Software Tool & Die Inc.

Presents

Web Services Primer

Getting Started

A Web Services Tour!

By *Chris Pollach* – President: Software Tool & Die Inc.

Ottawa, Canada

• Email:	cpollach@travel-net.com
• Blog:	http://chrispollach.blogspot.ca
• PBDJ:	http://chrispollach.sys-con.com
• LinkedIn:	http://ca.linkedin.com/in/chrispollach
• SourceForge:	http://sourceforge.net/projects/stdfndclass
• STD:	http://www.softdie.com

PowerBuilder Classic Primer

Web Services Primer

Web Services Goals

- Facilitate communication between systems
  - Different platforms
  - Different programming languages
  - Through firewalls easily
  - Self descriptive API
  - Self descriptive data

What are Web Services?

- A collection of operations that can be described, published, located, and accessed over a network using standardized XML messaging
- Proposed to World Wide Web Consortium (W3C) in Mar 2001
  - http://www.w3c.org
- Web Services utilize XML making them both platform and language independent
- XML gives us a mechanism for making cross-platform and/or cross-language communications

Web Service Components

- The primary components that make up Web Services are:
  - WSDL – Web Services Description Language
    - Used to describe Web services
  - SOAP – Simple Object Access Protocol
    - Used for sending and receiving messages from Web services

Describing Web Services

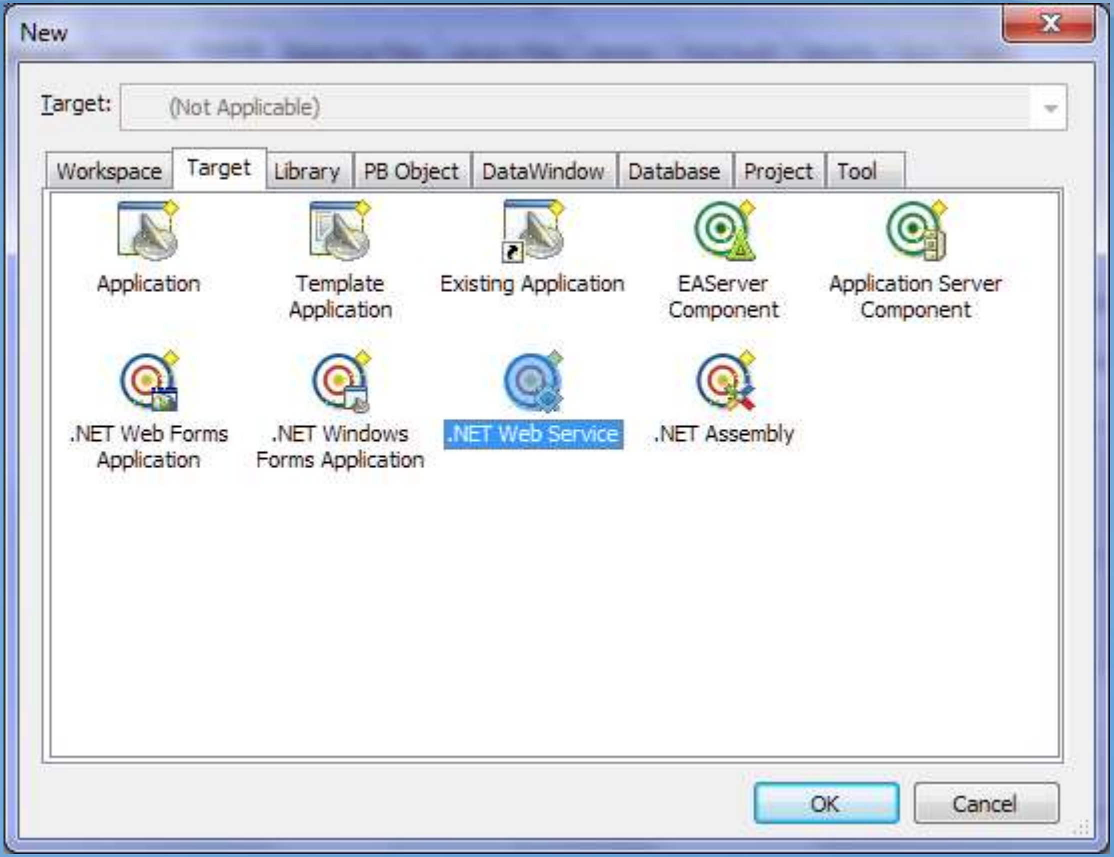
- Why does a Web service need to be described?
  - Web services could be used by anyone, anywhere, using any language on any platform
  - A description allows a developer to know how to interact with a Web service
    - PowerBuilder provides tools to read and integrate WSDL
- Web services are described using Web Services Description Language (WSDL)
- WSDL is written in XML
- Usually a developer of a Web Service does not have to manually write WSDL
  - PowerBuilder 11 (or higher) creates the ASMX, DISCO and WSDL

CREATING .NET Web Services in PowerBuilder Classsic

PowerBuilder/.Net Web Services

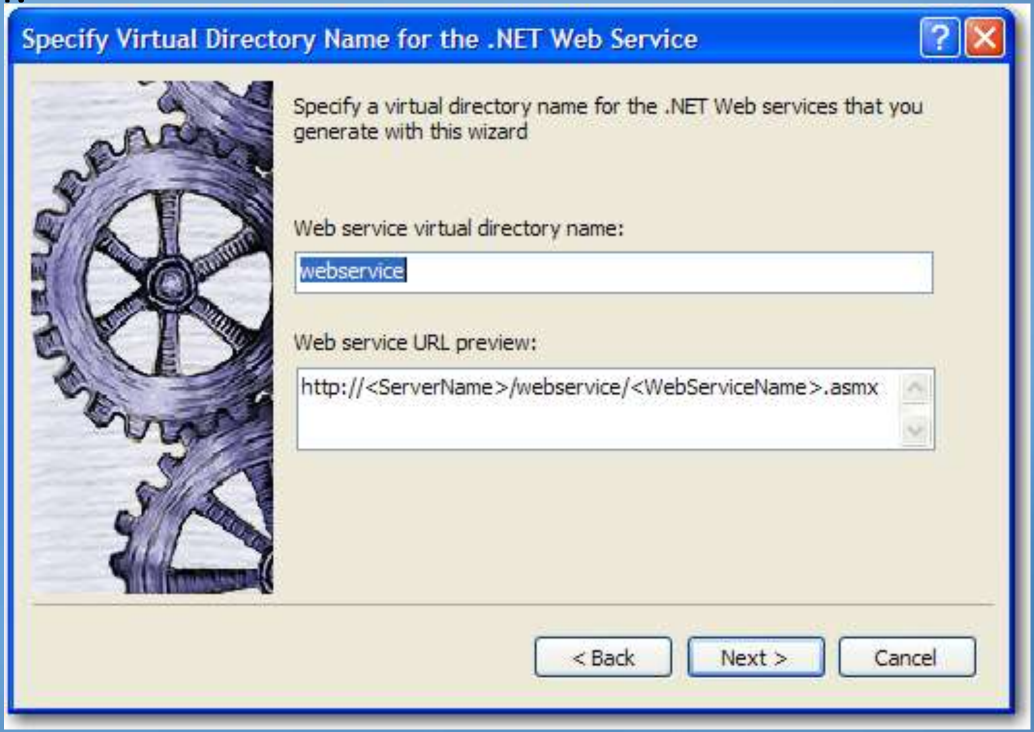
- PowerBuilder gives you the choice of outputting PowerScript code as an
  - Assembly
  - Web Service
- PowerBuilder Web Services are deployed to your Microsoft **IIS** Web Server or **EAServer** (EOL 2016)

.Net Web Service Target



Virtual Directory

- The wizard is virtually the same as for .NET assemblies, etc.
- You must specify a *virtual directory* name for your Web Service as it will live on the IIS server.


Specify Virtual Directory Name for the .NET Web Service

Specify a virtual directory name for the .NET Web services that you generate with this wizard

Web service virtual directory name:

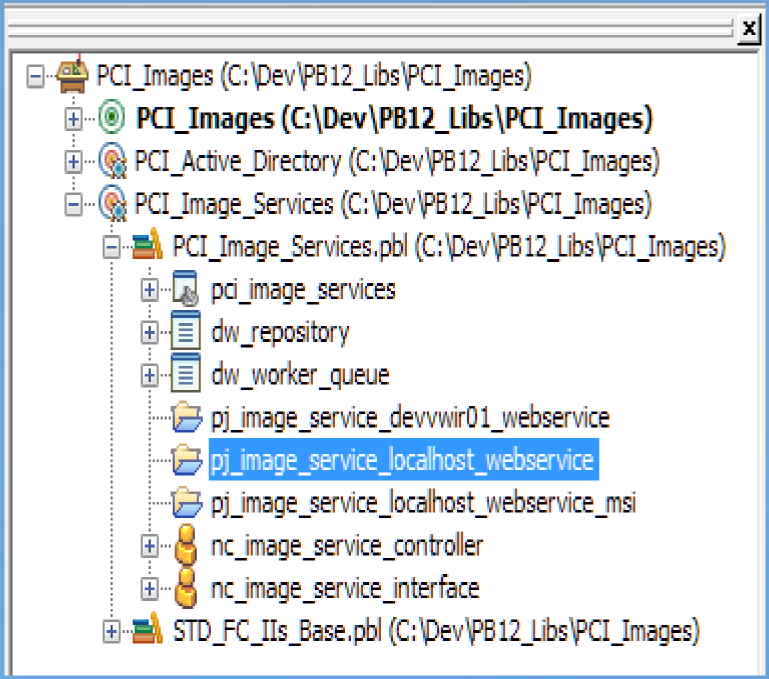
webservice

Web service URL preview:

http://<ServerName>/webservice/<WebServiceName>.asmx

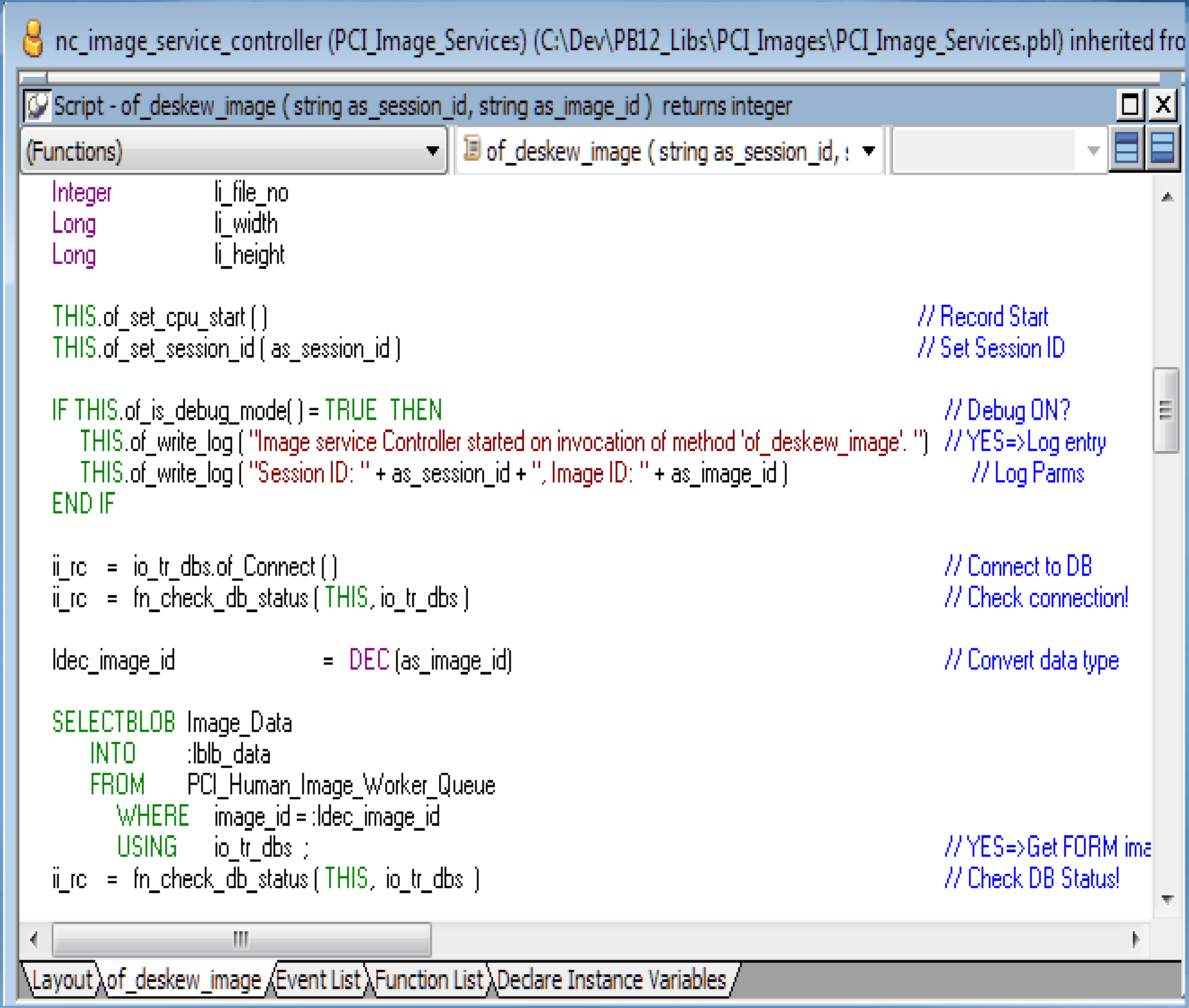
.Net Web Service Wizard Output

- PBL, Application Object, Project, NVO



```
PCI_Images (C:\Dev\PB12_Libs\PCI_Images)
  PCI_Images (C:\Dev\PB12_Libs\PCI_Images)
  PCI_Active_Directory (C:\Dev\PB12_Libs\PCI_Images)
  PCI_Image_Services (C:\Dev\PB12_Libs\PCI_Images)
    PCI_Image_Services.pbl (C:\Dev\PB12_Libs\PCI_Images)
      pci_image_services
      dw_repository
      dw_worker_queue
      pj_image_service_devvwir01_webservice
      pj_image_service_localhost_webservice
      pj_image_service_localhost_webservice_msi
      nc_image_service_controller
      nc_image_service_interface
    STD_FC_IIs_Base.pbl (C:\Dev\PB12_Libs\PCI_Images)
```

