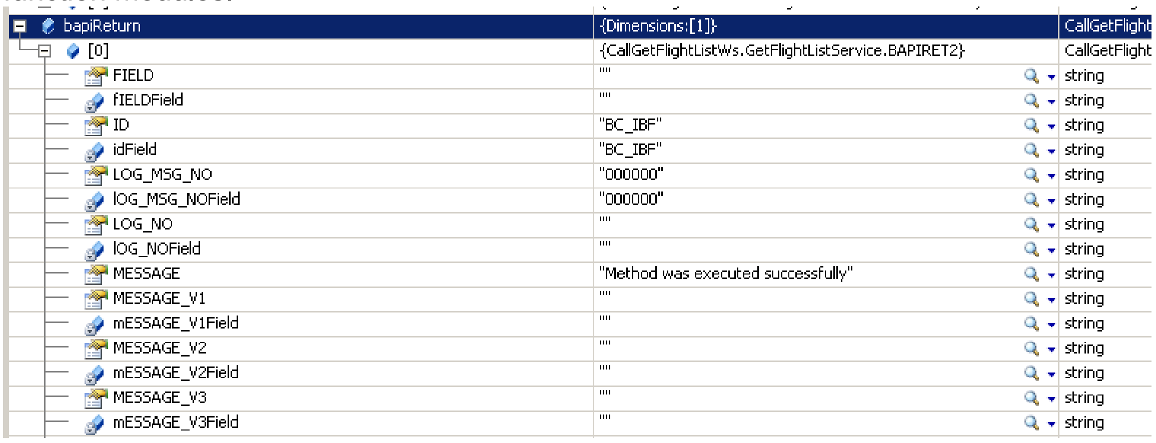
In the .NET Web Service client you can just start debugging and then look at the returned objects.
We got 2 matching flights as result.



Watch 1

Name	Value	Type
flightList	{Dimensions:[2]}	CallGetFlic
[0]	{CallGetFlightListWs.GetFlightListService.BAPISFLDAT}	CallGetFlic
[1]	{CallGetFlightListWs.GetFlightListService.BAPISFLDAT}	CallGetFlic
AIRLINE	"Lufthansa"	string
aIRLINEField	"Lufthansa"	string
AIRLINEID	"LH"	string
aIRLINEIDField	"LH"	string
AIRPORTFR	"SFO"	string
aIRPORTFRField	"SFO"	string
AIRPORTTO	"FRA"	string
aIRPORTTOField	"FRA"	string
ARRDATE	"1996-12-31"	string
aRRDATEField	"1996-12-31"	string
ARRTIME	"10:30:00"	string
aRRTIMEField	"10:30:00"	string
CITYFROM	"SAN FRANCISCO"	string
cITYFROMField	"SAN FRANCISCO"	string
CITYTO	"FRANKFURT"	string

We also got a BAPIRET2 object. This is object contains very important information for the calling application when working with SAP Web Services based on BAPIs or remote function modules.



Name	Value	Type
bapiReturn	{Dimensions:[1]}	CallGetFlight
[0]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFlight
FIELD	""	string
fIELDField	""	string
ID	"BC_IBF"	string
idField	"BC_IBF"	string
LOG_MSG_NO	"000000"	string
lOG_MSG_NOField	"000000"	string
LOG_NO	""	string
lOG_NOField	""	string
MESSAGE	"Method was executed successfully"	string
MESSAGE_V1	""	string
mESSAGE_V1Field	""	string
MESSAGE_V2	""	string
mESSAGE_V2Field	""	string
MESSAGE_V3	""	string
mESSAGE_V3Field	""	string

In this case it tells us that everything worked fine, but as you will see in the next chapter SAP uses this kind of objects to tell you also about errors.

So everything worked fine. Sure, we coded this sample around the common issues.
If we just wouldnôt set the destinationFrom value we wonôt get any results, but why? How you can find out the reason we will explain in the chapter ôSolve Issuesô

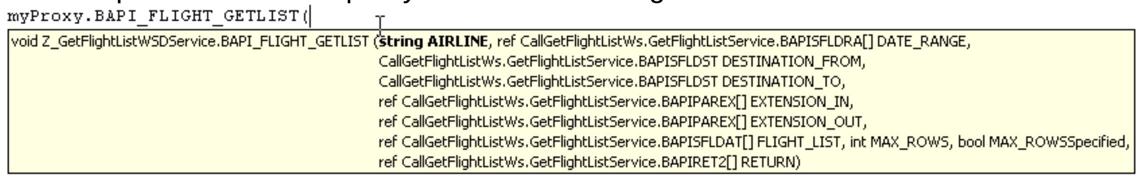
Some Particularities with SAP Web Services

XxxSpecified

Now have a look at the request message in the WSDL.

```
<xsd:element name="BAPI_FLIGHT_GETLIST">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element minOccurs="0" name="AIRLINE" type="tns:char3" />
            <xsd:element minOccurs="0" name="DATE_RANGE"
                                       type="tns:TableOfBAPISFLDRA" />
            <xsd:element minOccurs="0" name="DESTINATION_FROM"
                                       type="tns:BAPISFLDST" />
            <xsd:element minOccurs="0" name="DESTINATION_TO"
                                       type="tns:BAPISFLDST" />
            <xsd:element minOccurs="0" name="EXTENSION_IN"
                                       type="tns:TableOfBAPIPAREX" />
            <xsd:element minOccurs="0" name="EXTENSION_OUT"
                                       type="tns:TableOfBAPIPAREX" />
            <xsd:element minOccurs="0" name="FLIGHT_LIST"
                                       type="tns:TableOfBAPISFLDAT" />
            <xsd:element minOccurs="0" name="MAX_ROWS"
                                       type="xsd:int" />
            <xsd:element minOccurs="0" name="RETURN"
                                       type="tns:TableOfBAPIRET2" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

And compare this with the proxy classôs method signature.



```
myProxy.BAPI_FLIGHT_GETLIST(
void Z_GetFlightListWSDService.BAPI_FLIGHT_GETLIST (string AIRLINE, ref CallGetFlightListWs.GetFlightListService.BAPISFLDRA[] DATE_RANGE,
                                                    CallGetFlightListWs.GetFlightListService.BAPISFLDST DESTINATION_FROM,
                                                    CallGetFlightListWs.GetFlightListService.BAPISFLDST DESTINATION_TO,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIPAREX[] EXTENSION_IN,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIPAREX[] EXTENSION_OUT,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPISFLDAT[] FLIGHT_LIST, int MAX_ROWS, bool MAX_ROWSSpecified,
                                                    ref CallGetFlightListWs.GetFlightListService.BAPIRET2[] RETURN)
```

Can you find the MAX_ROWSpecified parameter in the WSDL?

The .NET Add Web Reference wizard creates ñXxxSpecifiedò parameters for parameters and attributes that have minOccurs=ò0ò set in the WSDL but the according data type in .NET is a value type, e.g. boolean, int, decimal or DateTime, that cannot be become null. When working with SAP Web Services we came across such attributes and parameters quite often.
By default the attributeôs value is false and this means that the according attributeôs (Xxx) value wonôt be set in the SOAP request. So to get your attribute value transferred to SAP you must explicitly set the XxxSpecified attribute to true. In the .NET Framework 1.1 XxxSpecified parameters or attributes were also created if the element was nillable, but the .NET Framework 2.0 introduced the new generic System.Nullable<T> that is used for these cases now.

XSD Data Type Restrictions

Or look at the type of the AIRLINE element ï char3, this is defined in the WSDL as simpleType of type string with a maximum length of 3.

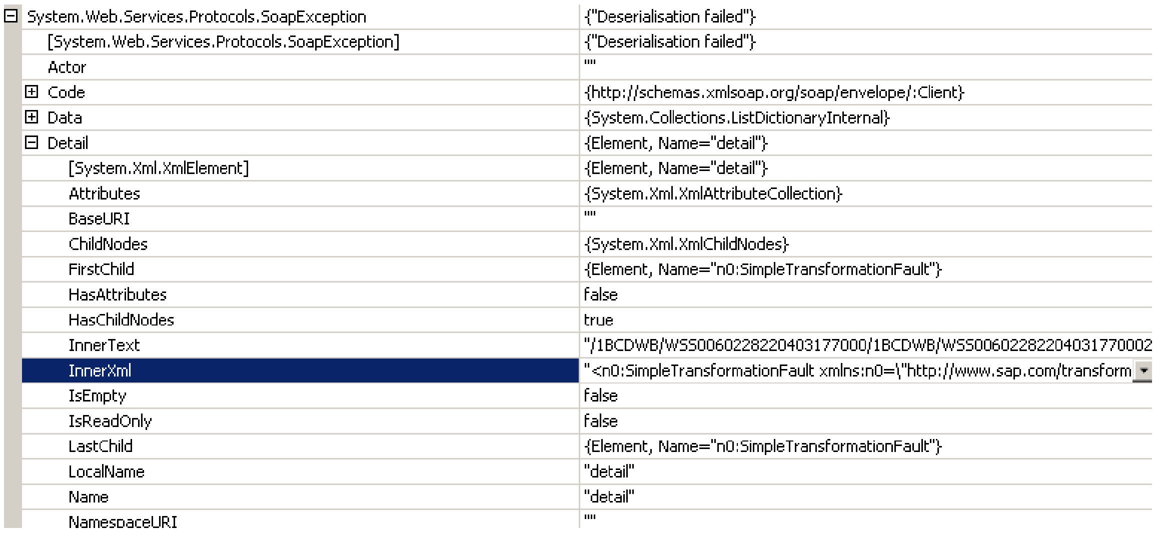
```
<xsd:simpleType name="char3">
    <xsd:restriction base="xsd:string">
       <xsd:maxLength value="3" />
    </xsd:restriction>
</xsd:simpleType>
```

Can you find this restriction in the methodôs signature or the proxy class? SAP uses the XSD to define the data types of the elements in the Web Service message so that they match the definition in SAP. But in the .NET Web Service these data types are not represented in that precision. The basic data type is used but without length, or pattern restrictions. This can lead to issues due to the fact that you can set values in the Web Service client that do not match the WSDL. In previous SAP WebAS version SAP tried to process those values, SAP even cut off strings that were too long instead of returning an error. Since SAP NetWeaver Application Server 6.40 SAP checks the SOAP message and returns a SOAP fault that tells you what is wrong.

For example if we use a value for the AIRLINE parameter that has more than three characters, e.g.

```
myProxy.BAPI_FLIGHT_GETLIST("Lufthansa",
                ref dateRange,
                destinationFrom,
                destinationTo,
                ref extensionIn,
                ref extensionOut,
                ref flightList,
                20,
                false,
                ref bapiReturn);
```

You will get a SoapException with the message ñDeserialisation failedò.
This doesnôt tell very detailed whatôs wrong. But SOAP faults have an element called ñDetailò for this purpose. The content of this element gives a hint about the error.