NVUOs – Code as you normally would



nc_image_service_controller (PCI_Image_Services) (C:\Dev\PB12_Libs\PCI_Images\PCI_Image_Services.pbl) inherited fro

Script - of_deskew_image (string as_session_id, string as_image_id) returns integer

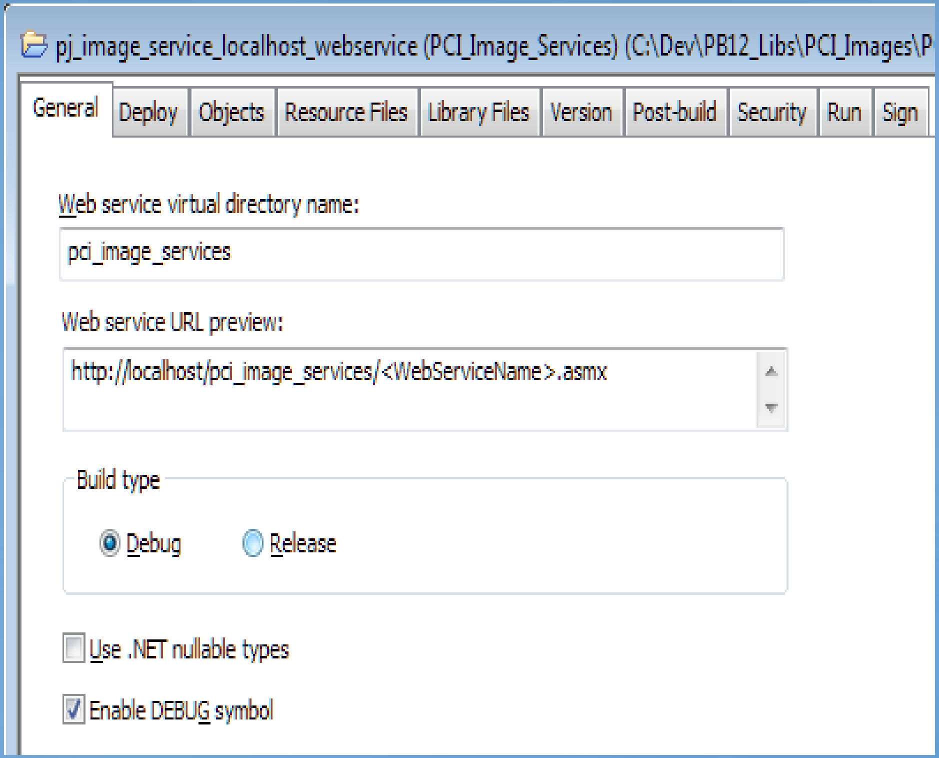
(Functions) | of_deskew_image (string as_session_id, :

```
Integer             li_file_no
Long                li_width
Long                li_height

THIS.of_set_cpu_start ( )                                                                  // Record Start
THIS.of_set_session_id ( as_session_id )                                                   // Set Session ID

IF THIS.of_is_debug_mode( ) = TRUE  THEN                                                    // Debug ON?
    THIS.of_write_log ( "Image service Controller started on invocation of method 'of_deskew_image'. ")  // YES=>Log entry
    THIS.of_write_log ( "Session ID: " + as_session_id + ", Image ID: " + as_image_id )       // Log Parms
END IF

ii_rc   =  io_tr_dbs.of_Connect ( )                                                          // Connect to DB
ii_rc   =  fn_check_db_status ( THIS, io_tr_dbs )                                            // Check connection!

ldec_image_id                =  DEC (as_image_id)                                            // Convert data type

SELECTBLOB  Image_Data
      INTO        :lblb_data
      FROM        PCI_Human_Image_Worker_Queue
          WHERE     image_id = :ldec_image_id
          USING     io_tr_dbs ;                                                              // YES=>Get FORM ima
ii_rc   =  fn_check_db_status ( THIS,  io_tr_dbs  )                                          // Check DB Status!
```

Layout | of_deskew_image | Event List | Function List | Declare Instance Variables

Web Service Project

- Wizard selections may always be changed in the Project


pj_image_service_localhost_webservice (PCI_Image_Services) (C:\Dev\PB12_Libs\PCI_Images\P

Web service virtual directory name:

pci_image_services

Web service URL preview:

http://localhost/pci_image_services/<WebServiceName>.asmx

Build type

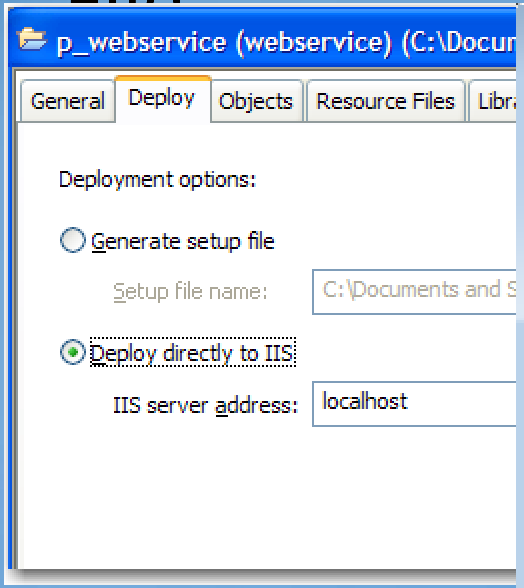
[✓ Debug] Release

☐ Use .NET nullable types

☑ Enable DEBUG symbol

Deployment Options

- Directly to IIS   or   create an MSI install File


p_webservice (webservice) (C:\Docun

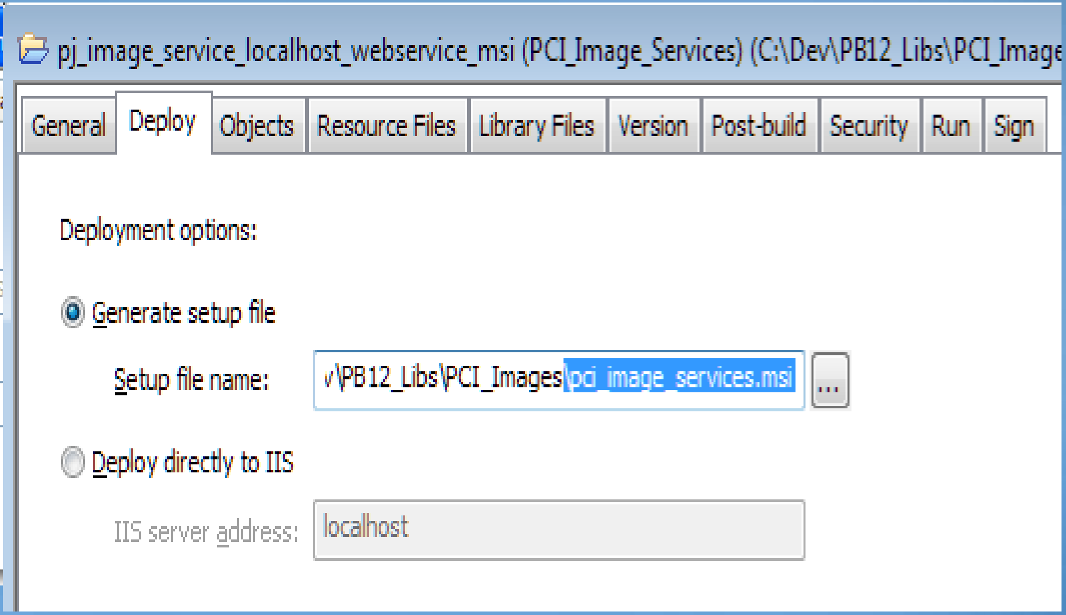
Deployment options:

( ) Generate setup file

Setup file name: C:\Documents and S

[● Deploy directly to IIS]

IIS server address: localhost


pj_image_service_localhost_webservice_msi (PCI_Image_Services) (C:\Dev\PB12_Libs\PCI_Image

Deployment options:

[● Generate setup file]

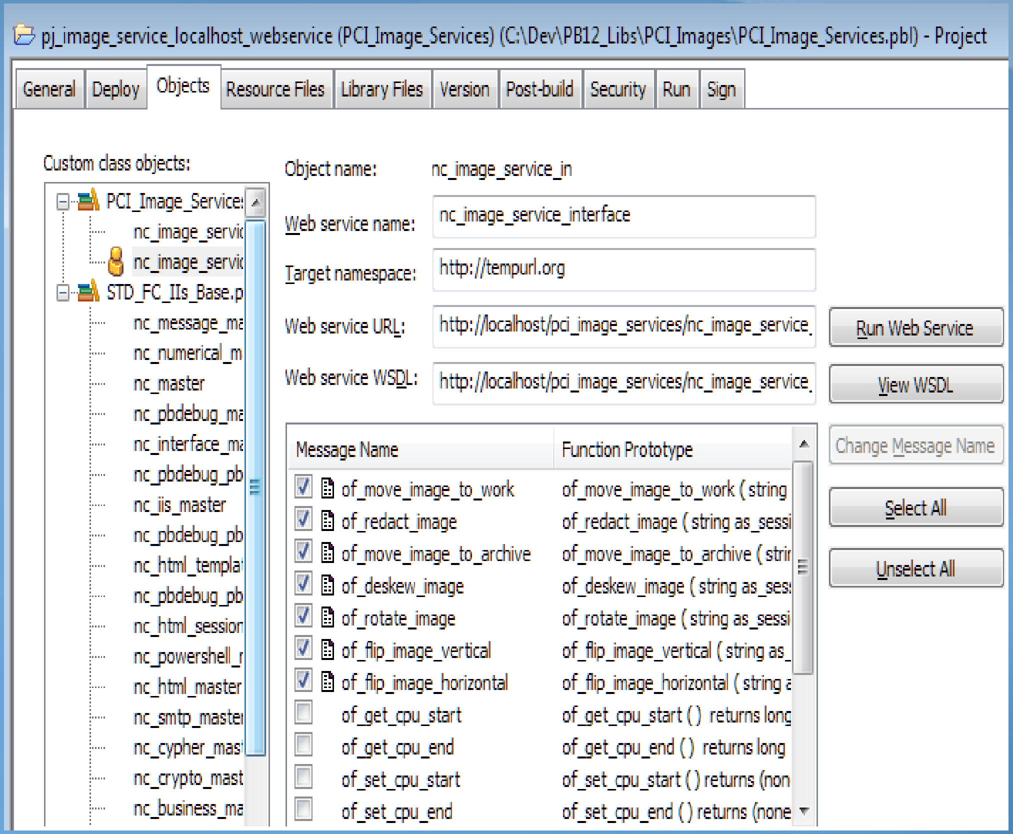
Setup file name: v\PB12_Libs\PCI_Images\pci_image_services.msi

( ) Deploy directly to IIS

IIS server address: localhost

Specifications

- You ***must*** select which methods you want to expose
- You can view WSDL and test your Web Service



pj_image_service_localhost_webservice (PCI_Image_Services) (C:\Dev\PB12_Libs\PCI_Images\PCI_Image_Services.pbl) - Project

General | Deploy | Objects | Resource Files | Library Files | Version | Post-build | Security | Run | Sign

Custom class objects:

- PCI_Image_Service
  - nc_image_servic
  - nc_image_servic
- STD_FC_IIs_Base.p
  - nc_message_ma
  - nc_numerical_m
  - nc_master
  - nc_pbdebug_ma
  - nc_interface_ma
  - nc_pbdebug_pb
  - nc_iis_master
  - nc_pbdebug_pb
  - nc_html_templa
  - nc_pbdebug_pb
  - nc_html_session
  - nc_powershell_r
  - nc_html_master
  - nc_smtp_maste
  - nc_cypher_mas
  - nc_crypto_mast
  - nc_business_ma

Object name: nc_image_service_in

Web service name: nc_image_service_interface

Target namespace: http://tempurl.org

Web service URL: http://localhost/pci_image_services/nc_image_service

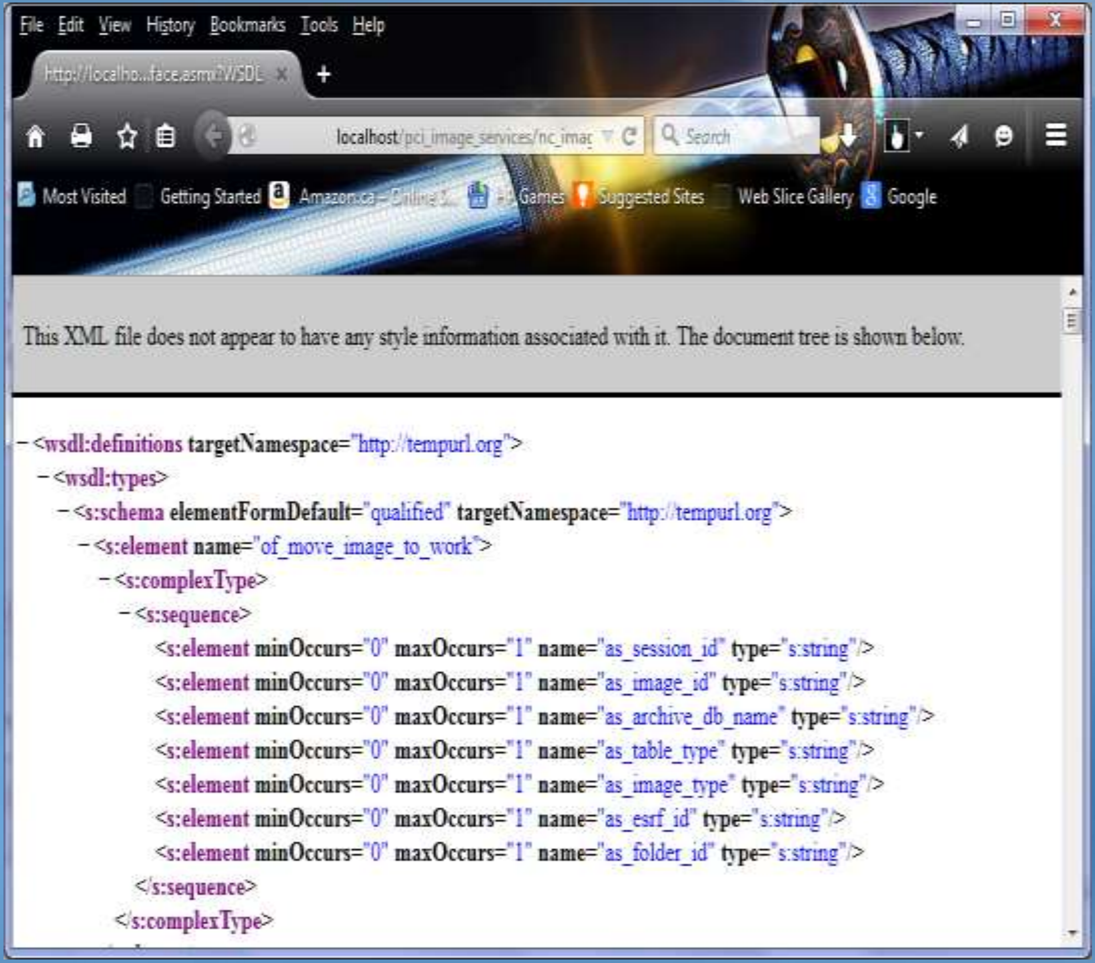
Web service WSDL: http://localhost/pci_image_services/nc_image_service

	Message Name	Function Prototype
✓	of_move_image_to_work	of_move_image_to_work (string
✓	of_redact_image	of_redact_image (string as_sessi
✓	of_move_image_to_archive	of_move_image_to_archive (strir
✓	of_deskew_image	of_deskew_image (string as_ses
✓	of_rotate_image	of_rotate_image (string as_sessi
✓	of_flip_image_vertical	of_flip_image_vertical (string as_
✓	of_flip_image_horizontal	of_flip_image_horizontal (string a
	of_get_cpu_start	of_get_cpu_start ( ) returns long
	of_get_cpu_end	of_get_cpu_end ( ) returns long
	of_set_cpu_start	of_set_cpu_start ( ) returns (non
	of_set_cpu_end	of_set_cpu_end ( ) returns (none

Viewing WSDL

- Must deploy your .NET Web Service target first
- Project View WSDL button OR
- In a browser

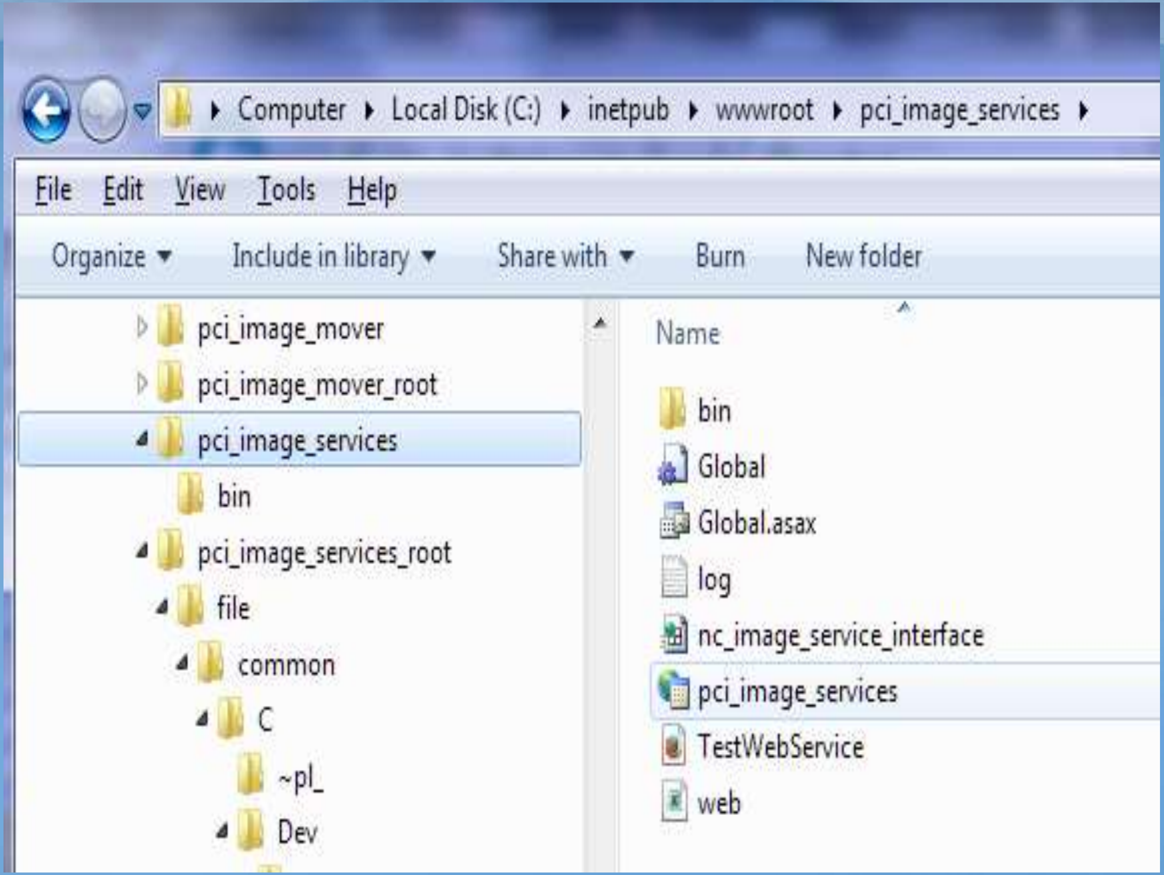
  http://hostname/virtdirname/service.asmx?WSDL


File Edit View History Bookmarks Tools Help
http://localho...face.asmx?WSDL
localhost/pci_image_services/nc_imag
Search
Most Visited Getting Started Amazon.ca - Online S... HP Games Suggested Sites Web Slice Gallery Google

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <wsdl:definitions targetNamespace="http://tempurl.org">
  - <wsdl:types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://tempurl.org">
      - <s:element name="of_move_image_to_work">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="as_session_id" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_image_id" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_archive_db_name" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_table_type" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_image_type" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_esrf_id" type="s:string"/>
              <s:element minOccurs="0" maxOccurs="1" name="as_folder_id" type="s:string"/>
            </s:sequence>
          </s:complexType>
```

IIS Directory – What is here?

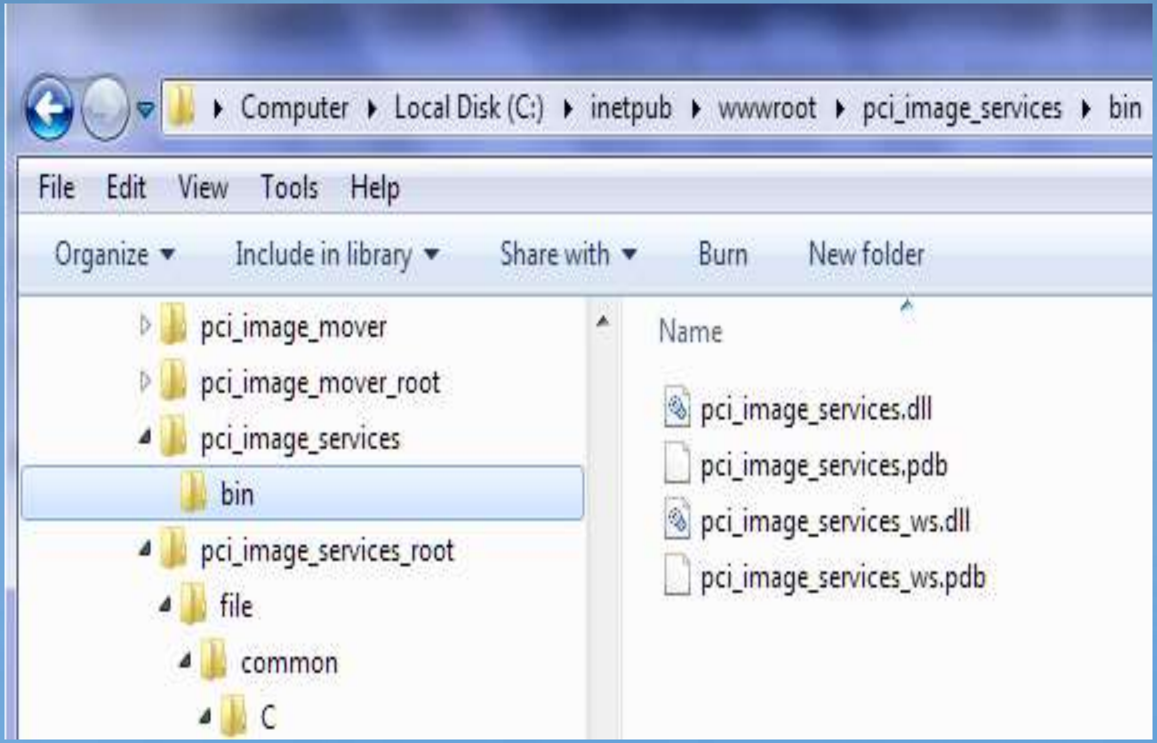

Computer ▸ Local Disk (C:) ▸ inetpub ▸ wwwroot ▸ pci_image_services ▸
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

```
pci_image_mover
pci_image_mover_root
pci_image_services
    bin
pci_image_services_root
    file
        common
            C
                ~pl_
                Dev
```

Name

- bin
- Global
- Global.asax
- log
- nc_image_service_interface
- pci_image_services
- TestWebService
- web

Web Service Virtual Root Directory


Computer ▸ Local Disk (C:) ▸ inetpub ▸ wwwroot ▸ pci_image_services ▸ bin
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder

pci_image_mover
pci_image_mover_root
pci_image_services
bin
pci_image_services_root
file
common
C

Name

pci_image_services.dll
pci_image_services.pdb
pci_image_services_ws.dll
pci_image_services_ws.pdb

Global.asax file

- A source file where developers can add application level logic into their Web applications. Located at the root of a particular Web application's virtual directory tree
- Application events such as *Application_Start*, *Application_End*, *Session_Start*, *Session_End* reside here.
- Automatically *parsed* and compiled into a dynamic .NET Framework class
- The first time any resource or URL within the application namespace is activated or requestedConfigured to automatically reject any direct URL request so that external users cannot download or view the code within

```
<%@ Application Codebehind="Global.asax.cs"
    Inherits="PBWebApp.Global" %>
```

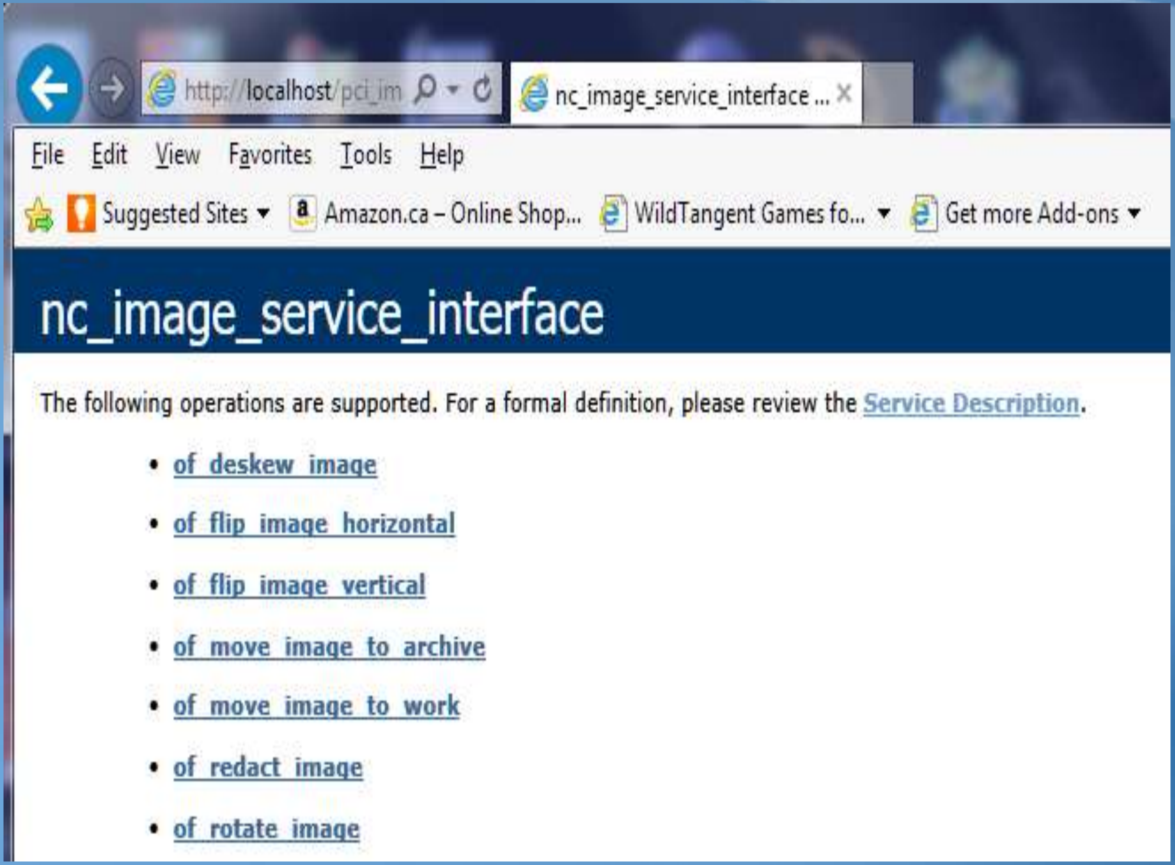
DISCO Files

- DISCO is a Microsoft technology for publishing and discovering Web Services
- DISCO files make it possible to discover the Web Services exposed on a given server
- DISCO files make it possible to discover the capabilities of each Web Service (via documentation) and how to interact with it
- DISCO files live in the Web Application’s virtual root

```
<?xml version="1.0" encoding="utf-8"?>
<discovery xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/disco/">
<contractRef ref="http://localhost/pci_image_services/nc_image_service_interface.asmx?wsdl"
      docRef="http://localhost/pci_image_services/nc_image_service_interface.asmx"
      xmlns="http://schemas.xmlsoap.org/disco/scl/" />
</discovery>
```

ASPX files

- ASP.NET provides support for Web Services with the.asmx file (a wrapper to your Web Service)
- Similar to an .aspx files
- From a browser, enter the following:
  - http://hostname/virtdirname/service.asmx
- The ASMX file lists your Web Service methods
- Clicking a link takes you to a test “harness” for that method

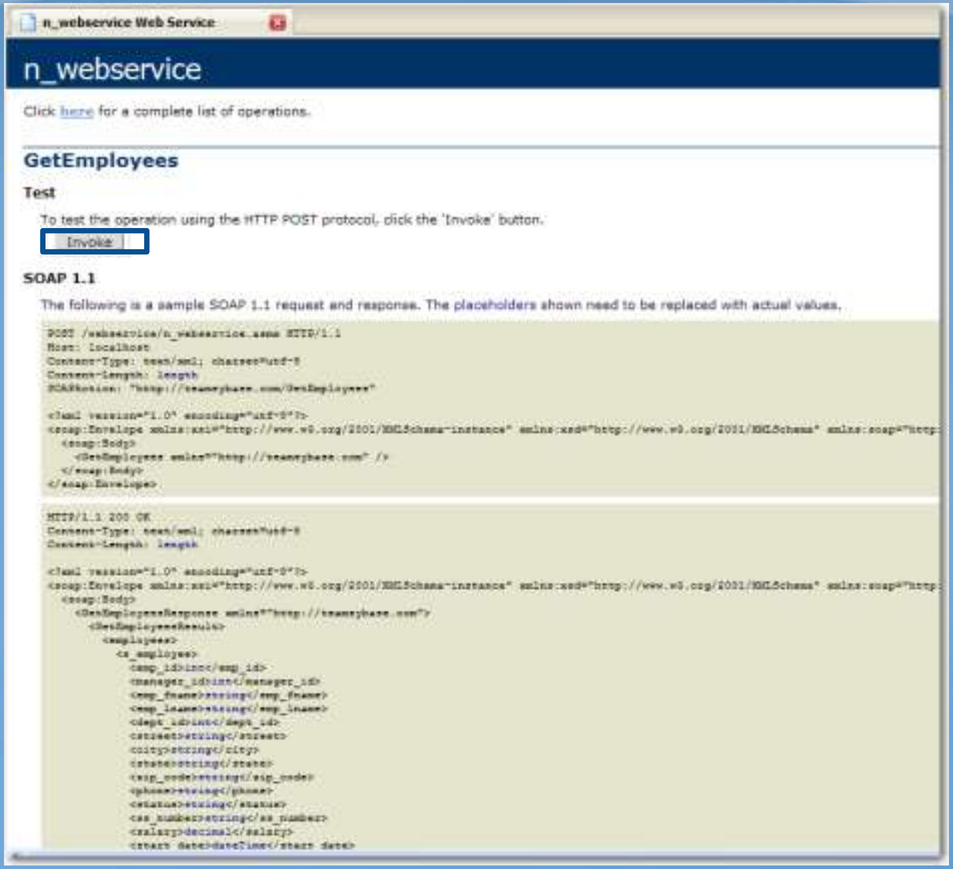

http://localhost/pci_im
nc_image_service_interface ...
File Edit View Favorites Tools Help
Suggested Sites
Amazon.ca – Online Shop...
WildTangent Games fo...
Get more Add-ons

nc_image_service_interface

The following operations are supported. For a formal definition, please review the Service Description.

- of_deskew_image
- of_flip_image_horizontal
- of_flip_image_vertical
- of_move_image_to_archive
- of_move_image_to_work
- of_redact_image
- of_rotate_image

Testing your Web Service



n_webservice

Click here for a complete list of operations.

GetEmployees

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

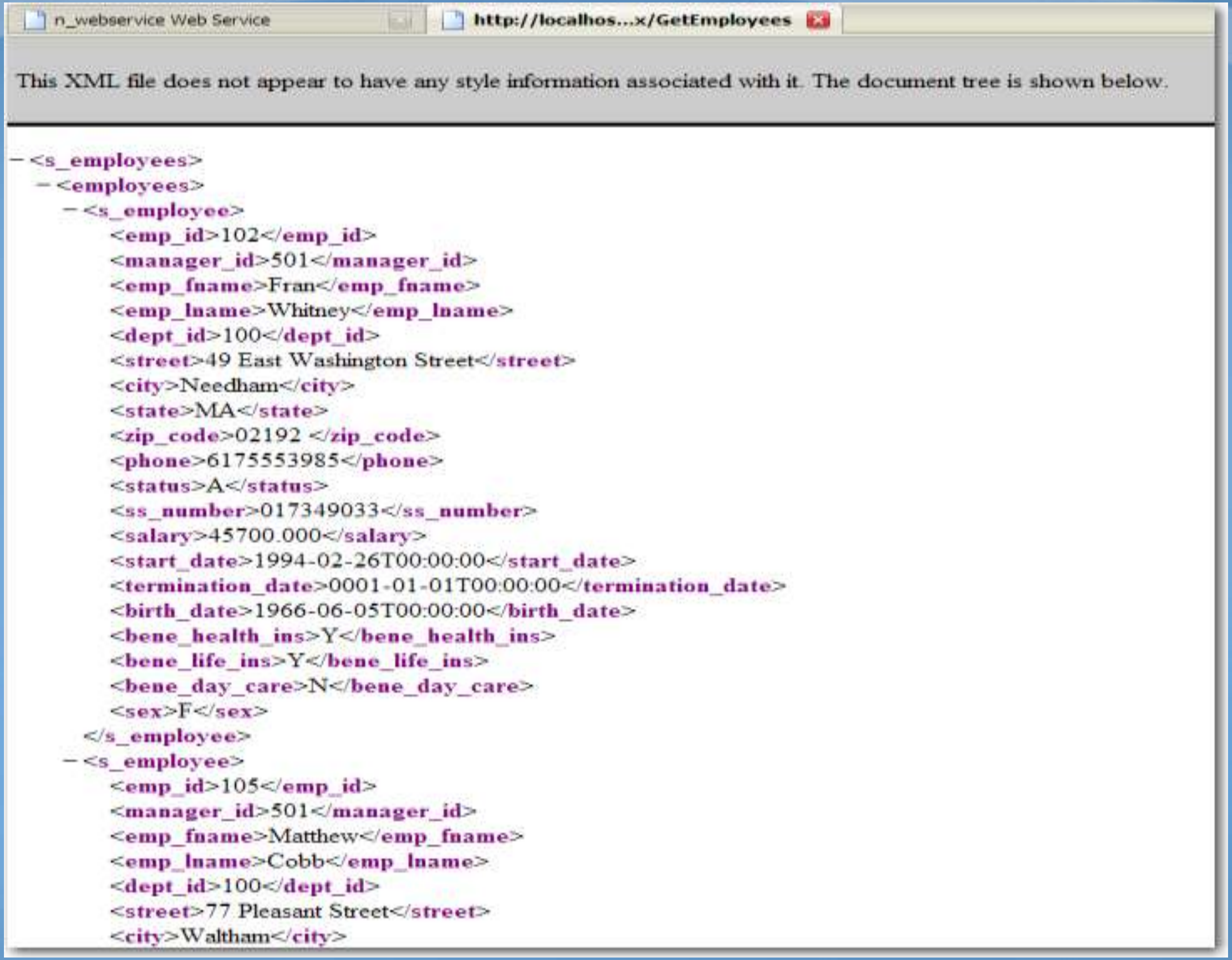
SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /webservice/n_webservice.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://teamsybase.com/GetEmployees"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http:
  <soap:Body>
    <GetEmployees xmlns="http://teamsybase.com" />
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http:
  <soap:Body>
    <GetEmployeesResponse xmlns="http://teamsybase.com">
      <GetEmployeesResult>
        <employees>
          <s_employee>
            <emp_id>int</emp_id>
            <manager_id>int</manager_id>
            <emp_fname>string</emp_fname>
            <emp_lname>string</emp_lname>
            <dept_id>int</dept_id>
            <street>string</street>
            <city>string</city>
            <state>string</state>
            <zip_code>string</zip_code>
            <phone>string</phone>
            <status>string</status>
            <ss_number>string</ss_number>
            <salary>decimal</salary>
            <start_date>dateTime</start_date>
```