```
⊟ System.Web.Services.Protocols.SoapException    {"Deserialisation failed"}
    [System.Web.Services.Protocols.SoapException] {"Deserialisation failed"}
    Actor                                        ""
  ⊞ Code                                         {http://schemas.xmlsoap.org/soap/envelope/:Client}
  ⊞ Data                                         {System.Collections.ListDictionaryInternal}
  ⊟ Detail                                       {Element, Name="detail"}
      [System.Xml.XmlElement]                    {Element, Name="detail"}
      Attributes                                 {System.Xml.XmlAttributeCollection}
      BaseURI                                    ""
      ChildNodes                                 {System.Xml.XmlChildNodes}
      FirstChild                                 {Element, Name="n0:SimpleTransformationFault"}
      HasAttributes                              false
      HasChildNodes                              true
      InnerText                                  "/1BCDWB/WSS0060228220403177000/1BCDWB/WSS00602282204031770002
      InnerXml                                   "<n0:SimpleTransformationFault xmlns:n0=\"http://www.sap.com/transform
      IsEmpty                                    false
      IsReadOnly                                 false
      LastChild                                  {Element, Name="n0:SimpleTransformationFault"}
      LocalName                                  "detail"
      Name                                       "detail"
      NamespaceURI                               ""
```

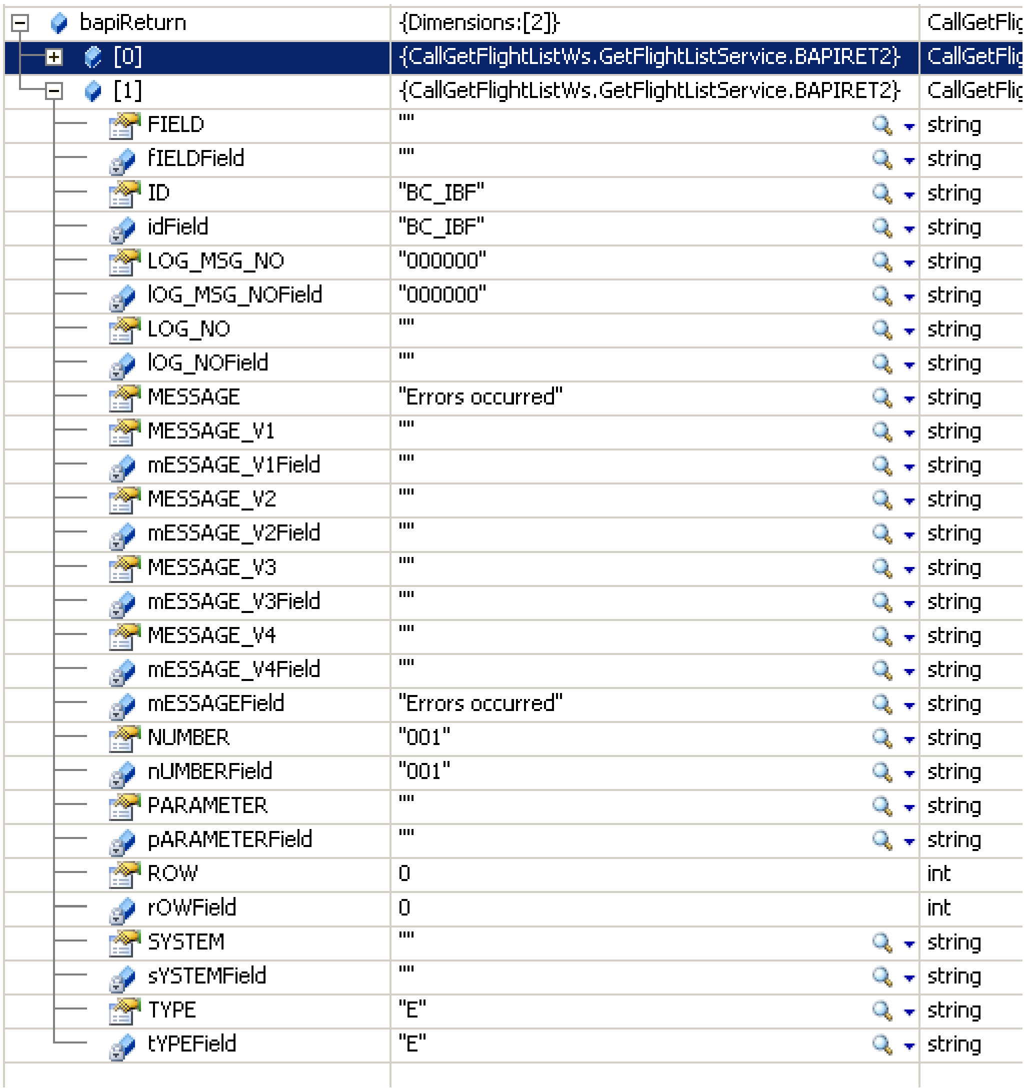
```
<n0:SimpleTransformationFault
xmlns:n0=\"http://www.sap.com/transformation-templates\">
<MainName>/1BCDWB/WSS0060228220403177000</MainName>
<ProgName>/1BCDWB/WSS0060228220403177000</ProgName>
<Line>28 </Line>
<Valid>X</Valid>
<DeserialisationFault>
<DescriptionText>An error occurred when deserializing in the simple
transformation program /1BCDWB/WSS0060228220403177000
</DescriptionText>
<DescriptionDetailText>Data loss occurred when converting
Lufthansa¬ã&lt;H—ã&lt;</DescriptionDetailText>
<TreePosition></TreePosition>
<ClassName>CX_SY_CONVERSION_DATA_LOSS</ClassName>
</DeserialisationFault>
<Caller><Class>CL_SRG_RFC_PROXY_CONTEXT</Class>
<Method>IF_SXML_PART~DECODE</Method>
<Positions>1 </Positions>
</Caller></n0:SimpleTransformationFault>
```

BAPIRET and BAPIRET2 Objects

Above we showed you how SAP reports errors that occur in the SOAP infrastructure, i.e. before the BAPI can be called internally. All errors that occur in a BAPI are returned in the parameter of type BAPIRET or BAPIRET2.
As .NET developer you should always evaluate the returned BAPIRET objects to know if the BAPI execution was really successful.

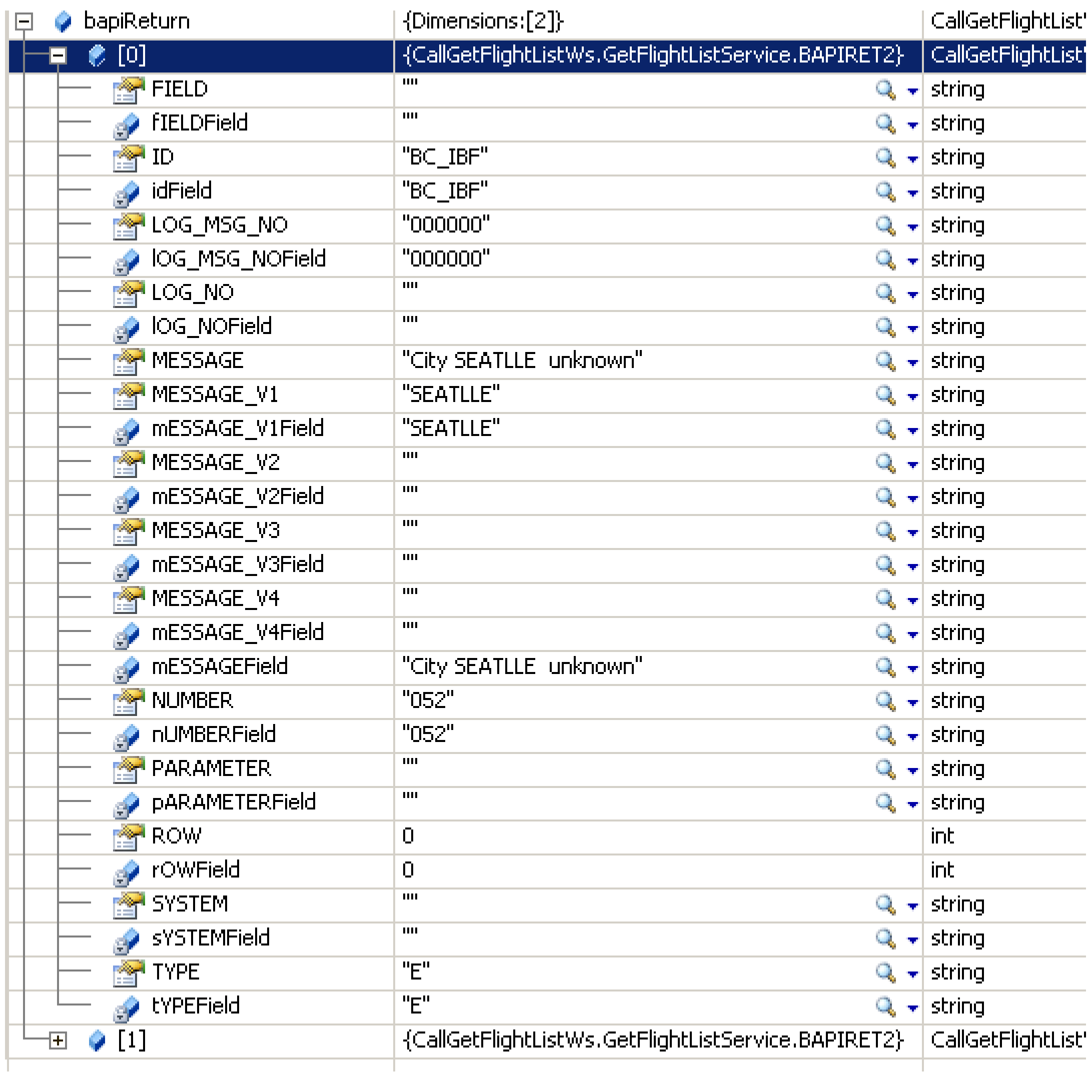
For example if you use a City from which no flights exist in the database. SAP reports an error by returning 2 BAPIRET2 objects:

One with a general error message



bapiReturn	{Dimensions:[2]}	CallGetFlig
[0]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFlig
[1]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFlig
FIELD	""	string
fIELDField	""	string
ID	"BC_IBF"	string
idField	"BC_IBF"	string
LOG_MSG_NO	"000000"	string
lOG_MSG_NOField	"000000"	string
LOG_NO	""	string
lOG_NOField	""	string
MESSAGE	"Errors occurred"	string
MESSAGE_V1	""	string
mESSAGE_V1Field	""	string
MESSAGE_V2	""	string
mESSAGE_V2Field	""	string
MESSAGE_V3	""	string
mESSAGE_V3Field	""	string
MESSAGE_V4	""	string
mESSAGE_V4Field	""	string
mESSAGEField	"Errors occurred"	string
NUMBER	"001"	string
nUMBERField	"001"	string
PARAMETER	""	string
pARAMETERField	""	string
ROW	0	int
rOWField	0	int
SYSTEM	""	string
sYSTEMField	""	string
TYPE	"E"	string
tYPEField	"E"	string

And a further object with detailed error information



bapiReturn	{Dimensions:[2]}	CallGetFlightList'
[0]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFlightList'
FIELD	""	string
fIELDField	""	string
ID	"BC_IBF"	string
idField	"BC_IBF"	string
LOG_MSG_NO	"000000"	string
lOG_MSG_NOField	"000000"	string
LOG_NO	""	string
lOG_NOField	""	string
MESSAGE	"City SEATLLE unknown"	string
MESSAGE_V1	"SEATLLE"	string
mESSAGE_V1Field	"SEATLLE"	string
MESSAGE_V2	""	string
mESSAGE_V2Field	""	string
MESSAGE_V3	""	string
mESSAGE_V3Field	""	string
MESSAGE_V4	""	string
mESSAGE_V4Field	""	string
mESSAGEField	"City SEATLLE unknown"	string
NUMBER	"052"	string
nUMBERField	"052"	string
PARAMETER	""	string
pARAMETERField	""	string
ROW	0	int
rOWField	0	int
SYSTEM	""	string
sYSTEMField	""	string
TYPE	"E"	string
tYPEField	"E"	string
[1]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFlightList'

In the .NET client you should check the TYPE attribute of the BAPIRET2 object for the value E that indicates that an error occurred.

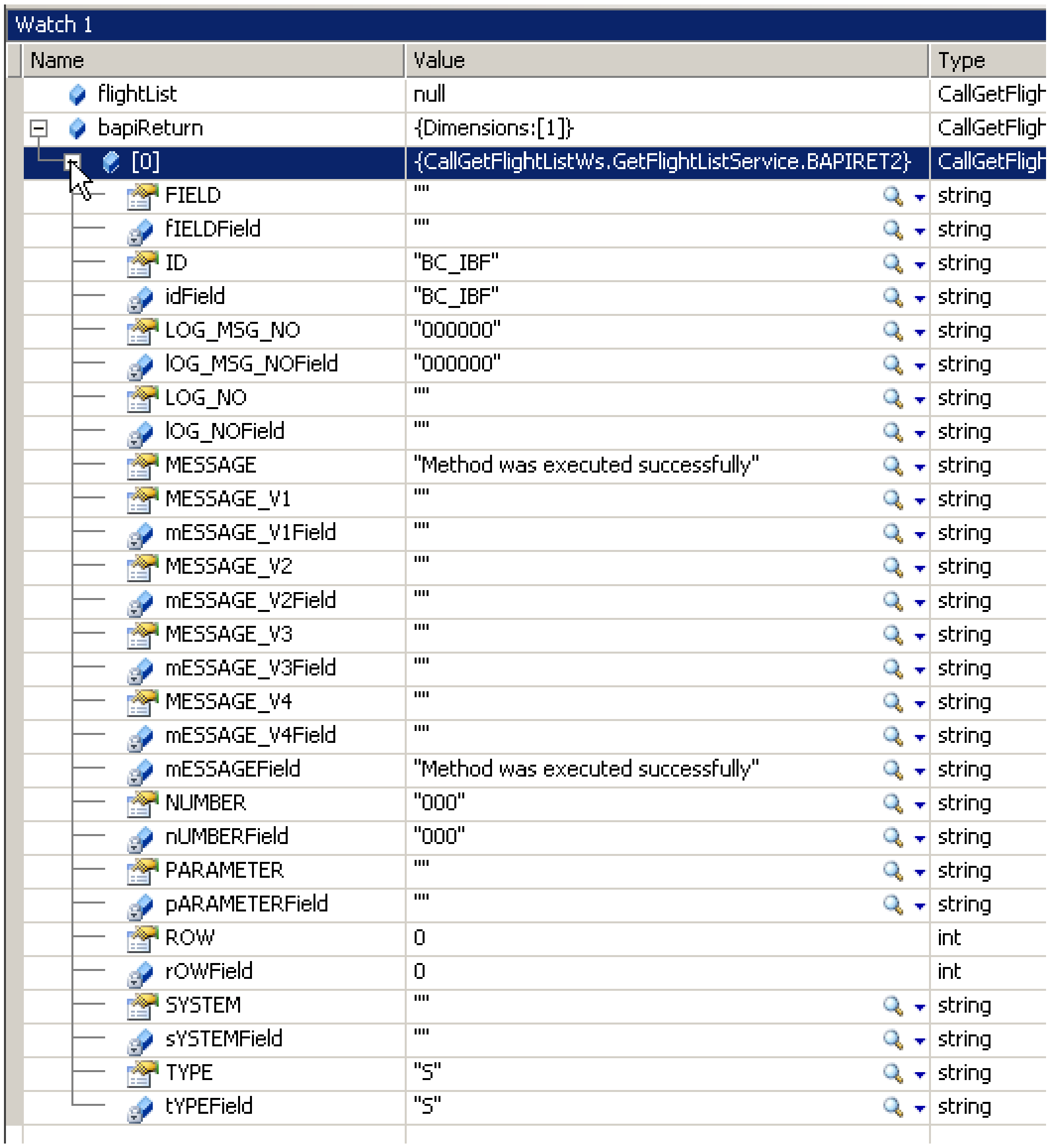
Call By Reference Array Parameters

As you might have noticed we initialized all array parameters that are passed by reference with an object array with dimension 0. Actually it does not matter what dimension you set, but it is important that you initialize the array otherwise the parameter won't be filled.

It is quite easy to test this. We use the client as in our successful example but we do not initialize the flightList array.

```
// define and initialize parameters
GetFlightListService.BAPISFLDRA[] dateRange = new
      GetFlightListService.BAPISFLDRA[0];
GetFlightListService.BAPISFLDST destinationFrom = new
      GetFlightListService.BAPISFLDST();
destinationFrom.AIRPORTID = "";
      destinationFrom.CITY = "San Francisco";
destinationFrom.COUNTR = "";
destinationFrom.COUNTR_ISO = "";
GetFlightListService.BAPISFLDST destinationTo = new
      GetFlightListService.BAPISFLDST();
destinationTo.AIRPORTID = "";
destinationTo.CITY = "Frankfurt";
destinationTo.COUNTR = "";
destinationTo.COUNTR_ISO = "";
GetFlightListService.BAPIPAREX[] extensionIn = new
      GetFlightListService.BAPIPAREX[0];
GetFlightListService.BAPIPAREX[] extensionOut = new
      GetFlightListService.BAPIPAREX[0];
GetFlightListService.BAPISFLDAT[] flightList = null;
GetFlightListService.BAPIRET2[] bapiReturn = new
      GetFlightListService.BAPIRET2[0];

// call SAP Web Service
myProxy.BAPI_FLIGHT_GETLIST("LH",
      ref dateRange,
      destinationFrom,
      destinationTo,
      ref extensionIn,
      ref extensionOut,
      ref flightList,
      20,
      false,
      ref bapiReturn);
```

The result of this call is a success message in the BAPIRET2 object but an empty flight list though we got two flights from San Francisco to Frankfurt when we used an initialized flightList array.


Watch 1

Name	Value	Type
flightList	null	CallGetFligh
bapiReturn	{Dimensions:[1]}	CallGetFligh
[0]	{CallGetFlightListWs.GetFlightListService.BAPIRET2}	CallGetFligh
FIELD	""	string
fIELDField	""	string
ID	"BC_IBF"	string
idField	"BC_IBF"	string
LOG_MSG_NO	"000000"	string
lOG_MSG_NOField	"000000"	string
LOG_NO	""	string
lOG_NOField	""	string
MESSAGE	"Method was executed successfully"	string
MESSAGE_V1	""	string
mESSAGE_V1Field	""	string
MESSAGE_V2	""	string
mESSAGE_V2Field	""	string
MESSAGE_V3	""	string
mESSAGE_V3Field	""	string
MESSAGE_V4	""	string
mESSAGE_V4Field	""	string
mESSAGEField	"Method was executed successfully"	string
NUMBER	"000"	string
nUMBERField	"000"	string
PARAMETER	""	string
pARAMETERField	""	string
ROW	0	int
rOWField	0	int
SYSTEM	""	string
sYSTEMField	""	string
TYPE	"S"	string
tYPEField	"S"	string

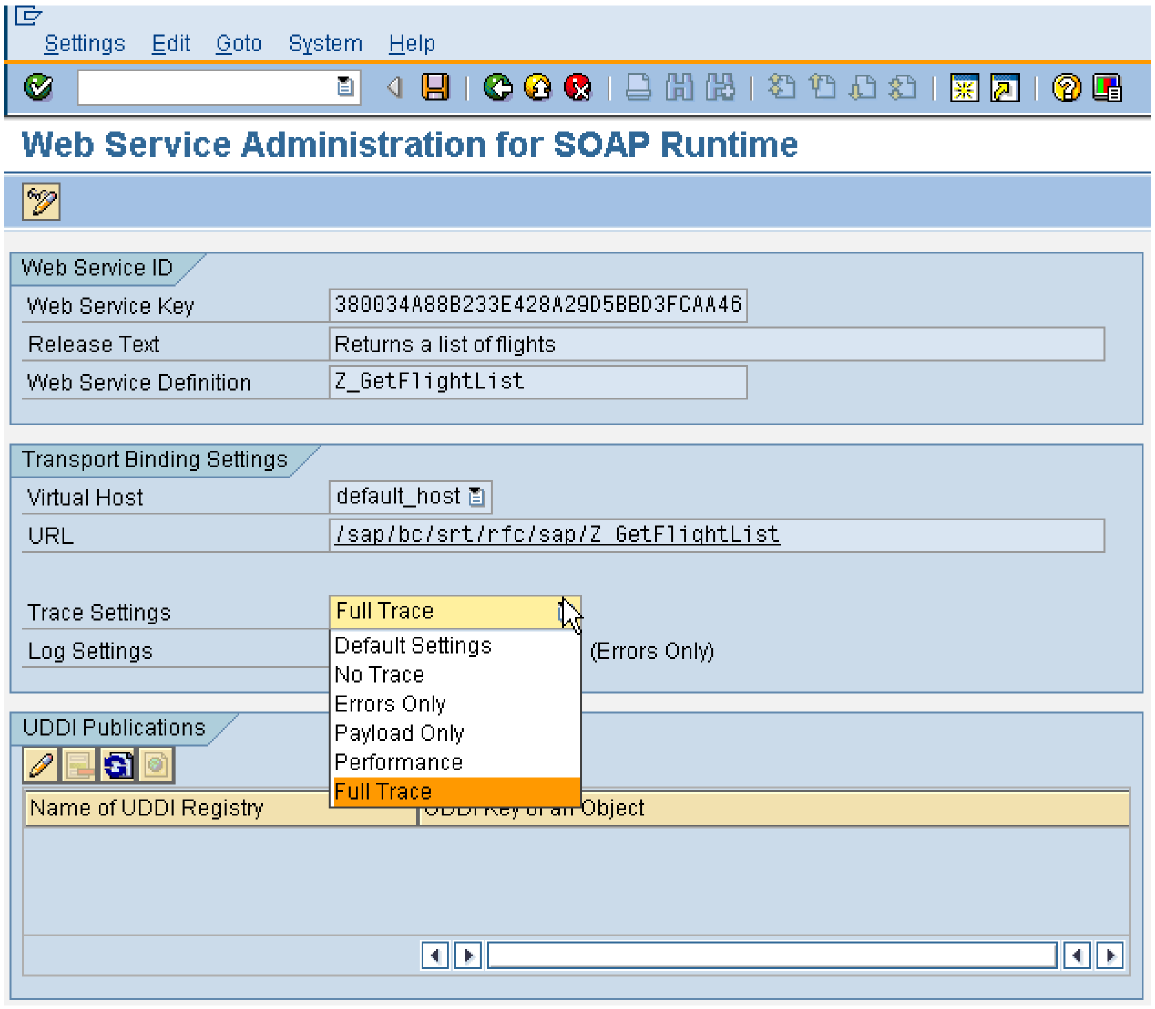
This is the point where it is interesting to have a closer look at the exchanged messages to find out if SAP does not fill the flight list or if .NET is not able to map the returned flights into the flightList object.

Tracing

Tracing is a very good way to troubleshoot problems in the Web Service communication as you can see the raw SOAP messages that are sent over the wire.

In SAP

For analyzing the above case we can just switch on tracing on the Web Service in SAP in the transaction WSADMIN.



Web Service Administration for SOAP Runtime

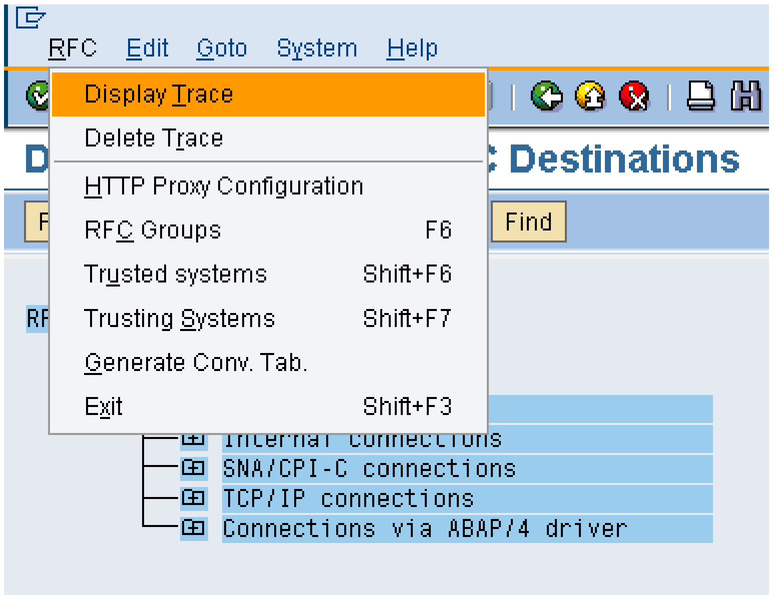
Web Service ID

Web Service Key	380034A88B233E428A29D5BBD3FCAA46
Release Text	Returns a list of flights
Web Service Definition	Z_GetFlightList

Transport Binding Settings

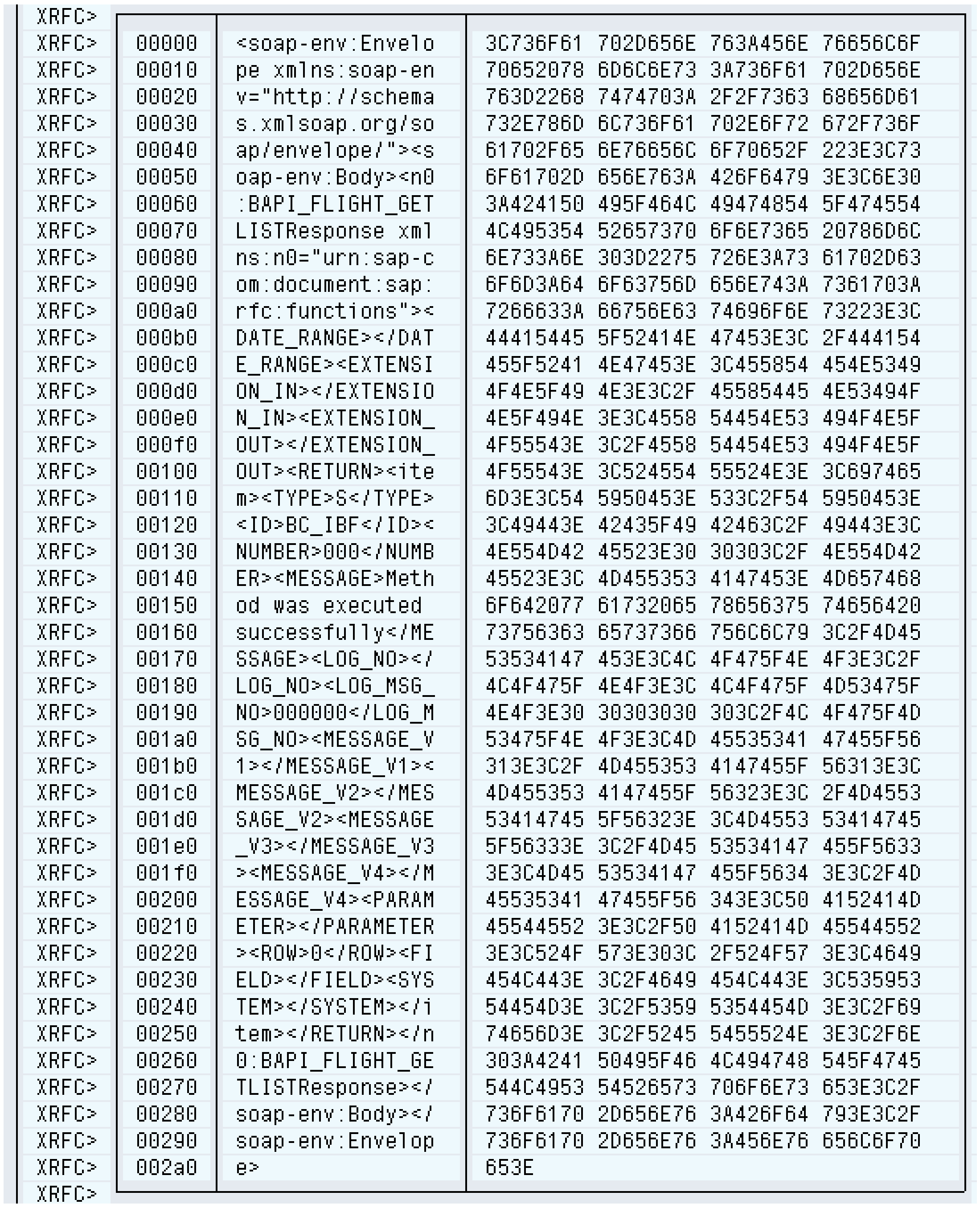
Virtual Host	default_host
URL	/sap/bc/srt/rfc/sap/Z_GetFlightList
Trace Settings	Full Trace
Log Settings	(Errors Only)

So when we execute the call with the not-initialized flightList object we can then view the exchanged SOAP message in the transaction SM59.



Microsoft softlab

At the bottom of the trace file we find the SOAP response. And there we see that SAP does not return an element called FLIGHT_LIST. This would contain the data for the flightList object in .NET.



```
XRFC>
XRFC>   00000   <soap-env:Envelo   3C736F61 702D656E 763A456E 76656C6F
XRFC>   00010   pe xmlns:soap-en   70652078 6D6C6E73 3A736F61 702D656E
XRFC>   00020   v="http://schema   763D2268 7474703A 2F2F7363 68656D61
XRFC>   00030   s.xmlsoap.org/so   732E786D 6C736F61 702E6F72 672F736F
XRFC>   00040   ap/envelope/"><s   61702F65 6E76656C 6F70652F 223E3C73
XRFC>   00050   oap-env:Body><n0   6F61702D 656E763A 426F6479 3E3C6E30
XRFC>   00060   :BAPI_FLIGHT_GET   3A424150 495F464C 49474854 5F474554
XRFC>   00070   LISTResponse xml   4C495354 52657370 6F6E7365 20786D6C
XRFC>   00080   ns:n0="urn:sap-c   6E733A6E 303D2275 726E3A73 61702D63
XRFC>   00090   om:document:sap:   6F6D3A64 6F63756D 656E743A 7361703A
XRFC>   000a0   rfc:functions"><   7266633A 66756E63 74696F6E 73223E3C
XRFC>   000b0   DATE_RANGE></DAT   44415445 5F52414E 47453E3C 2F444154
XRFC>   000c0   E_RANGE><EXTENSI   455F5241 4E47453E 3C455854 454E5349
XRFC>   000d0   ON_IN></EXTENSIO   4F4E5F49 4E3E3C2F 45585445 4E53494F
XRFC>   000e0   N_IN><EXTENSION_   4E5F494E 3E3C4558 54454E53 494F4E5F
XRFC>   000f0   OUT></EXTENSION_   4F55543E 3C2F4558 54454E53 494F4E5F
XRFC>   00100   OUT><RETURN><ite   4F55543E 3C524554 55524E3E 3C697465
XRFC>   00110   m><TYPE>S</TYPE>   6D3E3C54 5950453E 533C2F54 5950453E
XRFC>   00120   <ID>BC_IBF</ID><   3C49443E 42435F49 42463C2F 49443E3C
XRFC>   00130   NUMBER>000</NUMB   4E554D42 45523E30 30303C2F 4E554D42
XRFC>   00140   ER><MESSAGE>Meth   45523E3C 4D455353 4147453E 4D657468
XRFC>   00150   od was executed    6F642077 61732065 78656375 74656420
XRFC>   00160   successfully</ME   73756363 65737366 756C6C79 3C2F4D45
XRFC>   00170   SSAGE><LOG_NO></   53534147 453E3C4C 4F475F4E 4F3E3C2F
XRFC>   00180   LOG_NO><LOG_MSG_   4C4F475F 4E4F3E3C 4C4F475F 4D53475F
XRFC>   00190   NO>000000</LOG_M   4E4F3E30 30303030 303C2F4C 4F475F4D
XRFC>   001a0   SG_NO><MESSAGE_V   53475F4E 4F3E3C4D 45535341 47455F56
XRFC>   001b0   1></MESSAGE_V1><   313E3C2F 4D455353 4147455F 56313E3C
XRFC>   001c0   MESSAGE_V2></MES   4D455353 4147455F 56323E3C 2F4D4553
XRFC>   001d0   SAGE_V2><MESSAGE   53414745 5F56323E 3C4D4553 53414745
XRFC>   001e0   _V3></MESSAGE_V3   5F56333E 3C2F4D45 53534147 455F5633
XRFC>   001f0   ><MESSAGE_V4></M   3E3C4D45 53534147 455F5634 3E3C2F4D
XRFC>   00200   ESSAGE_V4><PARAM   45535341 47455F56 343E3C50 4152414D
XRFC>   00210   ETER></PARAMETER   45544552 3E3C2F50 4152414D 45544552
XRFC>   00220   ><ROW>0</ROW><FI   3E3C524F 573E303C 2F524F57 3E3C4649
XRFC>   00230   ELD></FIELD><SYS   454C443E 3C2F4649 454C443E 3C535953
XRFC>   00240   TEM></SYSTEM></i   54454D3E 3C2F5359 5354454D 3E3C2F69
XRFC>   00250   tem></RETURN></n   74656D3E 3C2F5245 5455524E 3E3C2F6E
XRFC>   00260   0:BAPI_FLIGHT_GE   303A4241 50495F46 4C494748 545F4745
XRFC>   00270   TLISTResponse></   544C4953 54526573 706F6E73 653E3C2F
XRFC>   00280   soap-env:Body></   736F6170 2D656E76 3A426F64 793E3C2F
XRFC>   00290   soap-env:Envelop   736F6170 2D656E76 3A456E76 656C6F70
XRFC>   002a0   e>                 653E
XRFC>
```

On the Microsoft Client

Sure it is also possible to trace the communication on the Microsoft client. But in contrast to SAP this functionality is not a built-in feature of the development environment. For this job we recommend to use Fiddler. This is an http tracing tool you can download for free from www.fiddlertool.com.

For tracing with fiddler you must set the Proxy attribute of the Web Service to address on that Fiddler is listening. This is usually localhost:8888.

```
myProxy.Proxy = new System.Net.WebProxy("localhost:8888");
```

Additionally you must modify the Reference.cs file and overwrite the method GetWebRequest to set the request KeepAlive attribute to false.