Test Results



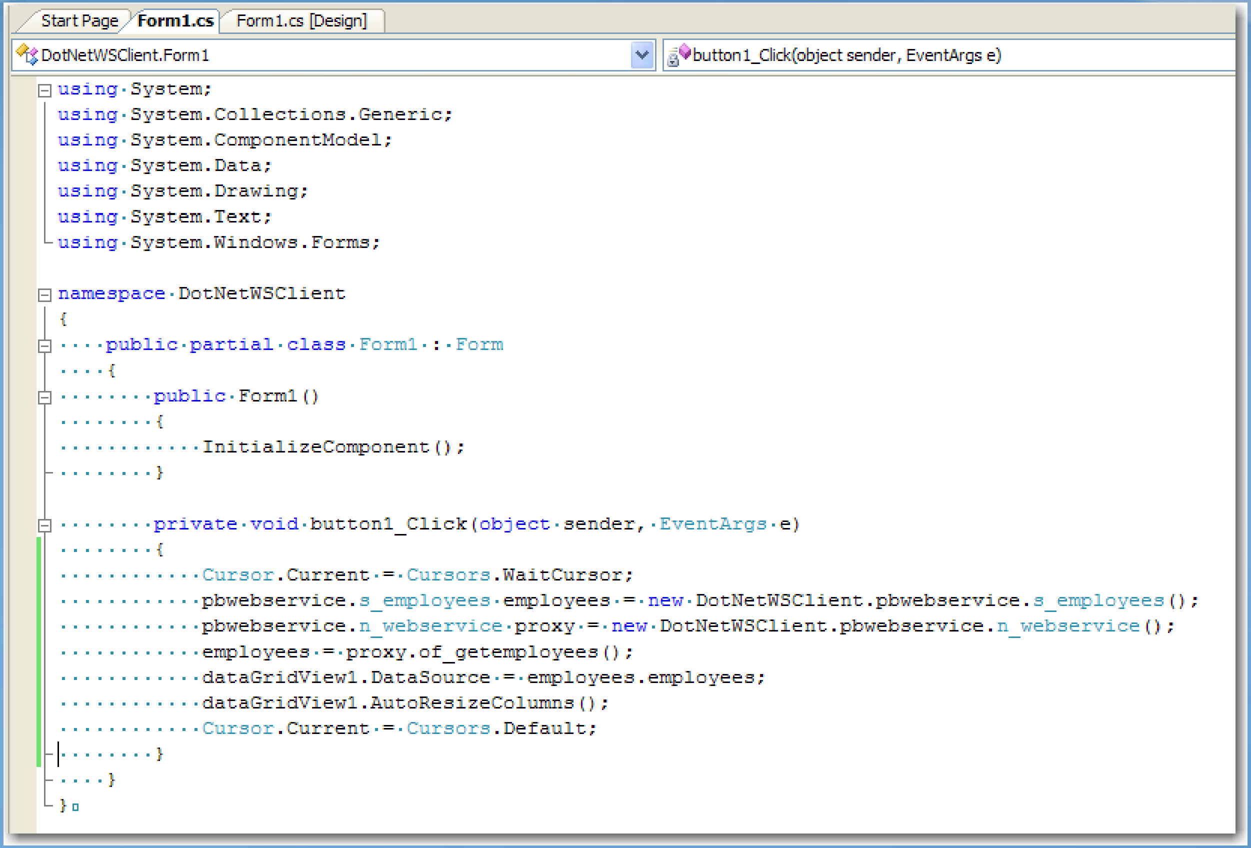
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <s_employees>
  - <employees>
    - <s_employee>
        <emp_id>102</emp_id>
        <manager_id>501</manager_id>
        <emp_fname>Fran</emp_fname>
        <emp_lname>Whitney</emp_lname>
        <dept_id>100</dept_id>
        <street>49 East Washington Street</street>
        <city>Needham</city>
        <state>MA</state>
        <zip_code>02192 </zip_code>
        <phone>6175553985</phone>
        <status>A</status>
        <ss_number>017349033</ss_number>
        <salary>45700.000</salary>
        <start_date>1994-02-26T00:00:00</start_date>
        <termination_date>0001-01-01T00:00:00</termination_date>
        <birth_date>1966-06-05T00:00:00</birth_date>
        <bene_health_ins>Y</bene_health_ins>
        <bene_life_ins>Y</bene_life_ins>
        <bene_day_care>N</bene_day_care>
        <sex>F</sex>
      </s_employee>
    - <s_employee>
        <emp_id>105</emp_id>
        <manager_id>501</manager_id>
        <emp_fname>Matthew</emp_fname>
        <emp_lname>Cobb</emp_lname>
        <dept_id>100</dept_id>
        <street>77 Pleasant Street</street>
        <city>Waltham</city>
```

Why Did We Do This?

- Interoperability
- You now have a Web Service ready to be accessed from:
  - PowerBuilder
  - Appeon
  - Java
  - C#
  - VB
  - Delphi
  - …

Sample: Calling PB Web Service from C#



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace DotNetWSClient
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Cursor.Current = Cursors.WaitCursor;
            pbwebservice.s_employees employees = new DotNetWSClient.pbwebservice.s_employees();
            pbwebservice.n_webservice proxy = new DotNetWSClient.pbwebservice.n_webservice();
            employees = proxy.of_getemployees();
            dataGridView1.DataSource = employees.employees;
            dataGridView1.AutoResizeColumns();
            Cursor.Current = Cursors.Default;
        }
    }
}
```

CONSUMING Web Services

Accessing Web Services

- Once you have the details and have built your web service consumer application, how do you call that web service's methods?
  - Create a Simple Object Access Protocol (SOAP) message
    - PowerBuilder provides two options capable of reading and writing SOAP messages
      - "Legacy" EasySoap PBNI extension
      - "New" .NET Engine

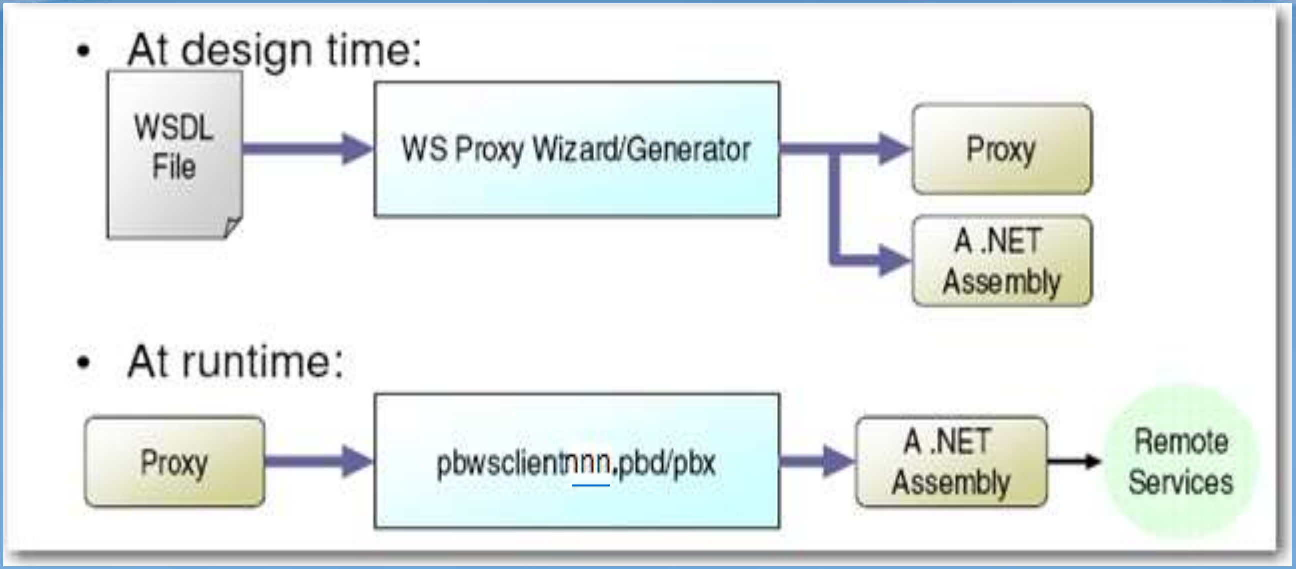
SOAP

- An XML-based communications protocol
  - “Everything is XML”
- Industry standard for cross-platform distributed messaging
- Defined by World Wide Web Consortium (W3C)

Web Service Consumption

- Consuming a Web Service from a PowerBuilder client requires a Web Service proxy.
- A network connection is needed, but Web Services require a special Soap Connection.
- The Web Service is similar to an NVUO as a container of methods which could be called via SOAP (Simple Object Access Protocol).
- Invoking Web services through SOAP requires:
  - Serialization and deserialization of data types
  - The building and parsing of XML-based SOAP messages
- A PowerBuilder Web Service client proxy performs these tasks for you eliminating the need to have extensive knowledge of :
  - The SOAP specification and schema
  - The XML Schema specification
  - The WSDL specification and schema

.Net Web Service Engine Flow



Prerequisites

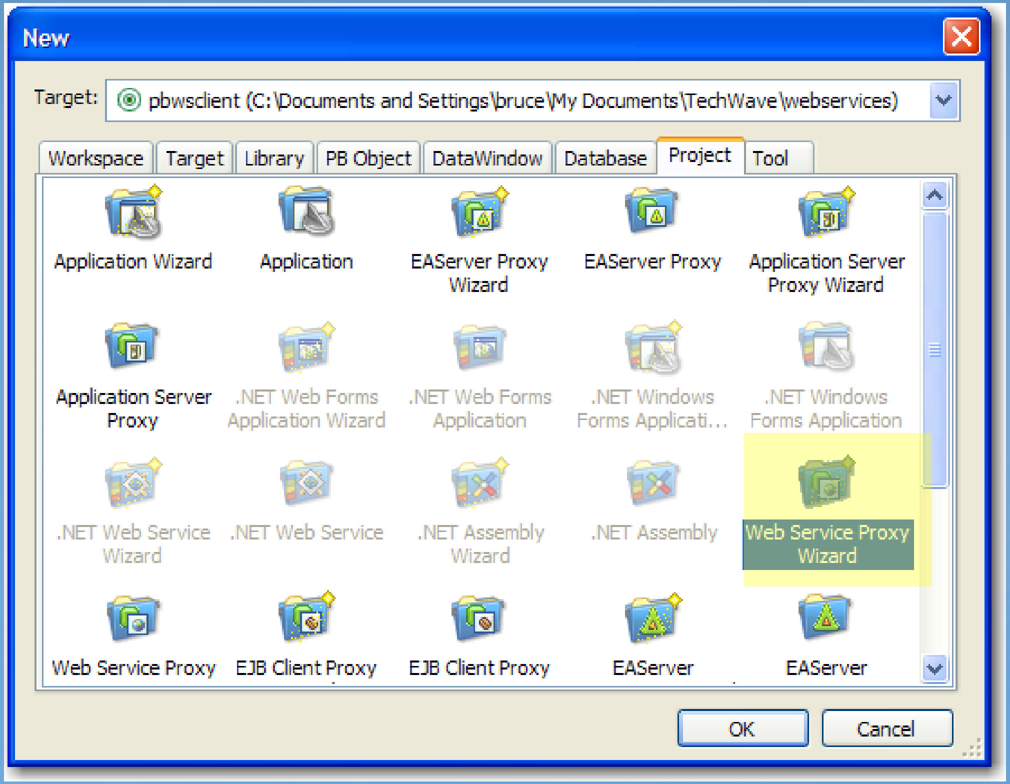
PB 11.x/11.5.x: .NET 2.0 Framework **SDK** on development machine + .NET 2.0 Framework (Runtime) on both development and deployment machine.

PB 12.x: .NET 3.5 Framework **SDK** on development machine + .NET 3.5 Framework (Runtime) on both development and deployment machine.