```
protected override System.Net.WebRequest GetWebRequest(Uri uri)
  {
   System.Net.HttpWebRequest webRequest =
    (System.Net.HttpWebRequest) base.GetWebRequest(uri);
   webRequest.KeepAlive = false;
   return webRequest;
  }
```

Without this modification you will encounter a communication problem.

Conclusion

In this whitepaper we showed you that it is not a big issue to establish a Web Service communication between .NET and SAP. But we really only showed the basics and you should be aware that for usage in a production environment it might be worth regarding other Web Service features or standards supported by SAP and .NET, e.g. WS-Security.

Step-by-Step Guide

In this section we will walk you through the creation of a SAP Web Service and a .NET client that will call the SAP Web Service. We will create a Web Service for the BAPI BAPI_FLIGHT_GETLIST. This is a BAPI from the standard SAP FLIGHT sample that is available on every SAP system and comes with some functionality and data in the database.
The .NET sample code is available for download on https://secure.softlab.de/fm/243/SAPWSFromVS2005_SampleCode.zip.

Prerequisites

Software

- SAP
  - Netweaver 04, e.g. Mini SAP installed on your local machine or on a server on the network.
  - SAP Logon 640 on your local machine

- .NET
  - .NET Framework 2.0 on your local machine
  - Visual Studio 2005 on your local machine

Permissions

In SAP your user must be at least assigned to the standard role for Web Service developers on the ABAP stack SAP_BC_WEBSERVICE_ADMIN. For executing a Web

Service your SAP login must have the authorization object S_SERVICE assigned. Additionally you need the permission S_FLBOOK for calling the BAPI_FLIGHT_GETLIST. When you are executing the steps of this guide on a test or development SAP system the easiest setup is to assign your user to the SAP_ALL and SAP_NEW profile. If this is not possible ask someone familiar with SAP permissions to assign the above mentioned permissions to you user.

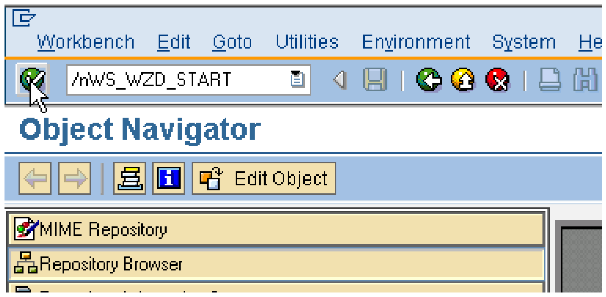
SAP Web Service

Follow the steps below to create a SAP Web Service for BAPI_FLIGHT_GETLIST.

1. Login to SAP.

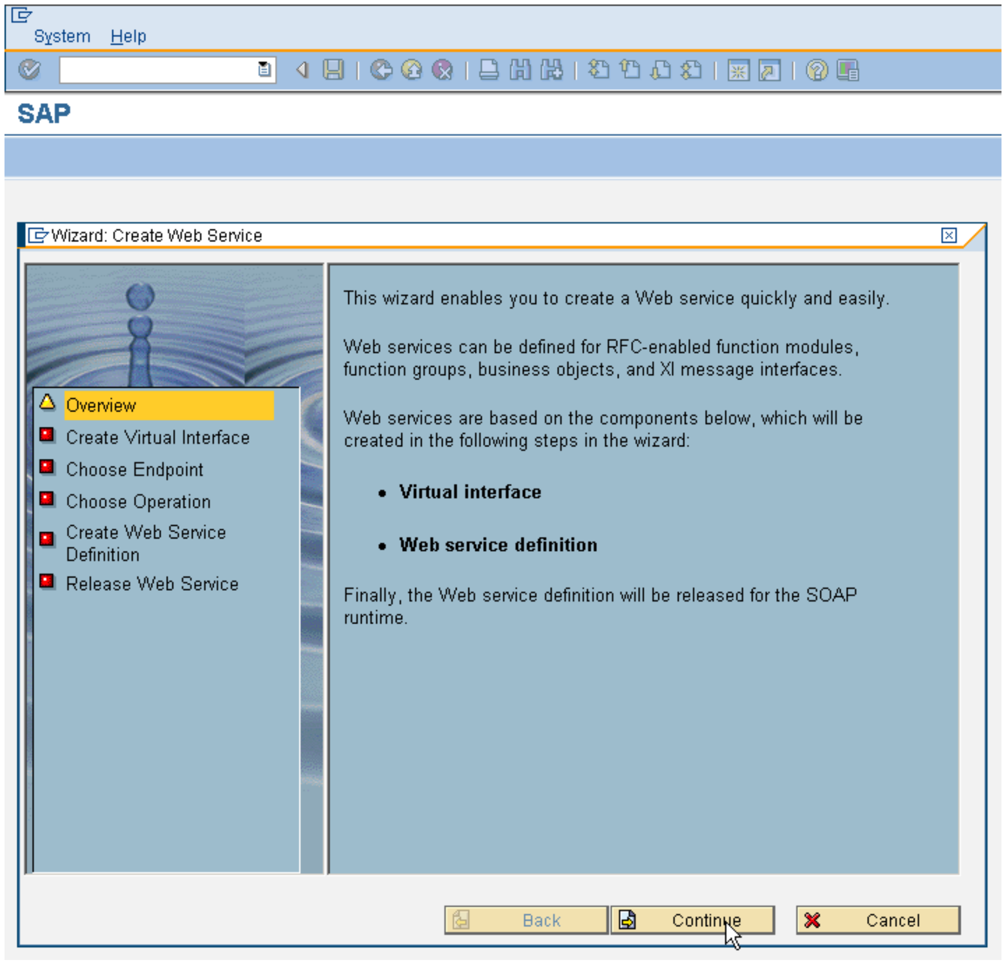
2. If you are not sure if you have package that you can use to store the objects follow the instructions in Check for Package.

3. Enter /nWS_WZD_START into the transaction code field to start the Web Service Creation Wizard.



/nWS_WZD_START

4. Click *Continue*.



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

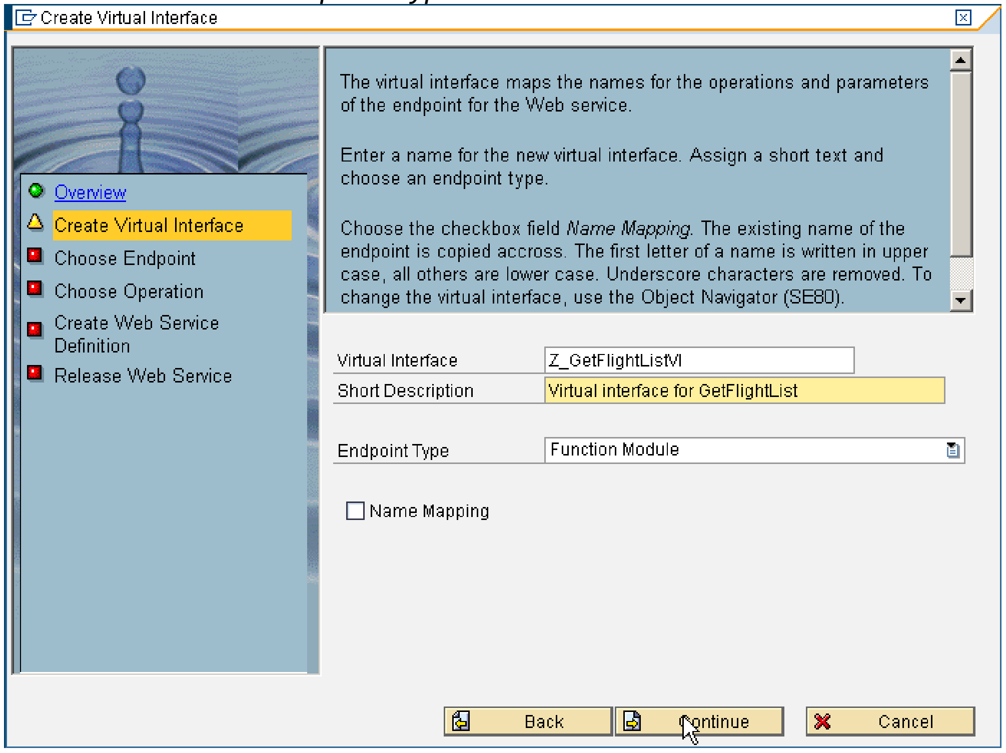
Web services can be defined for RFC-enabled function modules, function groups, business objects, and XI message interfaces.

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.

5. Enter Z_GetFlightListVI as Virtual Interface name and a short description. Select *Function Module* as *Endpoint Type* and click *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.

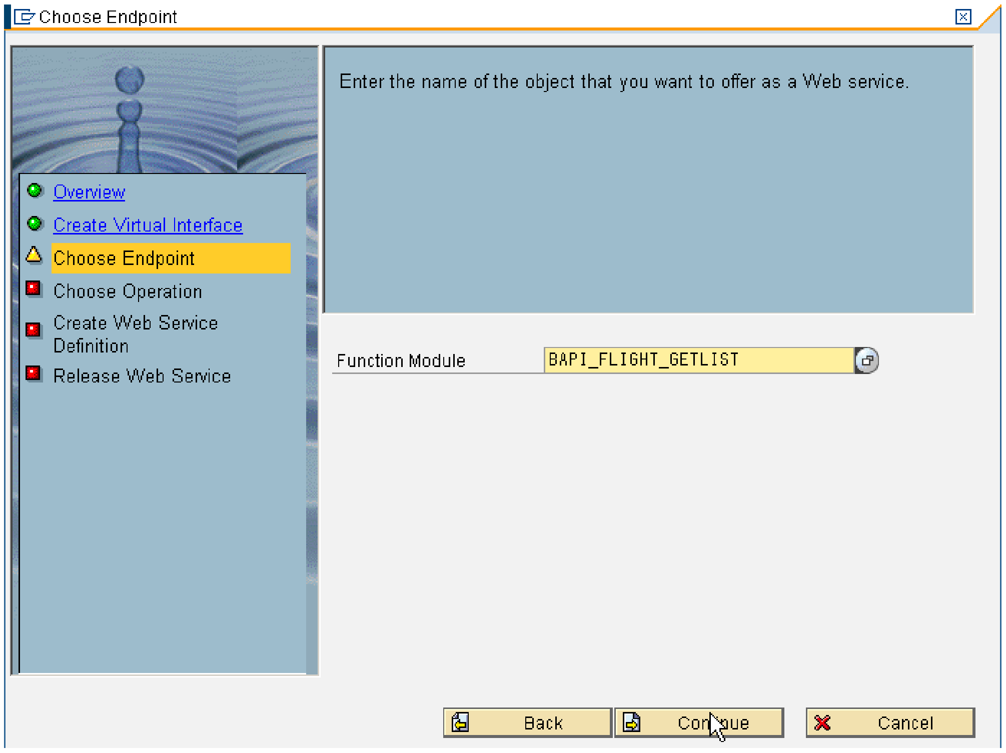
Choose the checkbox field *Name Mapping*. The existing name of the endpoint is copied accross. The first letter of a name is written in upper case, all others are lower case. Underscore characters are removed. To change the virtual interface, use the Object Navigator (SE80).

Virtual Interface	Z_GetFlightListVI
Short Description	Virtual interface for GetFlightList
Endpoint Type	Function Module

☐ Name Mapping

Click Continue.

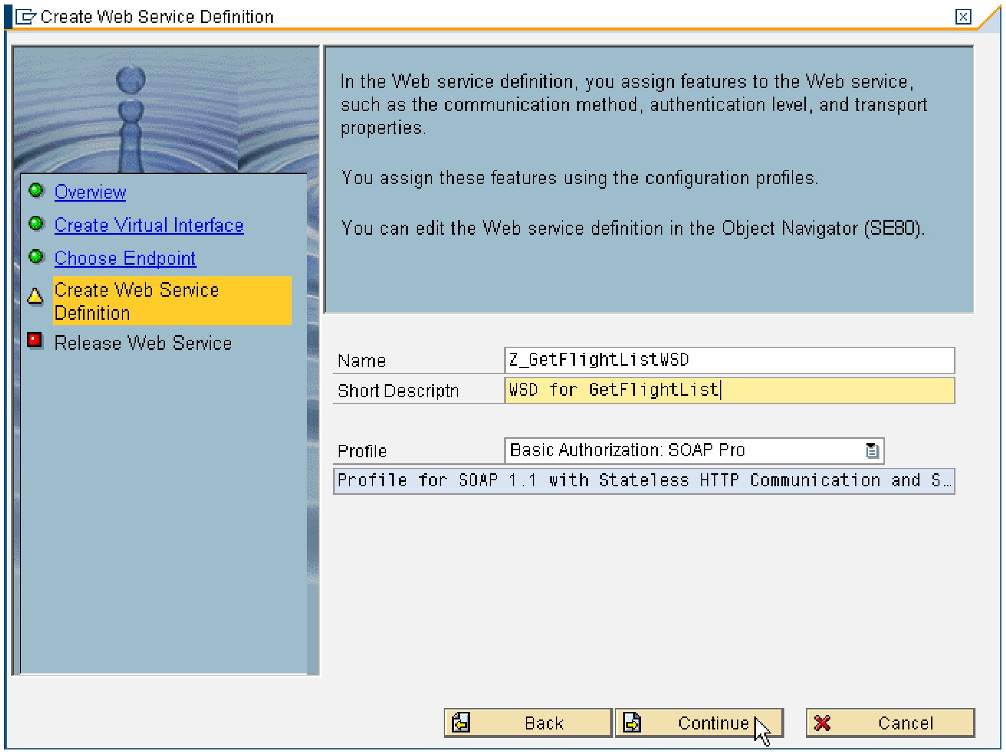
6. Enter BAPI_FLIGHT_GETLIST into the *Function Module* text box. Then click *Continue*.



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

Function Module BAPI_FLIGHT_GETLIST

7. Enter Z_GetFlightListWSD as Web Service definition name and a short description. Select *Basic Authentication: SOAP Pro* as *Profile* and then click *Continue*.



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

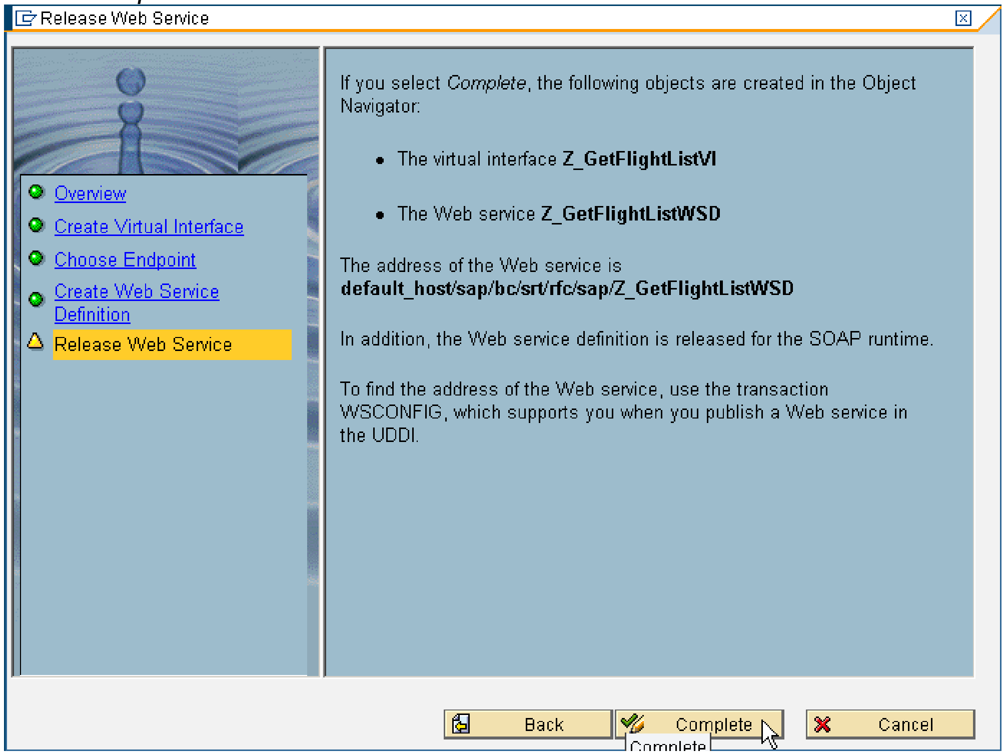
You assign these features using the configuration profiles.

You can edit the Web service definition in the Object Navigator (SE80).

Name	Z_GetFlightListWSD
Short Descriptn	WSD for GetFlightList
Profile	Basic Authorization: SOAP Pro

Profile for SOAP 1.1 with Stateless HTTP Communication and S...

8. Click *Complete*.



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

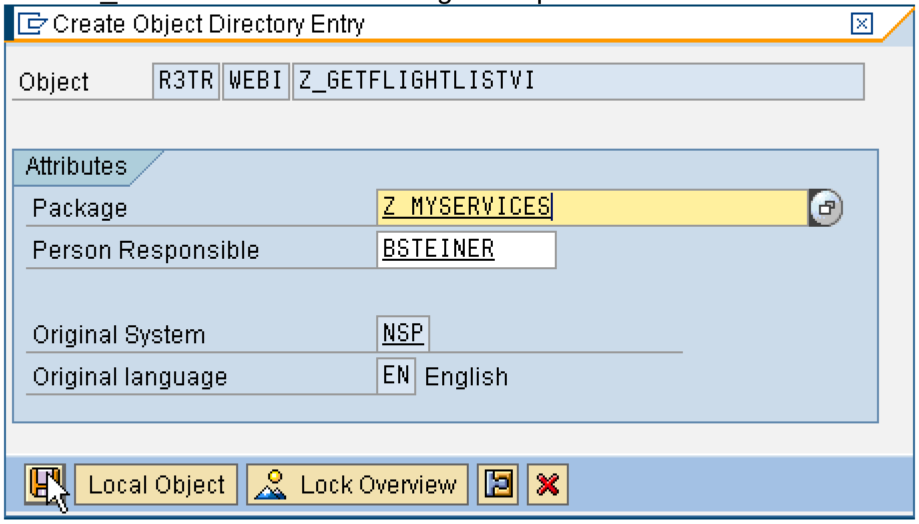
- The virtual interface **Z_GetFlightListVI**
- The Web service **Z_GetFlightListWSD**

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

In addition, the Web service definition is released for the SOAP runtime.

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

9. Enter Z_MYSERVICES as *Package* and press **Enter**.

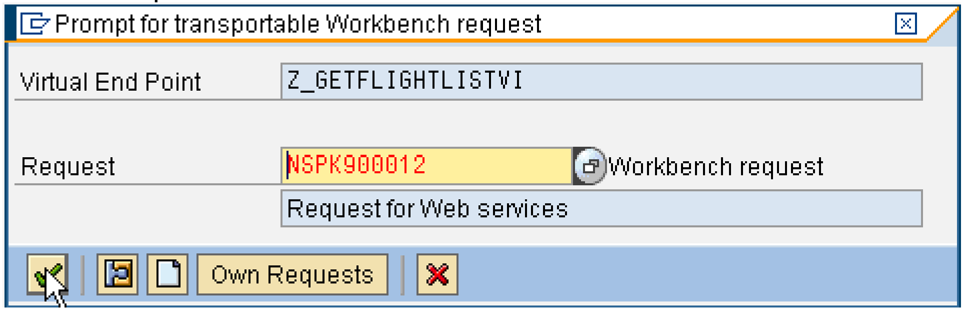


Object R3TR WEBI Z_GETFLIGHTLISTVI

Attributes

Package	Z_MYSERVICES
Person Responsible	BSTEINER
Original System	NSP
Original language	EN English

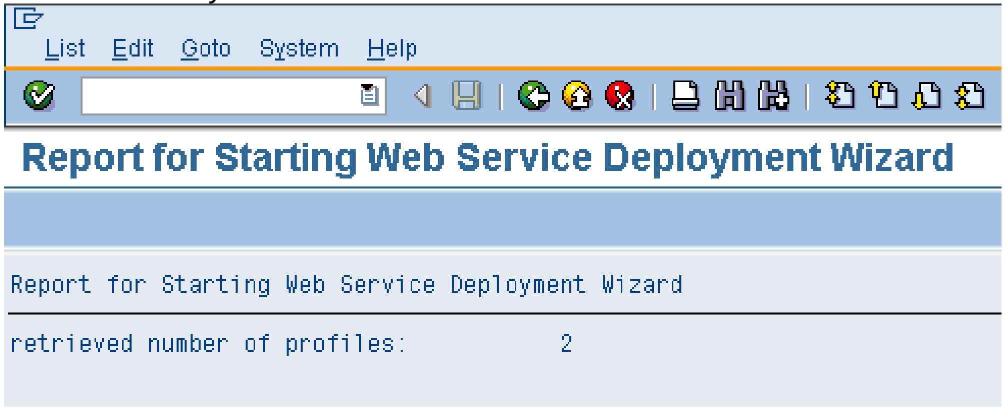
10. Enter or select your transport request and press **Enter**. If you do not have a transport request follow the instructions in section Create a Transport Request to create a request.



Virtual End Point	Z_GETFLIGHTLISTVI
Request	NSPK900012 Workbench request
	Request for Web services

11. The system will prompt you several times to specify a transport request for every single object that is being created by the wizard. By default the system will choose the same transport request for all objects. In case that no matching request exists just create a new one.

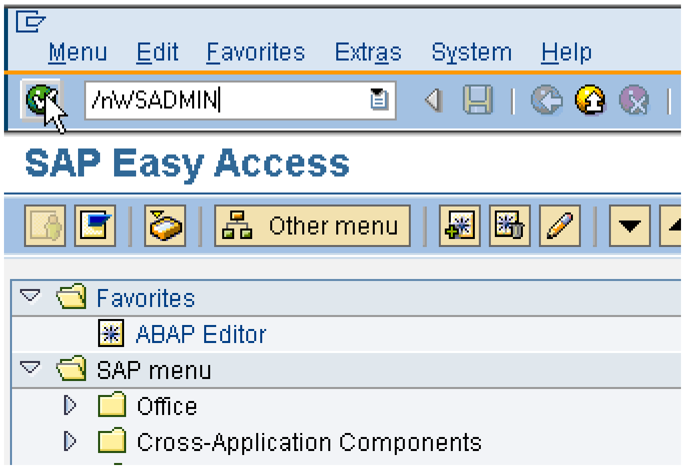
12. You successfully created a Web Service in SAP.



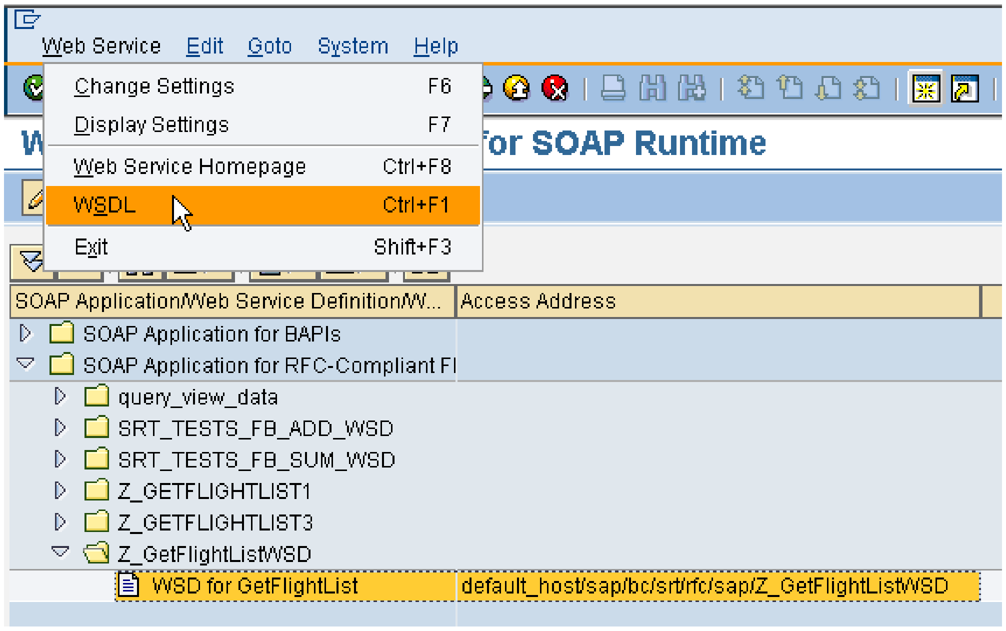
Report for Starting Web Service Deployment Wizard

```
Report for Starting Web Service Deployment Wizard

retrieved number of profiles:          2
```

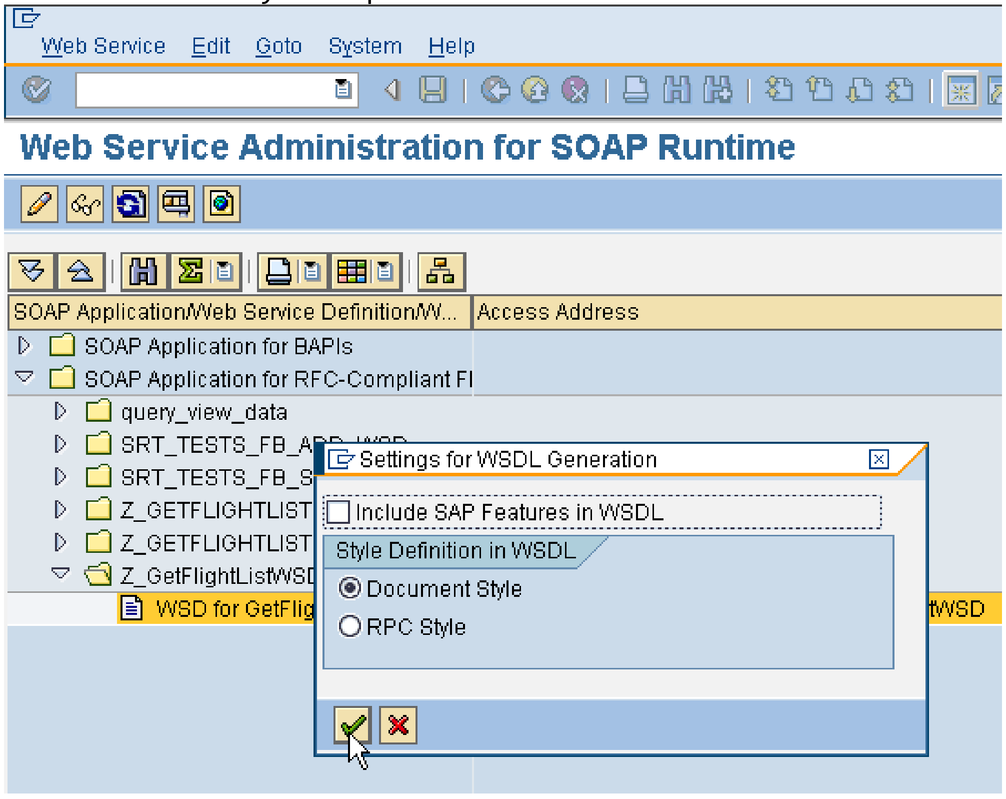
13. To view the WSDL enter /nWSADMIN in the transaction code field and press **Enter**.



14. Navigate to *SOAP Application for RFC-Compliant FMs -> Z_GetFlightListWSD* and select *WSD for GetFlightList*. Now open the Web Service menu and click *WSDL*.


Web Service Edit Goto System Help
Change Settings F6
Display Settings F7
Web Service Homepage Ctrl+F8
WSDL Ctrl+F1
Exit Shift+F3
for SOAP Runtime
SOAP Application/Web Service Definition/W... Access Address
SOAP Application for BAPIs
SOAP Application for RFC-Compliant FI
query_view_data
SRT_TESTS_FB_ADD_WSD
SRT_TESTS_FB_SUM_WSD
Z_GETFLIGHTLIST1
Z_GETFLIGHTLIST3
Z_GetFlightListWSD
WSD for GetFlightList default_host/sap/bc/srt/rfc/sap/Z_GetFlightListWSD

15. Select *Document Style* and press **Enter**.



Settings for WSDL Generation

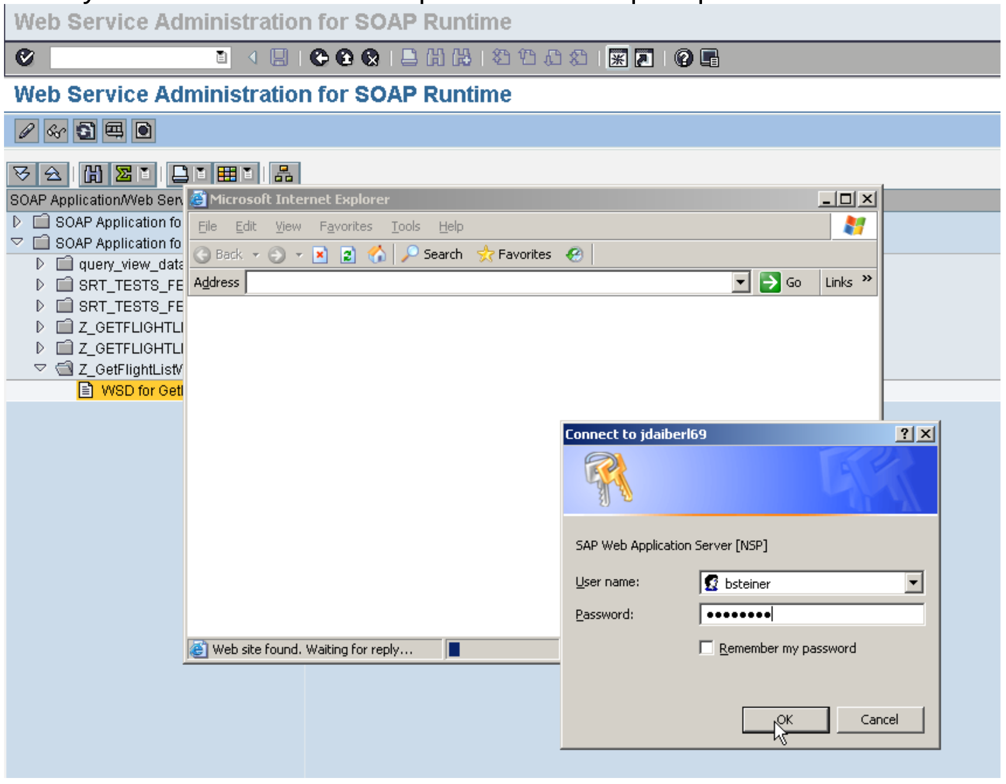
☐ Include SAP Features in WSDL

Style Definition in WSDL

● Document Style

○ RPC Style

16. Enter your SAP user name and password when prompted and click *OK*.



Connect to jdaiberl69

SAP Web Application Server [NSP]

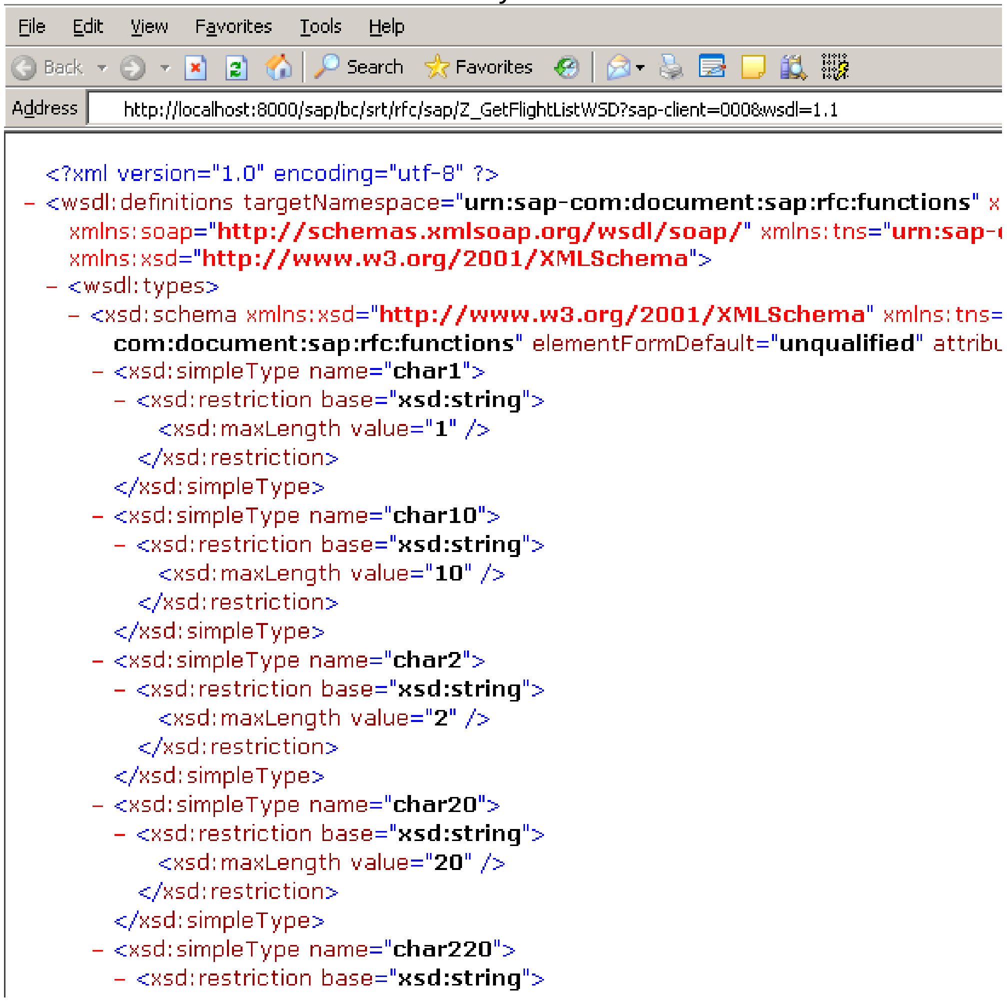
User name: bsteiner

Password: ••••••••

☐ Remember my password

OK Cancel

17. Now the browser shows the WSDL of your Web Service.



http://localhost:8000/sap/bc/srt/rfc/sap/Z_GetFlightListWSD?sap-client=000&wsdl=1.1

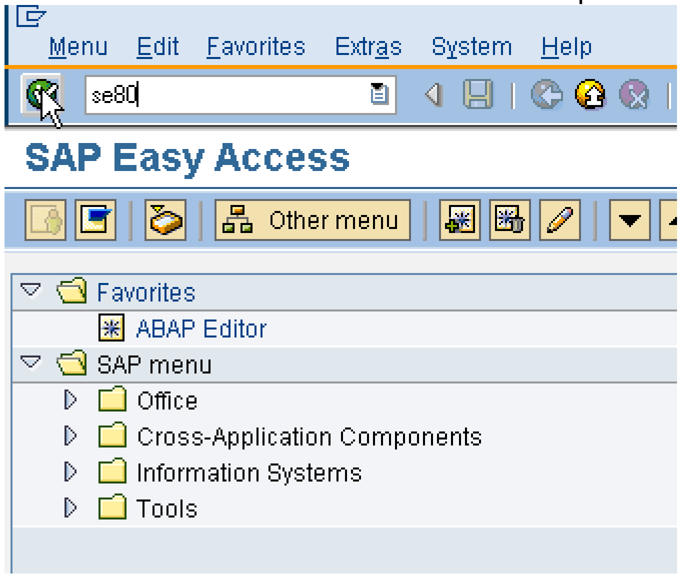
```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions targetNamespace="urn:sap-com:document:sap:rfc:functions" x
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="urn:sap-
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <wsdl:types>
    - <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns=
        com:document:sap:rfc:functions" elementFormDefault="unqualified" attribu
      - <xsd:simpleType name="char1">
        - <xsd:restriction base="xsd:string">
            <xsd:maxLength value="1" />
          </xsd:restriction>
        </xsd:simpleType>
      - <xsd:simpleType name="char10">
        - <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10" />
          </xsd:restriction>
        </xsd:simpleType>
      - <xsd:simpleType name="char2">
        - <xsd:restriction base="xsd:string">
            <xsd:maxLength value="2" />
          </xsd:restriction>
        </xsd:simpleType>
      - <xsd:simpleType name="char20">
        - <xsd:restriction base="xsd:string">
            <xsd:maxLength value="20" />
          </xsd:restriction>
        </xsd:simpleType>
      - <xsd:simpleType name="char220">
        - <xsd:restriction base="xsd:string">
```

18. Copy the URL from the address bar or save the WSDL as file.

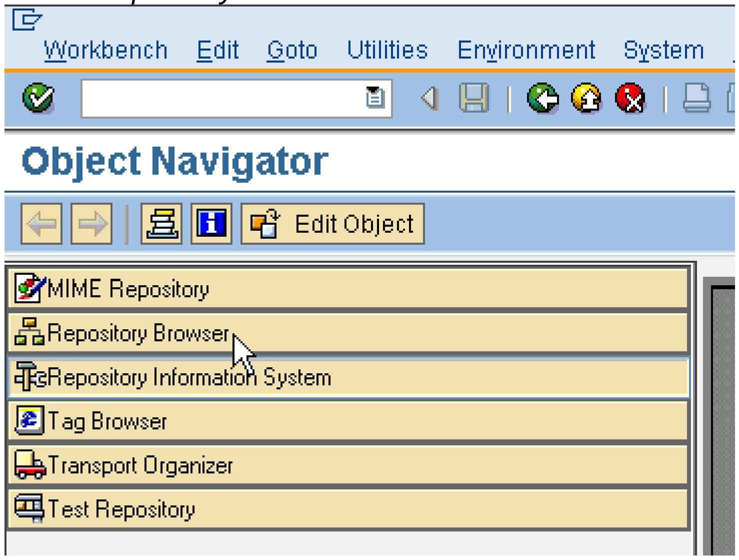
19. Now we are ready to write the .NET client.

Check for Package

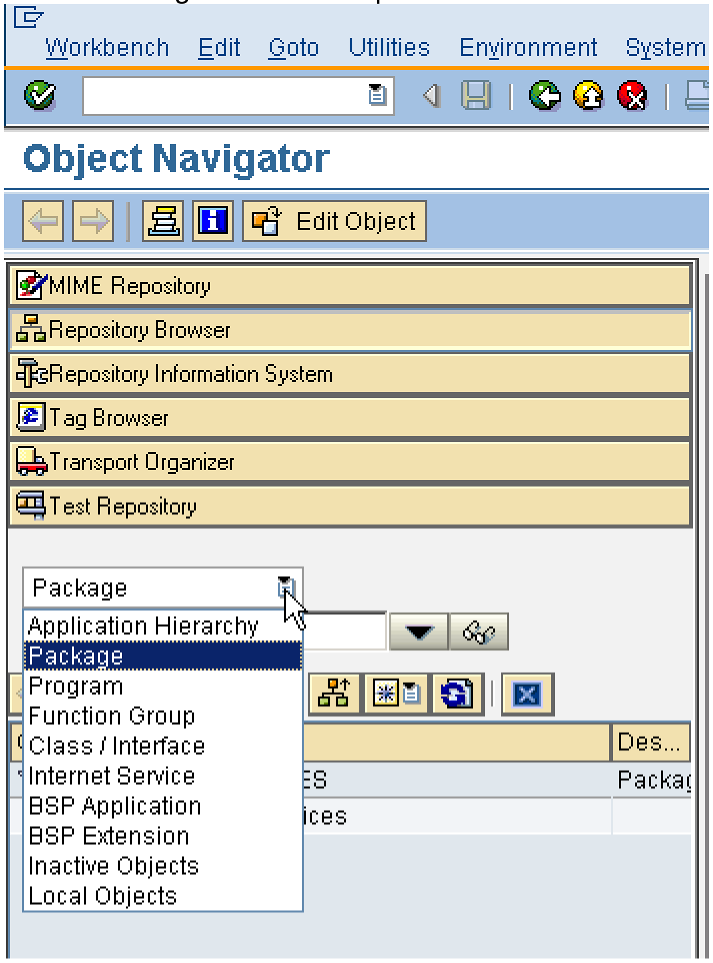
1. Enter se80 in the transaction code field and press **Enter**.



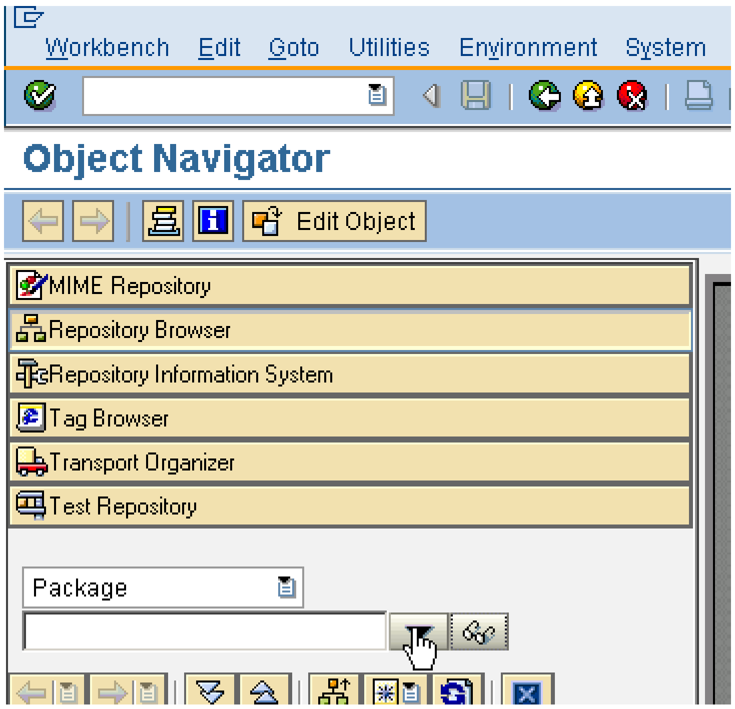
2. Select *Repository Browser*.



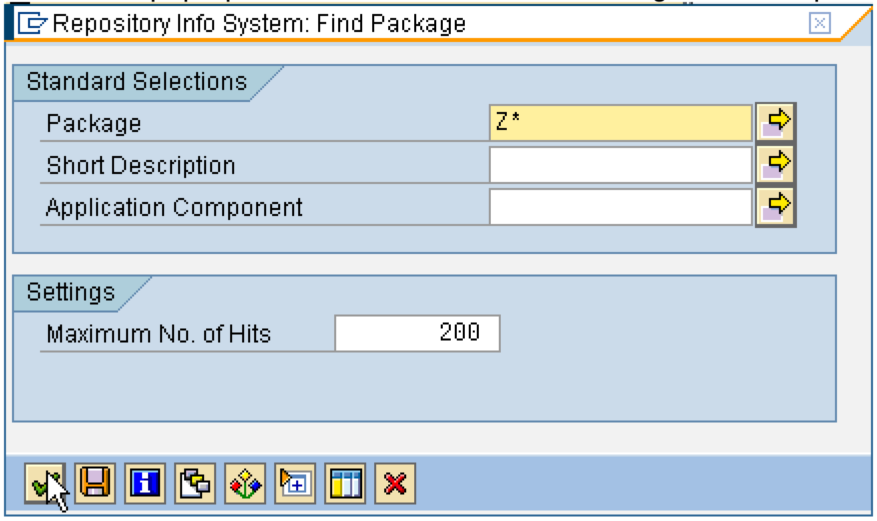
3. Select *Package* from the drop-down list.



4. Click the icon ▼.



5. In the next pop-up window enter Z* into the *Package* field and press **Enter**.


Repository Info System: Find Package

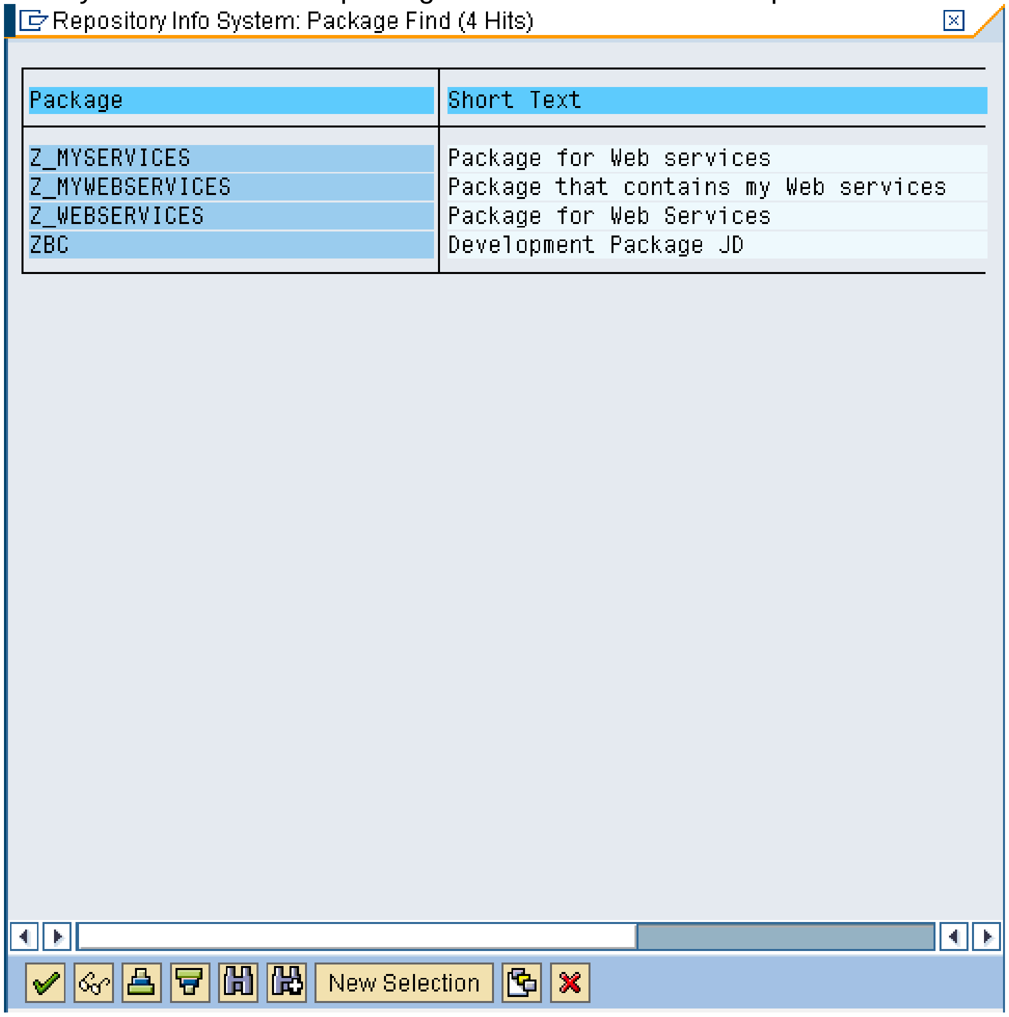
Standard Selections

Package	Z*
Short Description	
Application Component	

Settings

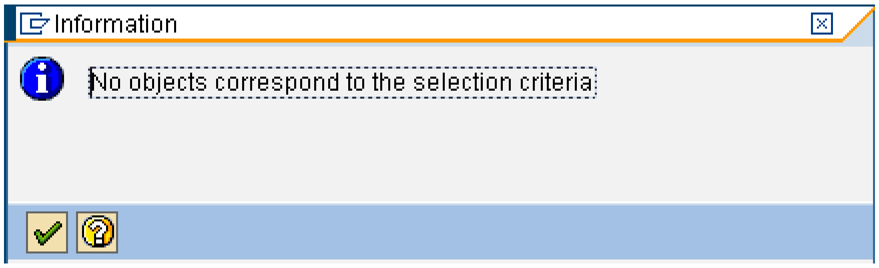
Maximum No. of Hits	200

6. Then you will see a list of packages in the customer namespace Z.