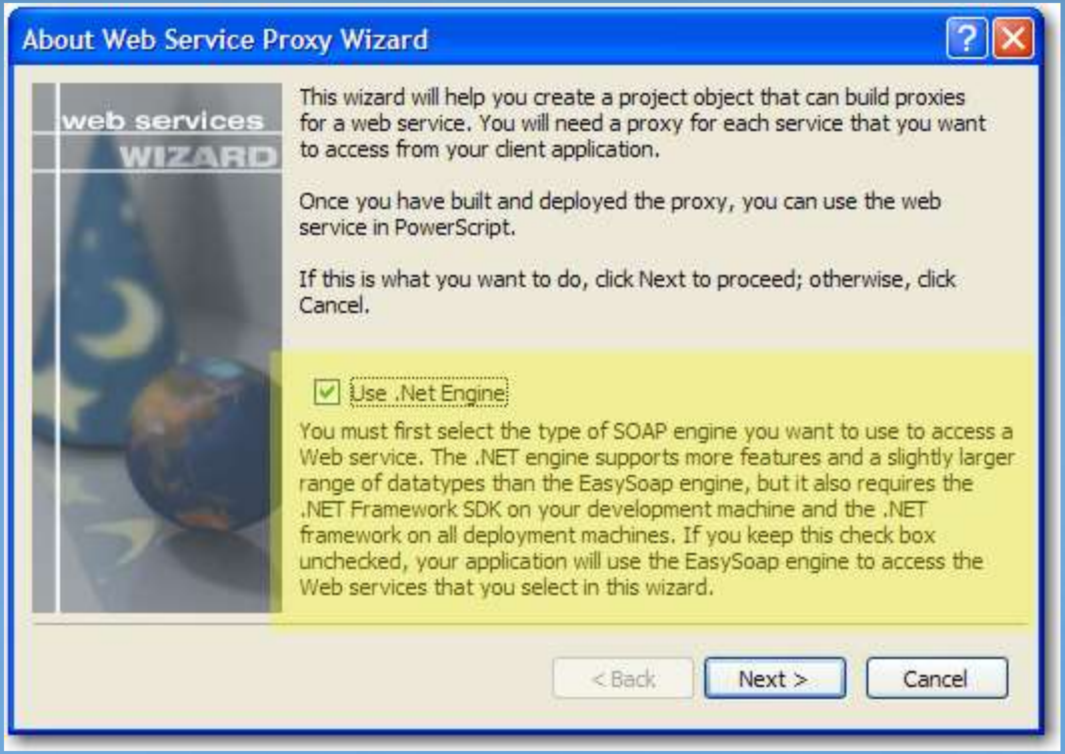
PB 12.5.x: .NET 4.0 Framework **SDK** on development machine + .NET 4.0 Framework (Runtime) on both development and deployment machine.

PB 12.6.x: .NET 4.0 Framework **SDK** on development machine + .NET 4.5 Framework (Runtime) on both development and deployment machine.

Web Service Proxy Wizard



Choose the Web Service Engine



This wizard will help you create a project object that can build proxies for a web service. You will need a proxy for each service that you want to access from your client application.

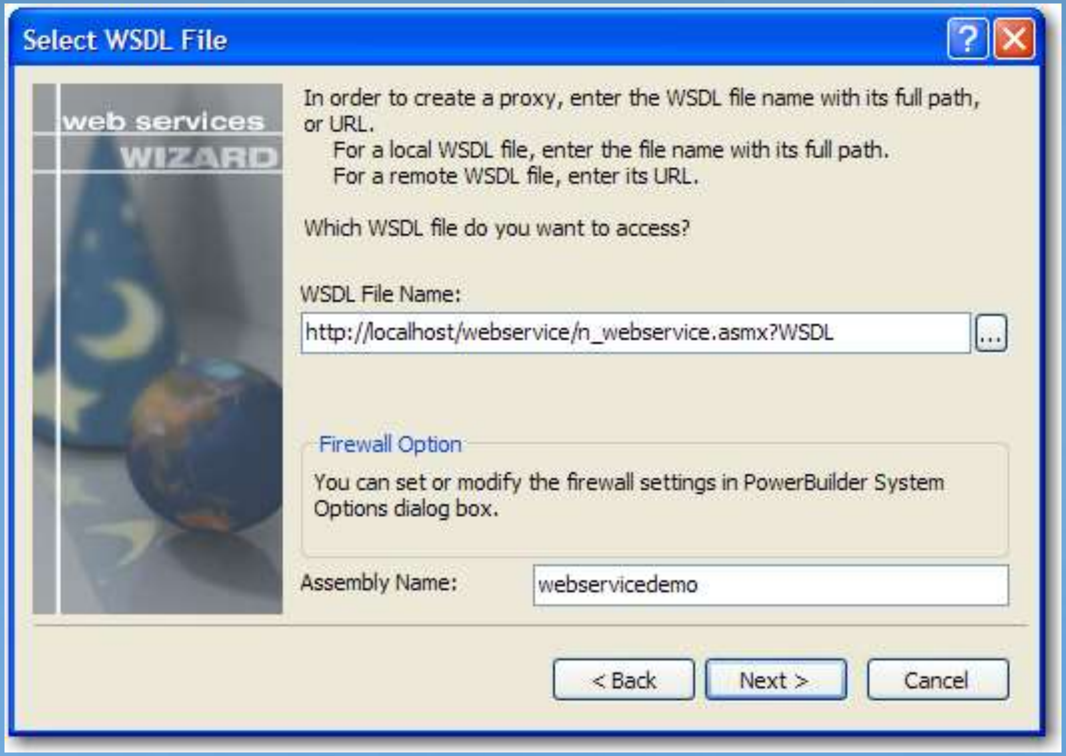
Once you have built and deployed the proxy, you can use the web service in PowerScript.

If this is what you want to do, click Next to proceed; otherwise, click Cancel.

☑ Use .Net Engine

You must first select the type of SOAP engine you want to use to access a Web service. The .NET engine supports more features and a slightly larger range of datatypes than the EasySoap engine, but it also requires the .NET Framework SDK on your development machine and the .NET framework on all deployment machines. If you keep this check box unchecked, your application will use the EasySoap engine to access the Web services that you select in this wizard.

Specify WSDL



In order to create a proxy, enter the WSDL file name with its full path, or URL.
For a local WSDL file, enter the file name with its full path.
For a remote WSDL file, enter its URL.

Which WSDL file do you want to access?

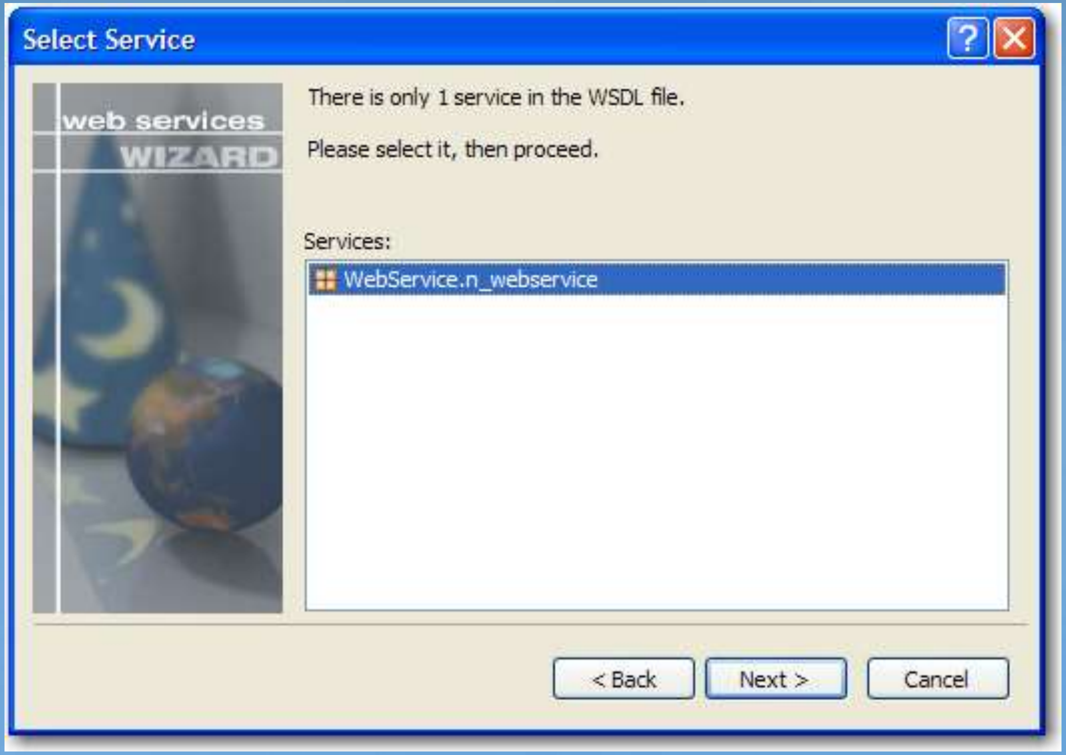
WSDL File Name: http://localhost/webservice/n_webservice.asmx?WSDL

Firewall Option

You can set or modify the firewall settings in PowerBuilder System Options dialog box.

Assembly Name: webservicedemo

Select a Service From WSDL



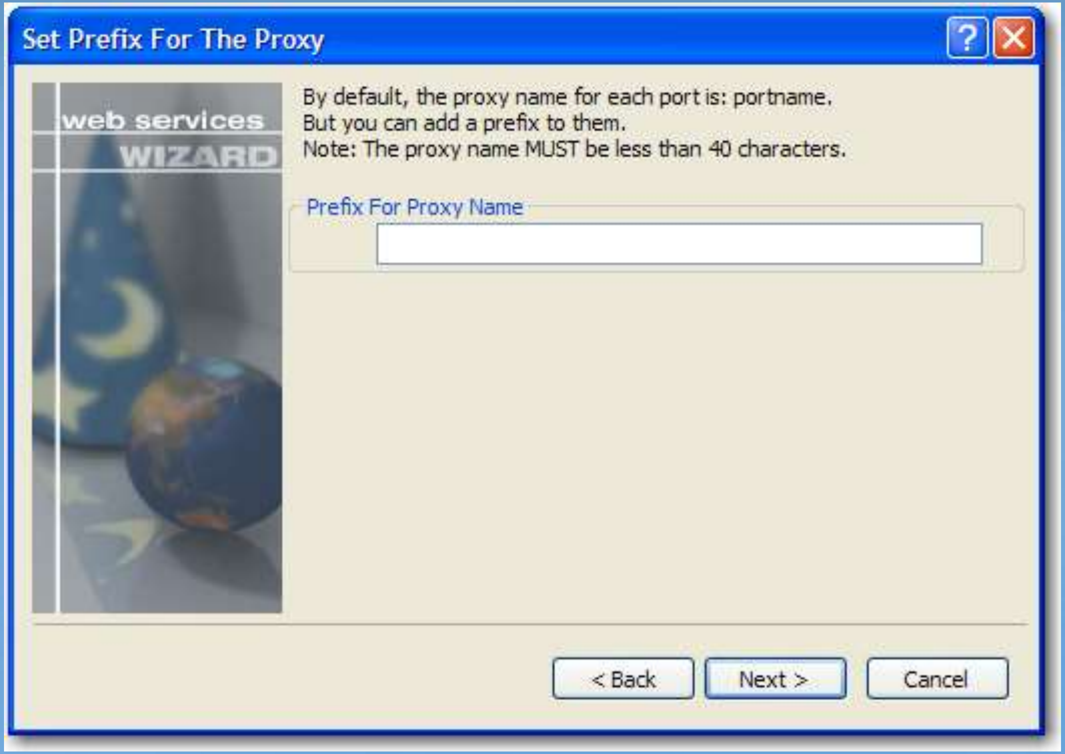
There is only 1 service in the WSDL file.

Please select it, then proceed.

Services:

WebService.n_webservice

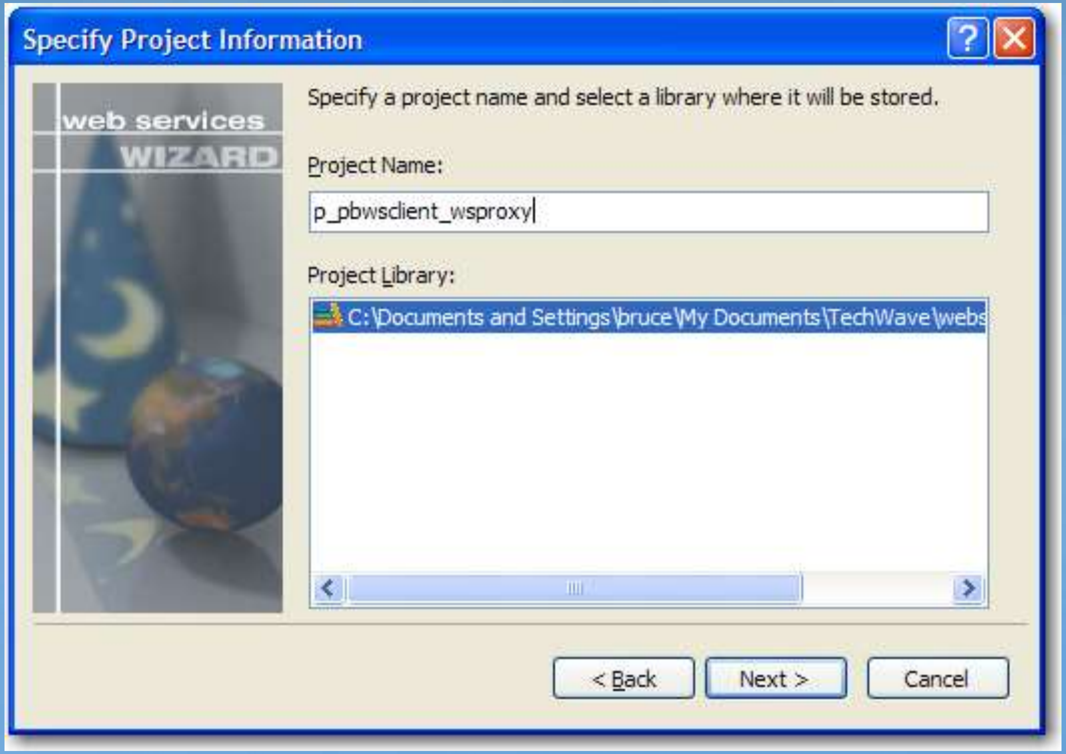
Define Prefix for Proxy (Optional)



By default, the proxy name for each port is: portname.
But you can add a prefix to them.
Note: The proxy name MUST be less than 40 characters.