Repository Info System: Package Find (4 Hits)

Package	Short Text
Z_MYSERVICES	Package for Web services
Z_MYWEBSERVICES	Package that contains my Web services
Z_WEBSERVICES	Package for Web Services
ZBC	Development Package JD

If you get the message below you do not have a package in the Z-namespace. Then repeat the search for X*. If you still get the message below or if you do not want to use the existing package execute the steps below. Otherwise continue with SAP Web Service.


Information

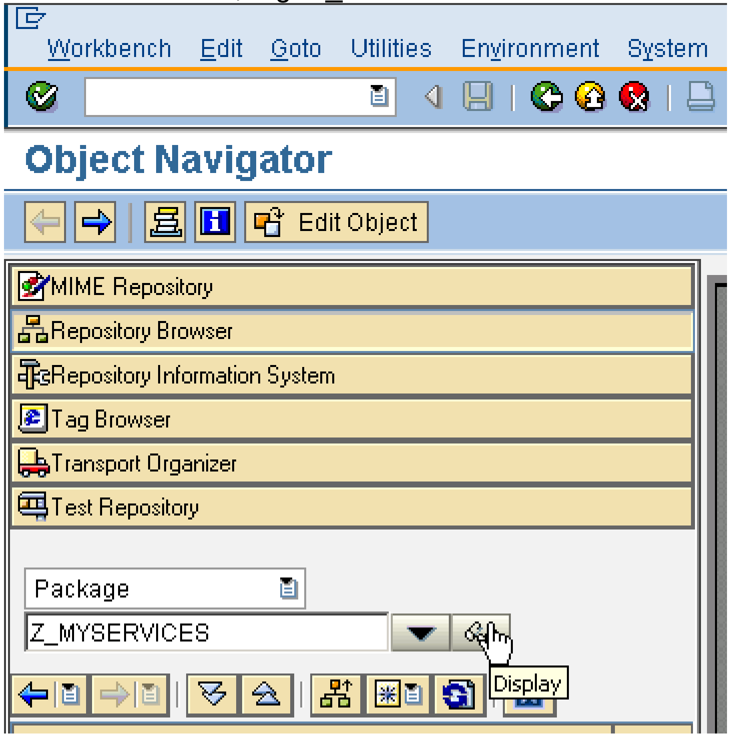
No objects correspond to the selection criteria

7. Close all pop-up windows.

8. If you do not have a package in a customer namespace yet enter a name for your package.
For this scenario we use the package Z_MYSERVICES. You can actually name

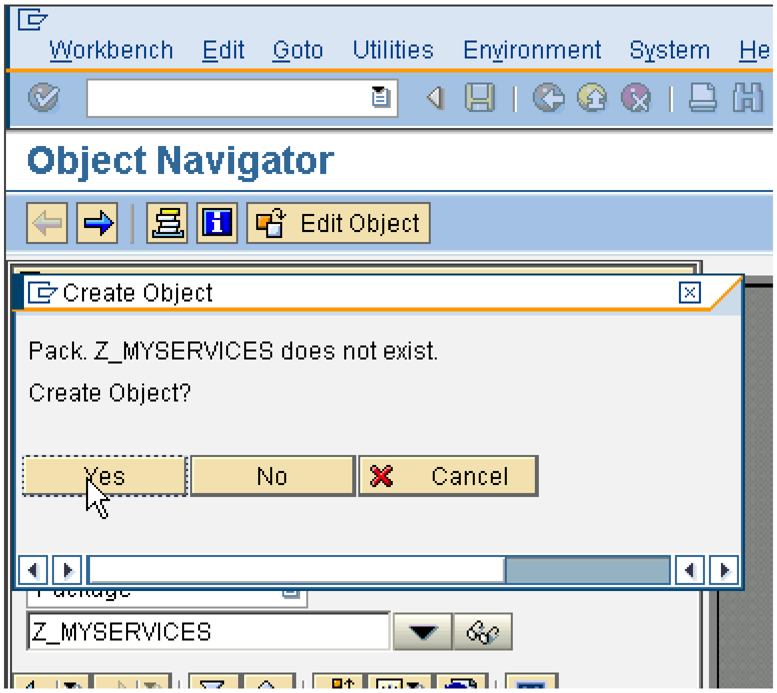
your package as you like, but the name must be compliant with SAPô Naming Conventions for Packages (http://help.sap.com/saphelp_nw04/helpdata/en/ea/c05da6f01011d3964000a0c94260a5/content.htm). For testing the $-namespace would be the handiest one as objects in this namespace cannot be transported. You donô have to create any transport requests. In some cases the Web Service Creation Wizard broke when finishing the setup process, in this case you can use the customer namespace Z for the sample scenario.

Enter for instance, e.g. Z_MYSERVICES in the text box and press **Enter**.



Z_MYSERVICES

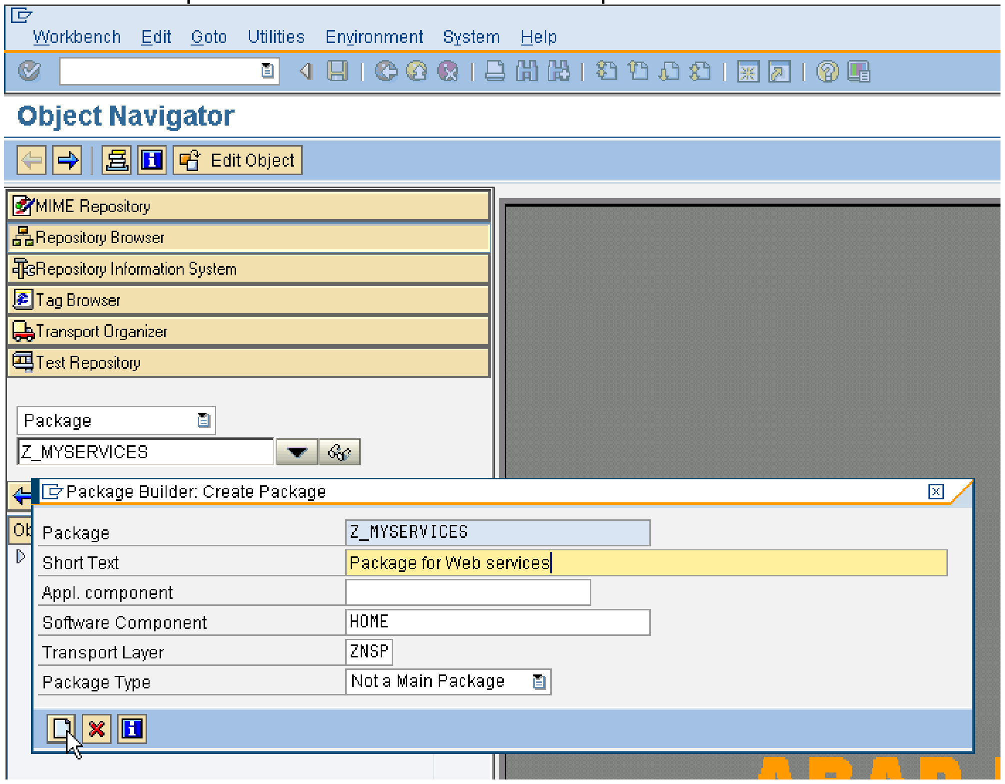
1. Click *Yes*.



Pack. Z_MYSERVICES does not exist.
Create Object?

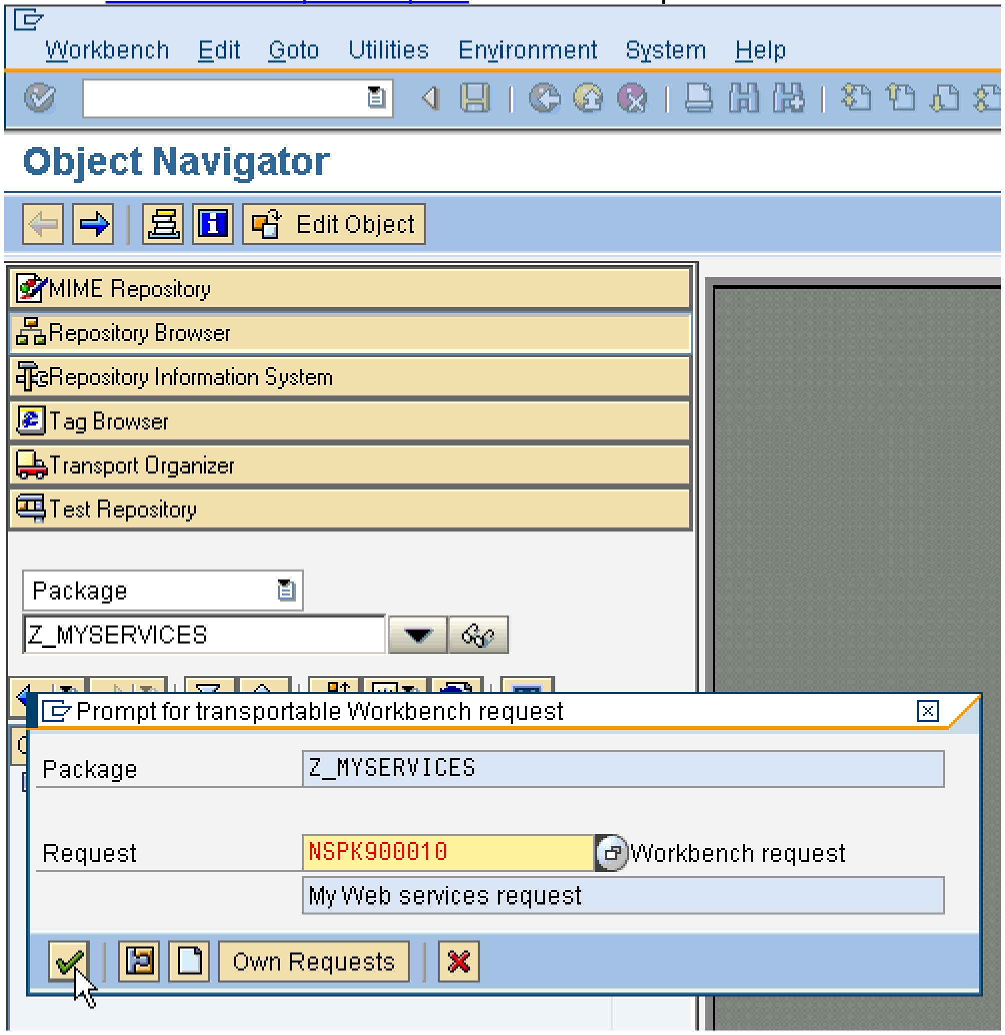
Z_MYSERVICES

2. Enter a description in the *Short Text* field and press **Enter**.



Package	Z_MYSERVICES
Short Text	Package for Web services
Appl. component	
Software Component	HOME
Transport Layer	ZNSP
Package Type	Not a Main Package

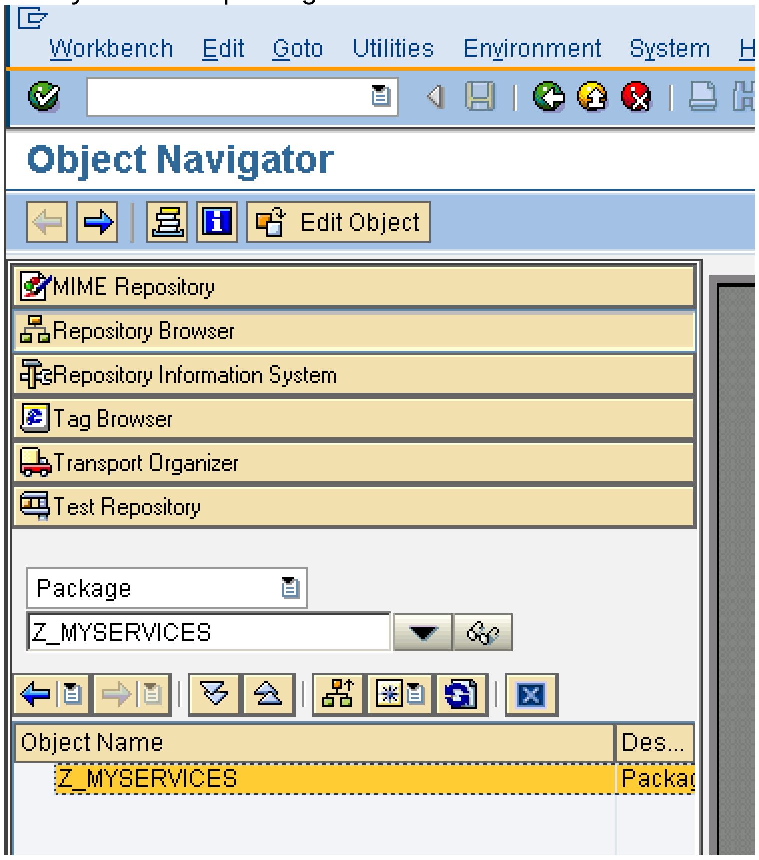
3. If you have a transport request press **Enter**, if not follow the instructions in section Create a Transport Request to create a request.



Prompt for transportable Workbench request

Package	Z_MYSERVICES
Request	NSPK900010 Workbench request
	My Web services request

4. Now you have a package.



Package

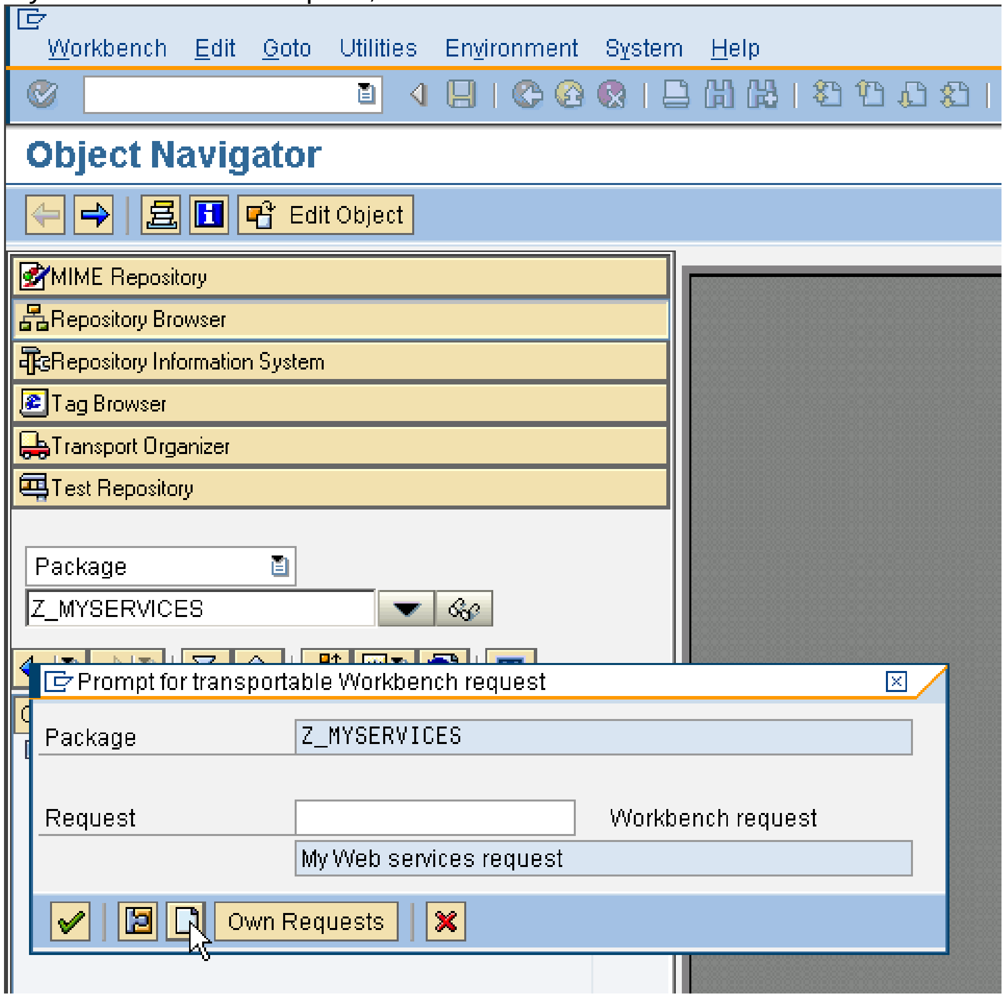
Z_MYSERVICES

Object Name	Des...
Z_MYSERVICES	Packag

5. Continue with SAP Web Service.

Create a Transport Request

1. If you do not have a request, create a new one.



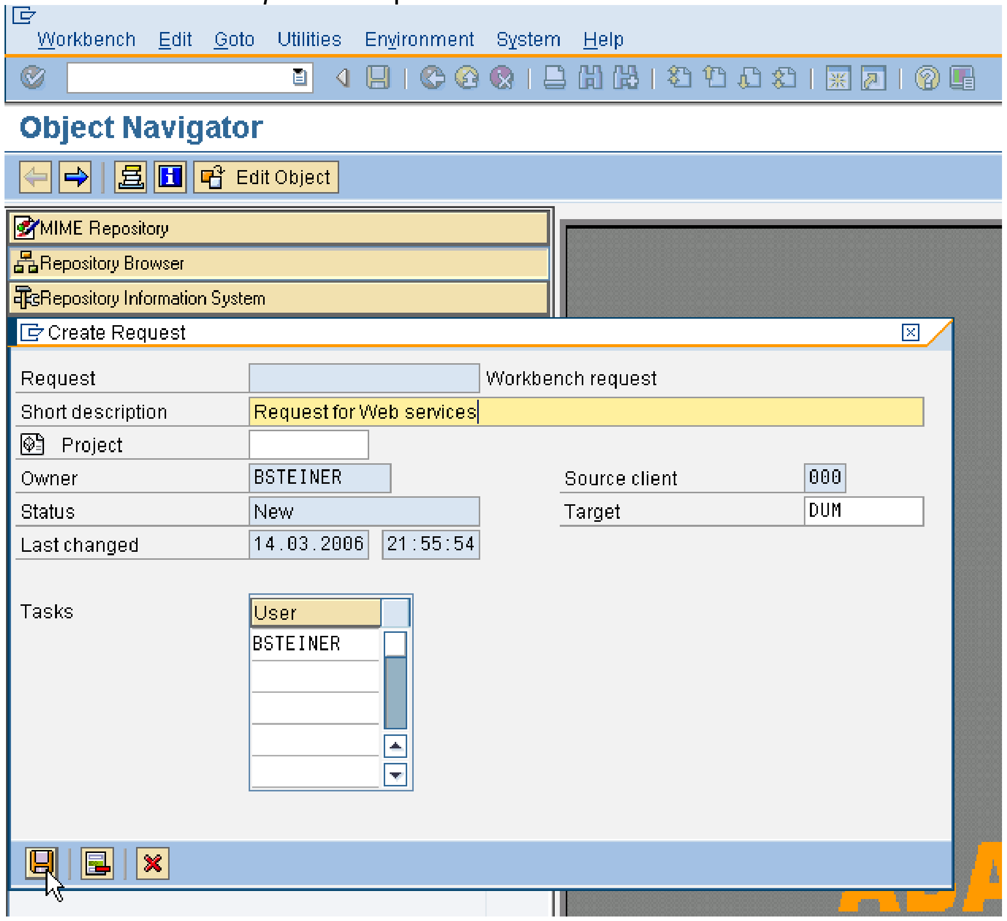
Object Navigator

Package: Z_MYSERVICES

Prompt for transportable Workbench request

Package	Z_MYSERVICES	
Request		Workbench request
	My Web services request	

2. Enter a *Short description* and press **Enter**.



Request		Workbench request		
Short description	Request for Web services			
Project				
Owner	BSTEINER		Source client	000
Status	New		Target	DUM
Last changed	14.03.2006	21:55:54		

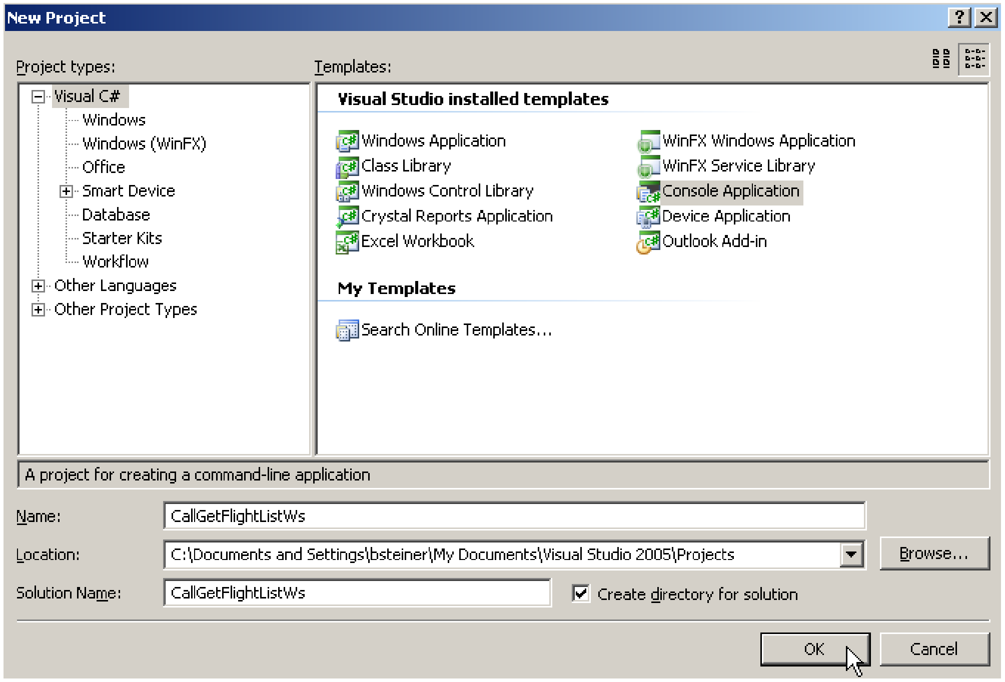
Tasks	User
	BSTEINER

.NET client

Follow the steps below to create a .NET client for the SAP Web Service created before.

1. Start Visual Studio 2005.

2. Create a Console Application and enter CallGetFlightListWs as *Name* and click *OK*.



A project for creating a command-line application

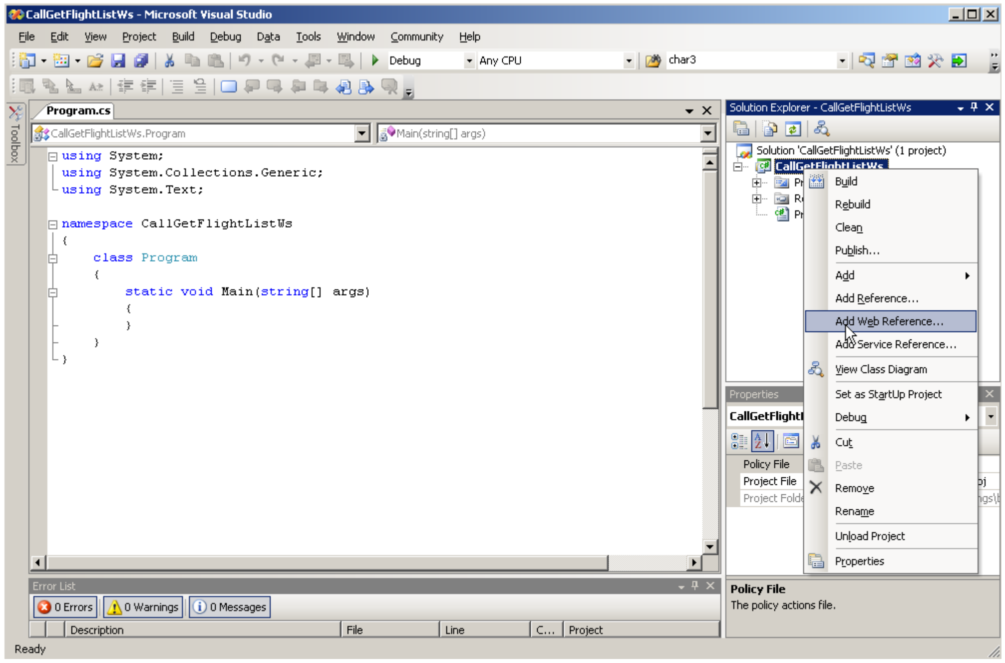
Name: CallGetFlightListWs

Location: C:\Documents and Settings\bsteiner\My Documents\Visual Studio 2005\Projects

Solution Name: CallGetFlightListWs

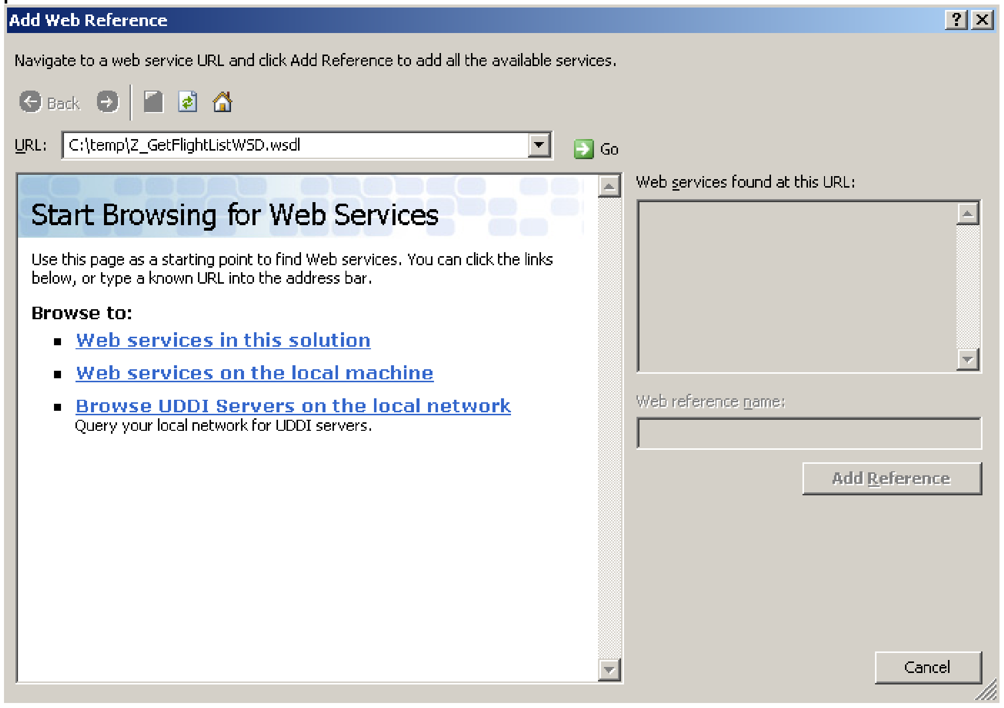
[✓] Create directory for solution

3. In the Solution Explorer right-click *CallGetFlightListWs* and select *Add Web Reference*.