Prefix For Proxy Name

Specify Project Name and Library



Specify Project Information

Specify a project name and select a library where it will be stored.

Project Name:

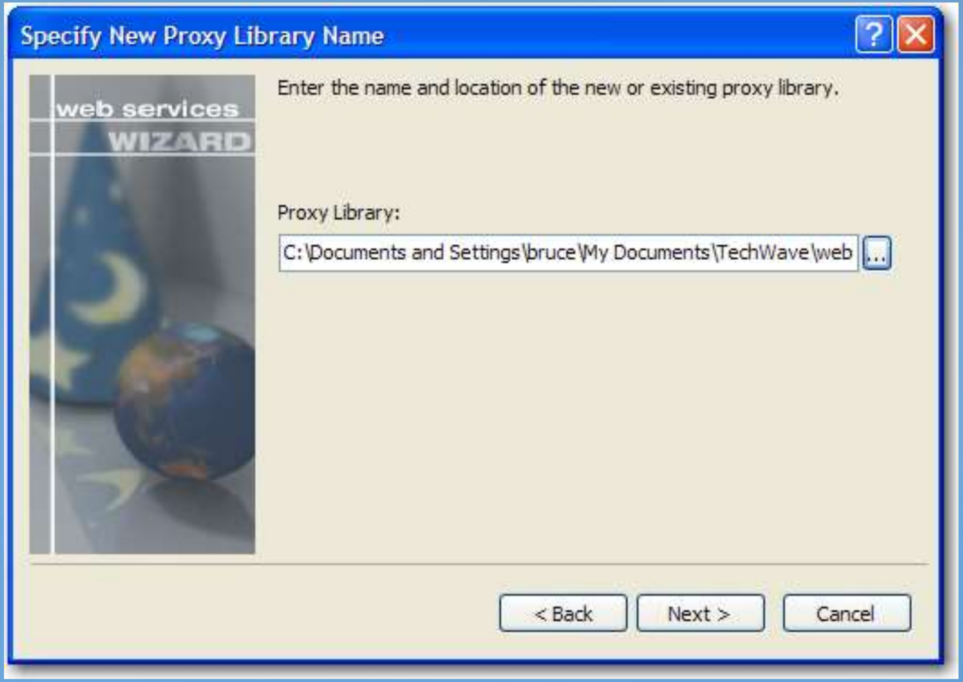
p_pbwsclient_wsproxy

Project Library:

C:\Documents and Settings\bruce\My Documents\TechWave\webs

Specify PBL for generated proxy(ies)

- It is a *good* practice to store your proxies in a separate PBL in your library list



Specify New Proxy Library Name

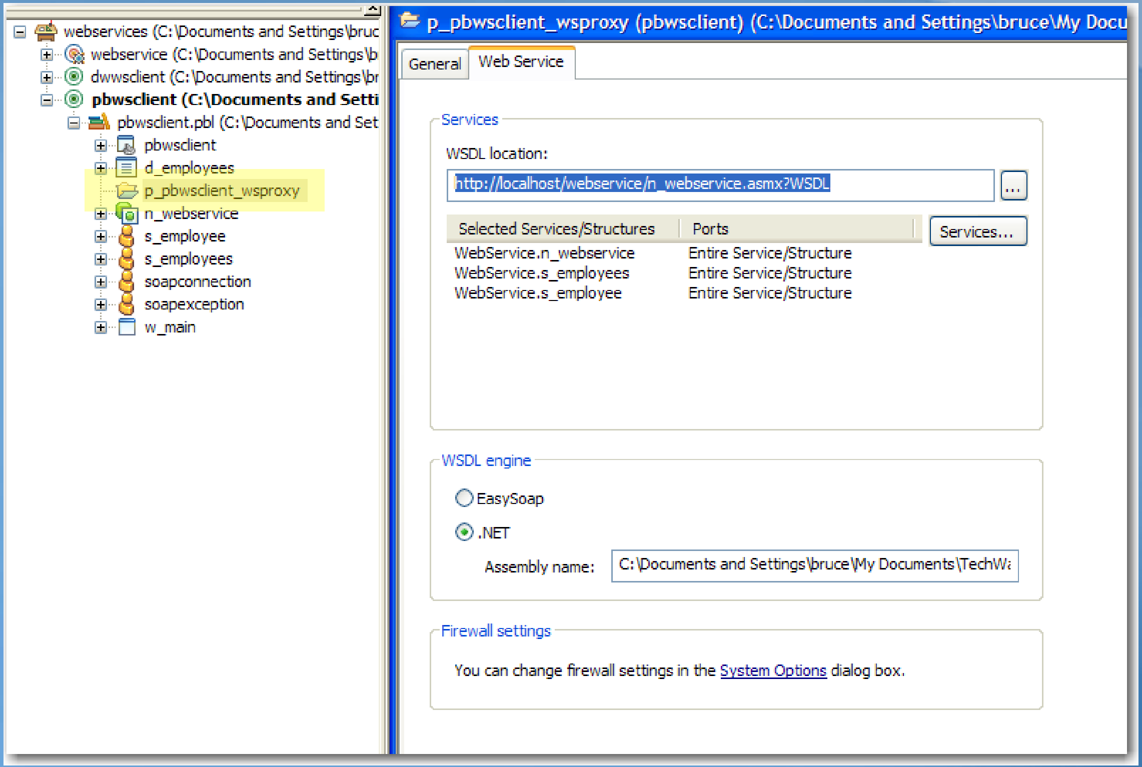
Enter the name and location of the new or existing proxy library.

Proxy Library:

C:\Documents and Settings\bruce\My Documents\TechWave\web

Proxy Project

- Upon completion of the WSP Wizard, the new project is visible in the System Tree, and the project will be open in the painter
- Next, deploy the project to have the PB IDE build the appropriate proxy components!



webservices (C:\Documents and Settings\bruc
webservice (C:\Documents and Settings\b
dwwsclient (C:\Documents and Settings\br
pbwsclient (C:\Documents and Setti
pbwsclient.pbl (C:\Documents and Set
pbwsclient
d_employees
p_pbwsclient_wsproxy
n_webservice
s_employee
s_employees
soapconnection
soapexception
w_main

p_pbwsclient_wsproxy (pbwsclient) (C:\Documents and Settings\bruce\My Docu

General | Web Service

Services

WSDL location: http://localhost/webservice/n_webservice.asmx?WSDL

Selected Services/Structures	Ports
WebService.n_webservice	Entire Service/Structure
WebService.s_employees	Entire Service/Structure
WebService.s_employee	Entire Service/Structure

Services...

WSDL engine

( ) EasySoap
(●) .NET

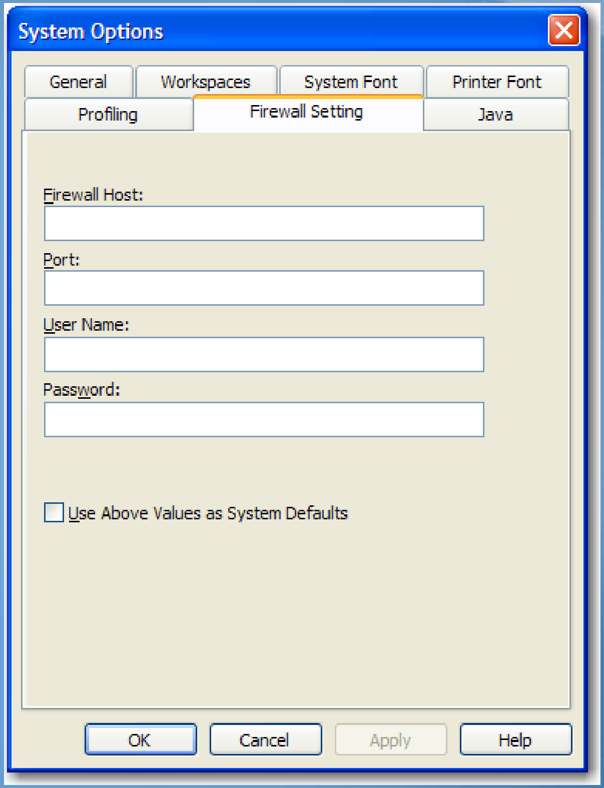
Assembly name: C:\Documents and Settings\bruce\My Documents\TechW

Firewall settings

You can change firewall settings in the System Options dialog box.

Use Proxy Servers?

- If your company uses a Proxy Server to bridge between you and the Firewall, visit the Tools → System Options dialog
- Input the name of your Proxy Server, port, your user id and password to that proxy server
- This is for design-time Internet connections only



Firewall Host:

Port:

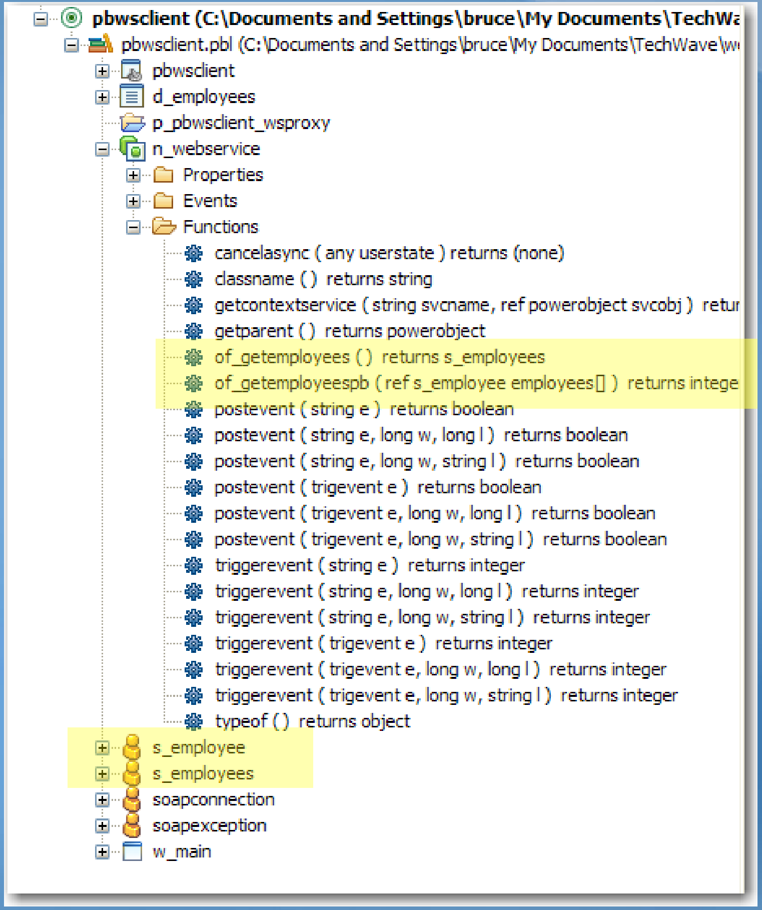
User Name:

Password:

[ ] Use Above Values as System Defaults

The Web Service Proxy

- System Tree (expanded), following the deploy of the proxy project
- The function(s) available from the Web Service will be visible under the proxy
- Be sure you understand that the proxy project is separate from the actual proxy object