```
using System;
using System.Collections.Generic;
using System.Text;

namespace CallGetFlightListWs
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

4. Enter the path to the previously saved WSDL or the SAP Web Serviceś URL and press **Enter**.


Add Web Reference
Back
Go
Web services in this solution
Web services on the local machine
Browse UDDI Servers on the local network
Add Reference
Cancel

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

URL: C:\temp\Z_GetFlightListWSD.wsdl

Start Browsing for Web Services

Use this page as a starting point to find Web services. You can click the links below, or type a known URL into the address bar.

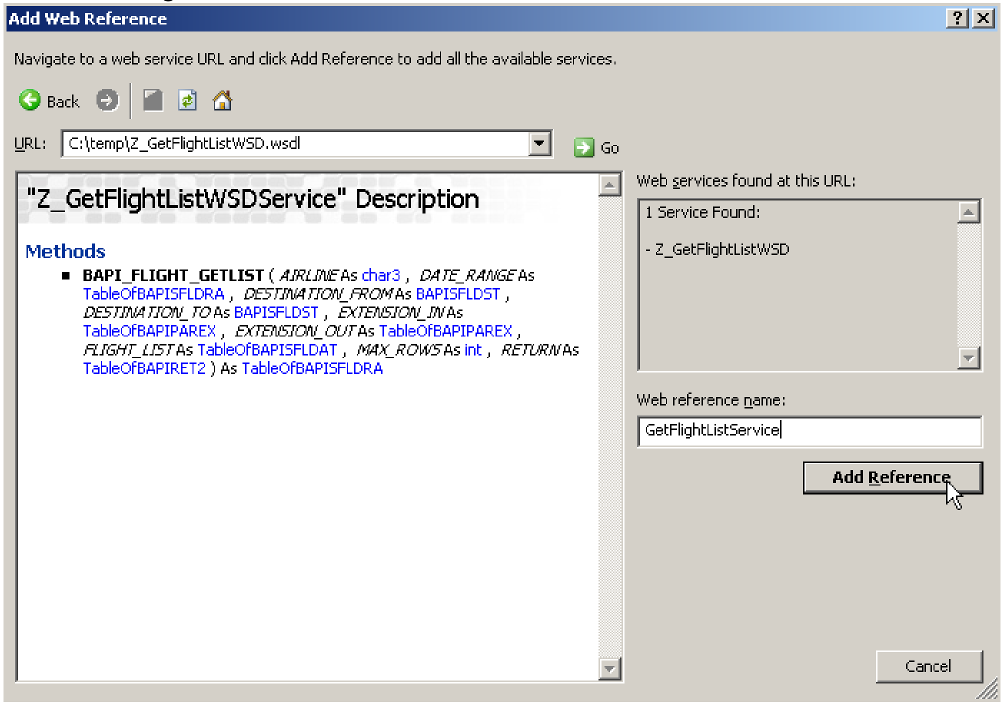
Browse to:

Query your local network for UDDI servers.

Web services found at this URL:

Web reference name:

5. Enter GetFlightListService as *Web reference name* and click *Add Reference*.



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

URL: C:\temp\Z_GetFlightListWSD.wsdl

"Z_GetFlightListWSDService" Description

- **BAPI_FLIGHT_GETLIST** (*AIRLINE* As char3 , *DATE_RANGE* As TableOfBAPISFLDRA , *DESTINATION_FROM* As BAPISFLDST , *DESTINATION_TO* As BAPISFLDST , *EXTENSION_IN* As TableOfBAPIPAREX , *EXTENSION_OUT* As TableOfBAPIPAREX , *FLIGHT_LIST* As TableOfBAPISFLDAT , *MAX_ROWS* As int , *RETURN* As TableOfBAPIRET2) As TableOfBAPISFLDRA

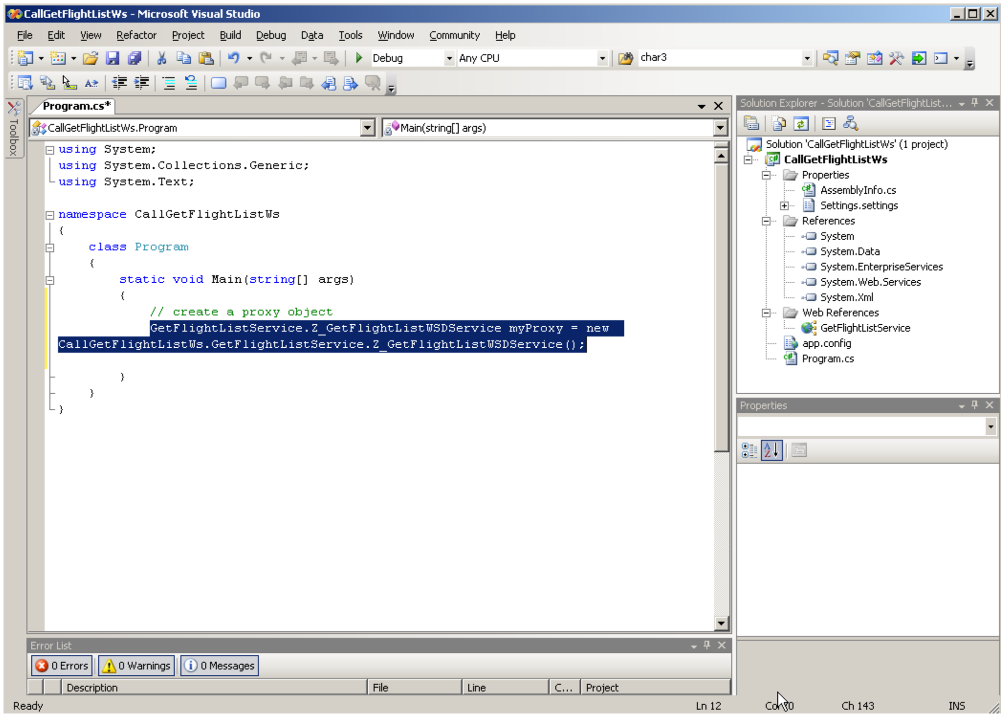
Web services found at this URL:

1 Service Found:

- Z_GetFlightListWSD

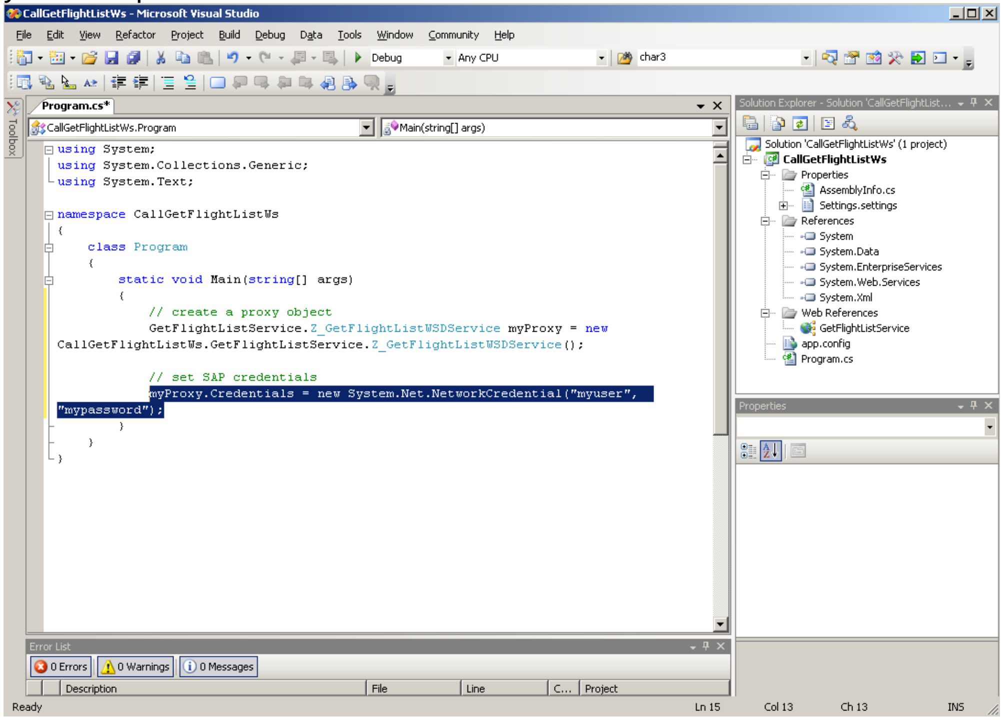
Web reference name: GetFlightListService

6. Add the code highlighted below to the Main method. This creates a proxy object for the SAP Web Service.



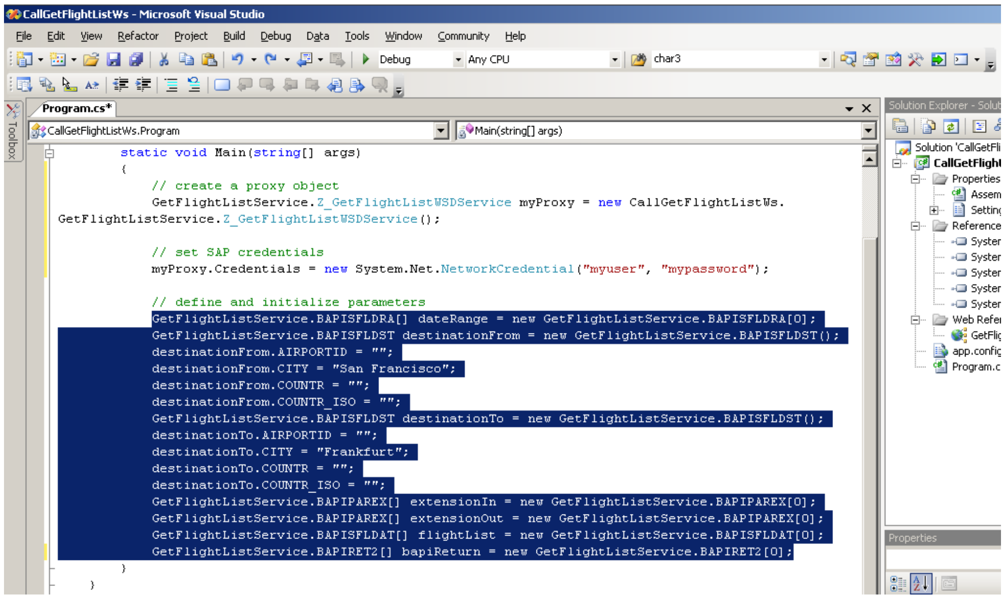
```
using System;
using System.Collections.Generic;
using System.Text;

namespace CallGetFlightListWs
{
    class Program
    {
        static void Main(string[] args)
        {
            // create a proxy object
            GetFlightListService.Z_GetFlightListWSDService myProxy = new
CallGetFlightListWs.GetFlightListService.Z_GetFlightListWSDService();

        }
    }
}
```

7. Now add the highlighted code to set the SAP credentials used to call the SAP Web Service. Replace myuser with your SAP user name and mypassword with your SAP password.



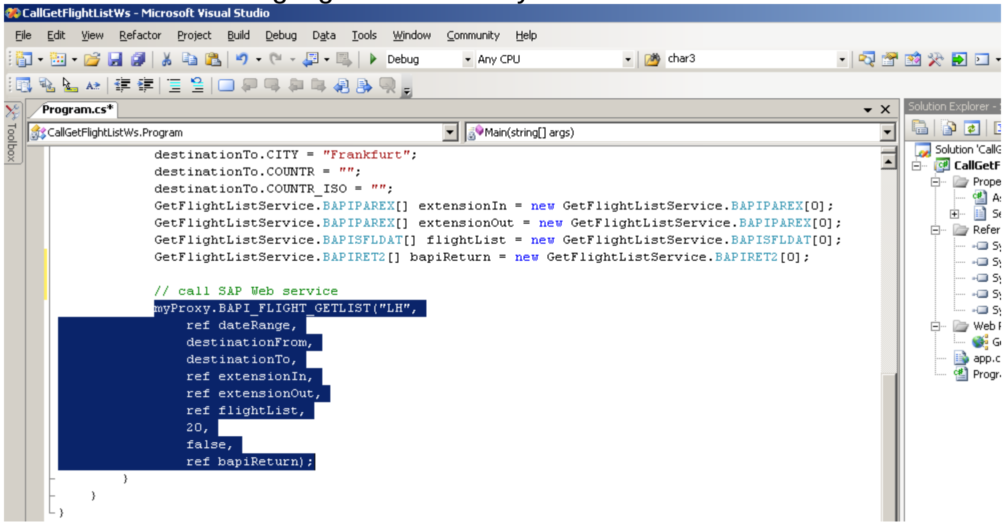
```
using System;
using System.Collections.Generic;
using System.Text;

namespace CallGetFlightListWs
{
    class Program
    {
        static void Main(string[] args)
        {
            // create a proxy object
            GetFlightListService.Z_GetFlightListWSDService myProxy = new
CallGetFlightListWs.GetFlightListService.Z_GetFlightListWSDService();

            // set SAP credentials
            myProxy.Credentials = new System.Net.NetworkCredential("myuser",
"mypassword");
        }
    }
}
```

8. Enter the code highlighted below. This creates all parameters required for calling the SAP Web Service.



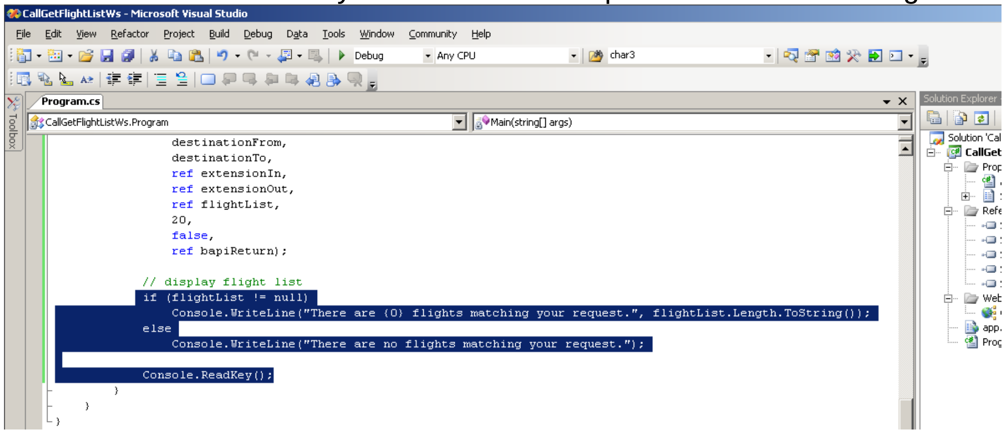
```
        static void Main(string[] args)
        {
            // create a proxy object
            GetFlightListService.Z_GetFlightListWSDService myProxy = new CallGetFlightListWs.
GetFlightListService.Z_GetFlightListWSDService();

            // set SAP credentials
            myProxy.Credentials = new System.Net.NetworkCredential("myuser", "mypassword");

            // define and initialize parameters
            GetFlightListService.BAPISFLDRA[] dateRange = new GetFlightListService.BAPISFLDRA[0];
            GetFlightListService.BAPISFLDST destinationFrom = new GetFlightListService.BAPISFLDST();
            destinationFrom.AIRPORTID = "";
            destinationFrom.CITY = "San Francisco";
            destinationFrom.COUNTR = "";
            destinationFrom.COUNTR_ISO = "";
            GetFlightListService.BAPISFLDST destinationTo = new GetFlightListService.BAPISFLDST();
            destinationTo.AIRPORTID = "";
            destinationTo.CITY = "Frankfurt";
            destinationTo.COUNTR = "";
            destinationTo.COUNTR_ISO = "";
            GetFlightListService.BAPIPAREX[] extensionIn = new GetFlightListService.BAPIPAREX[0];
            GetFlightListService.BAPIPAREX[] extensionOut = new GetFlightListService.BAPIPAREX[0];
            GetFlightListService.BAPISFLDAT[] flightList = new GetFlightListService.BAPISFLDAT[0];
            GetFlightListService.BAPIRET2[] bapiReturn = new GetFlightListService.BAPIRET2[0];
        }
    }
```

9. Now add the code highlighted to actually call the SAP Web Service.



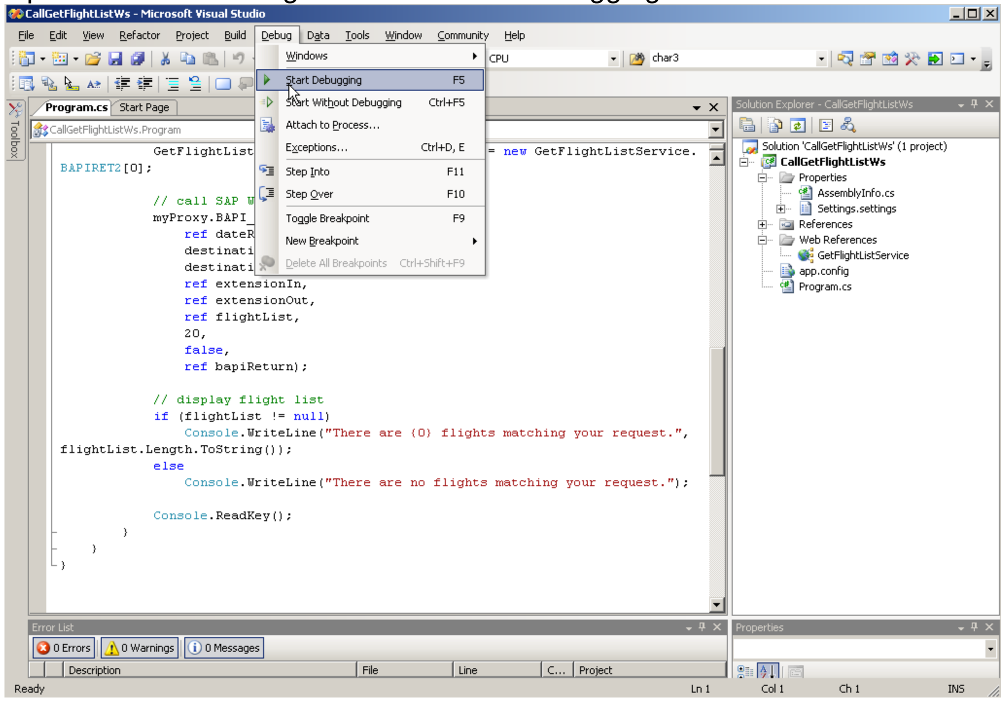
```
            destinationTo.CITY = "Frankfurt";
            destinationTo.COUNTR = "";
            destinationTo.COUNTR_ISO = "";
            GetFlightListService.BAPIPAREX[] extensionIn = new GetFlightListService.BAPIPAREX[0];
            GetFlightListService.BAPIPAREX[] extensionOut = new GetFlightListService.BAPIPAREX[0];
            GetFlightListService.BAPISFLDAT[] flightList = new GetFlightListService.BAPISFLDAT[0];
            GetFlightListService.BAPIRET2[] bapiReturn = new GetFlightListService.BAPIRET2[0];

            // call SAP Web service
            myProxy.BAPI_FLIGHT_GETLIST("LH",
                ref dateRange,
                destinationFrom,
                destinationTo,
                ref extensionIn,
                ref extensionOut,
                ref flightList,
                20,
                false,
                ref bapiReturn);
        }
    }
}
```

10. Finally add the highlighted lines to display the number of flights found or a message that no flight was found.
With Console.ReadKey() your console application waits for a key entered before it exits. This ensures that you can read the output before the console gets closed.



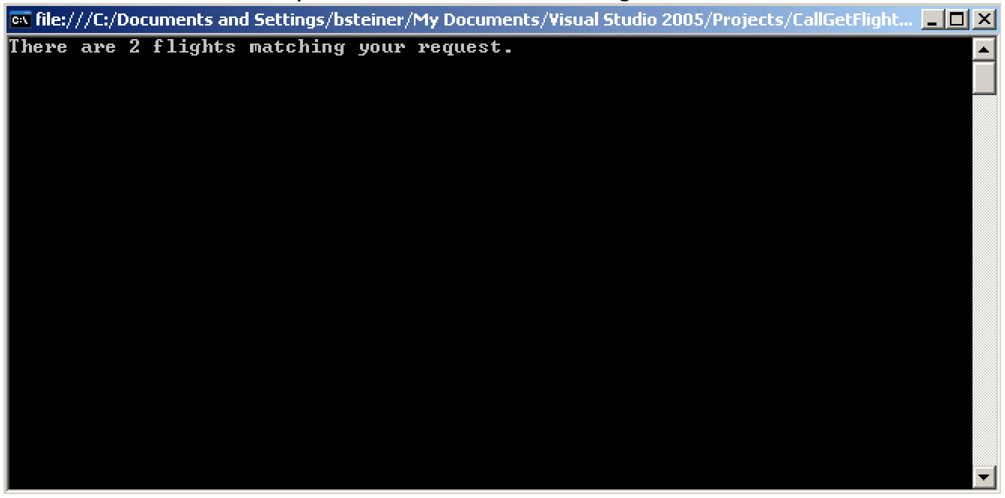
```
                        destinationFrom,
                        destinationTo,
                        ref extensionIn,
                        ref extensionOut,
                        ref flightList,
                        20,
                        false,
                        ref bapiReturn);

                // display flight list
                if (flightList != null)
                    Console.WriteLine("There are {0} flights matching your request.", flightList.Length.ToString());
                else
                    Console.WriteLine("There are no flights matching your request.");

                Console.ReadKey();
            }
        }
}
```

11. Open the menu *Debug* and select *Start Debugging*.



```
                GetFlightList                                  = new GetFlightListService.
BAPIRET2[0];

                // call SAP W
                myProxy.BAPI_
                    ref dateR
                    destinati
                    destinati
                    ref extensionIn,
                    ref extensionOut,
                    ref flightList,
                    20,
                    false,
                    ref bapiReturn);

                // display flight list
                if (flightList != null)
                    Console.WriteLine("There are {0} flights matching your request.",
flightList.Length.ToString());
                else
                    Console.WriteLine("There are no flights matching your request.");

                Console.ReadKey();
            }
        }
}
```

12. You should see an output similar to the following.



file:///C:/Documents and Settings/bsteiner/My Documents/Visual Studio 2005/Projects/CallGetFlight...

```
There are 2 flights matching your request.
```

SAP Transactions

Manage Packages

Transaction	
SE80	Object Navigator, the central development transcation, e.g. to create new packages

Web Service Creation

Transaction	
SE37	Function Builder, e.g. to select a BAPI for Web Service creation
SE80	Object Navigator, e.g. to select a BAPI for Web Service creation
WS_WZD_START	Web Service Creation Wizard

Web Service Administration and Configuration

Transaction	
WSADMIN	Web Service Administration for SOAP Runtime, e.g. to get WSDL or configure tracing
WSCONFIG	Release Web Service for SOAP Runtime, e.g. to change address
SICF	Maintain ICF service

Web Service Troubleshooting

Transaction	
SM59	Display and Maintain RFC Destinations, e.g. to view trace files
SM21	System Log
SMICM	ICM Monitor, e.g. to view log files or to restart ICM
ST01	System Trace, e.g. to trace required permissions
SU53	Display Authorization Data for User, e.g. to see failed authorization attempts