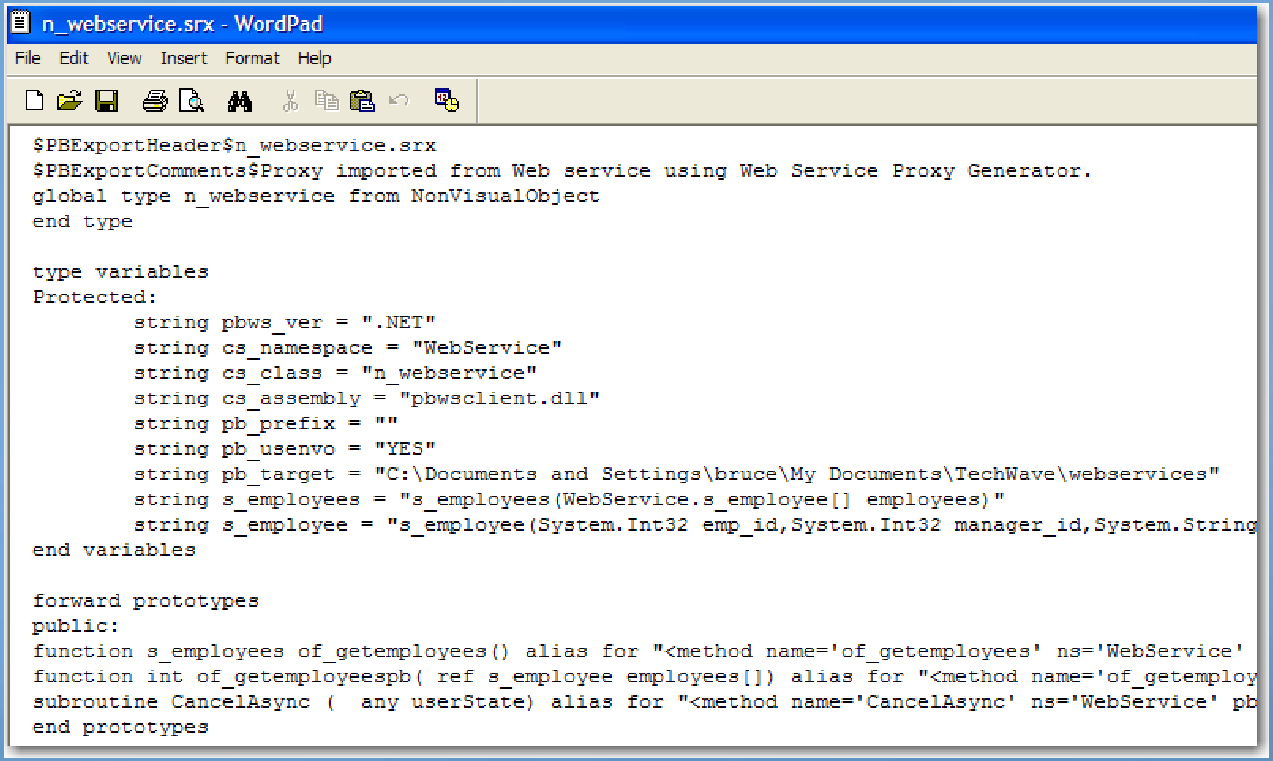
```
pbwsclient (C:\Documents and Settings\bruce\My Documents\TechWa
  pbwsclient.pbl (C:\Documents and Settings\bruce\My Documents\TechWave\w
    pbwsclient
    d_employees
    p_pbwsclient_wsproxy
    n_webservice
      Properties
      Events
      Functions
        cancelasync ( any userstate ) returns (none)
        classname ( ) returns string
        getcontextservice ( string svcname, ref powerobject svcobj ) retur
        getparent ( ) returns powerobject
        of_getemployees ( ) returns s_employees
        of_getemployeespb ( ref s_employee employees[] ) returns intege
        postevent ( string e ) returns boolean
        postevent ( string e, long w, long l ) returns boolean
        postevent ( string e, long w, string l ) returns boolean
        postevent ( trigevent e ) returns boolean
        postevent ( trigevent e, long w, long l ) returns boolean
        postevent ( trigevent e, long w, string l ) returns boolean
        triggerevent ( string e ) returns integer
        triggerevent ( string e, long w, long l ) returns integer
        triggerevent ( string e, long w, string l ) returns integer
        triggerevent ( trigevent e ) returns integer
        triggerevent ( trigevent e, long w, long l ) returns integer
        triggerevent ( trigevent e, long w, string l ) returns integer
        typeof ( ) returns object
    s_employee
    s_employees
    soapconnection
    soapexception
    w_main
```

Use of Aliases in Proxy

- PowerBuilder is **not** case sensitive
- XML (SOAP) and .NET **are** case sensitive
- To get around that difference, each method in the proxy uses an alias
- The string that follows “alias for” contains the case-sensitive name and the signature of the corresponding XML or SOAP method

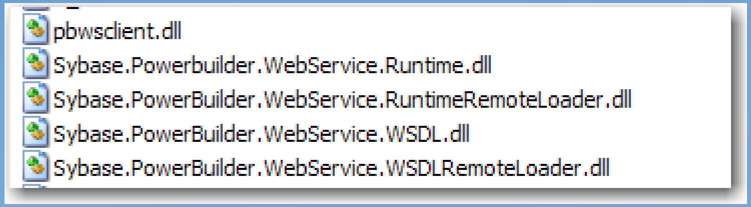
Exported Web Service Proxy

- Note the “alias for” clauses in the function or subroutine declarations

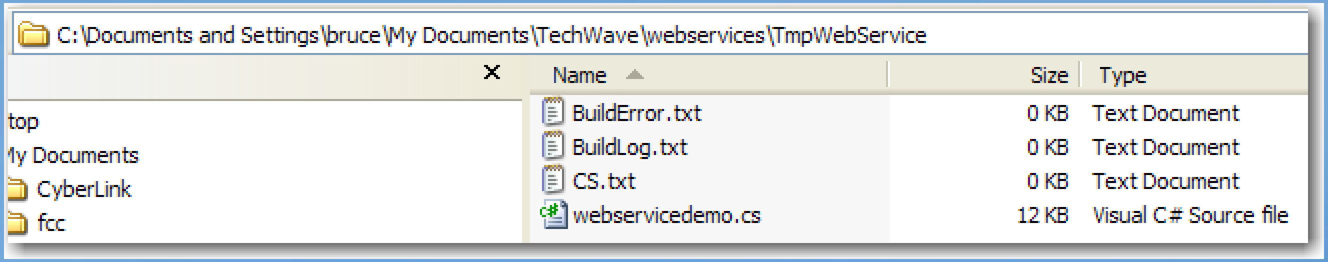


```
$PBExportHeader$n_webservice.srx
$PBExportComments$Proxy imported from Web service using Web Service Proxy Generator.
global type n_webservice from NonVisualObject
end type

type variables
Protected:
        string pbws_ver = ".NET"
        string cs_namespace = "WebService"
        string cs_class = "n_webservice"
        string cs_assembly = "pbwsclient.dll"
        string pb_prefix = ""
        string pb_usenvo = "YES"
        string pb_target = "C:\Documents and Settings\bruce\My Documents\TechWave\webservices"
        string s_employees = "s_employees(WebService.s_employee[] employees)"
        string s_employee = "s_employee(System.Int32 emp_id,System.Int32 manager_id,System.String
end variables

forward prototypes
public:
function s_employees of_getemployees() alias for "<method name='of_getemployees' ns='WebService'
function int of_getemployeespb( ref s_employee employees[]) alias for "<method name='of_getemploy
subroutine CancelAsync (  any userState) alias for "<method name='CancelAsync' ns='WebService' pb
end prototypes
```

.Net Web Service Engine – Files Created from Proxy



pbwsclient.dll
Sybase.Powerbuilder.WebService.Runtime.dll
Sybase.PowerBuilder.WebService.RuntimeRemoteLoader.dll
Sybase.PowerBuilder.WebService.WSDL.dll
Sybase.PowerBuilder.WebService.WSDLRemoteLoader.dll



C:\Documents and Settings\bruce\My Documents\TechWave\webservices\TmpWebService

top
ly Documents
CyberLink
fcc

Name	Size	Type
BuildError.txt	0 KB	Text Document
BuildLog.txt	0 KB	Text Document
CS.txt	0 KB	Text Document
webservicedemo.cs	12 KB	Visual C# Source file

Web Service Runtime Engines

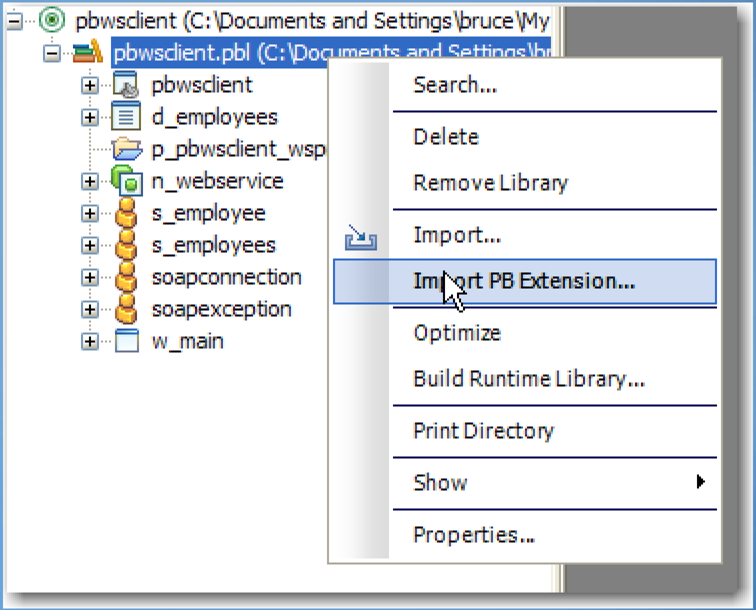
- EasySoap Engine – pbsoapclient*nnn*.pbd/pbx
  - This engine is backward compatible with the PB9=>PB12.6 Web Service engine
  - It can work on machines that don’t have the .NET framework
- .NET Engine – pbwsclient*nnn*.pbd/pbx
  - This is new .NET SOAP engine
- Both of the above define two classes:
  - SoapConnection
  - SoapException

What Was that PBX Reference?

- An extension to PowerBuilder functionality created using the PowerBuilder Native Interface (PBNI)
- *Before* 10.5, a PBNI extension (*.pbx or *.dll) developer had to:
  - Use the pbx2pbd utility to create a PBD file from an extension
  - Be sure to put the extension file (PBX) in the application's search path ***and*** add the PBD file to the target's library list
- Now there are fewer steps:
  - Import the *.pbx directly into your *.pbl’s using the System Tree
  - Must still deploy the extension in the application’s path

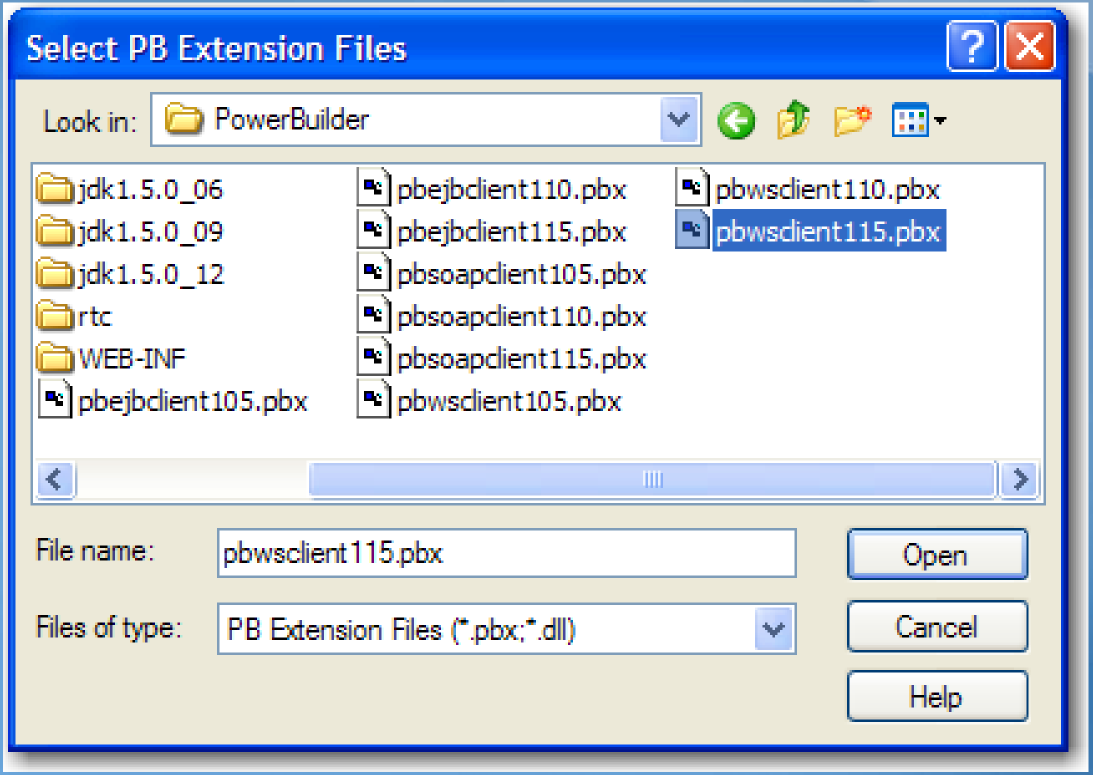
Importing PowerBuilder Extensions

- *Prior* to PB 10.5, to gain a SoapConnection, you needed to add pbsoap*nnn*.pbd to your library list
- Pbsoap*nnn*.pbd was a PBNI extension for EasySoap
- Now you can import the *.pbx directly to a PBL
- To do so, right-click over a PBL



Choosing the SOAP Flavour

- ***PbwsclientNNN.pbx*** is the extension for the ***.NET Web Service*** engine
- ***PbsoapclientNNN.pbx*** is the extension for ***EasySoap***



File name: pbwsclient115.pbx

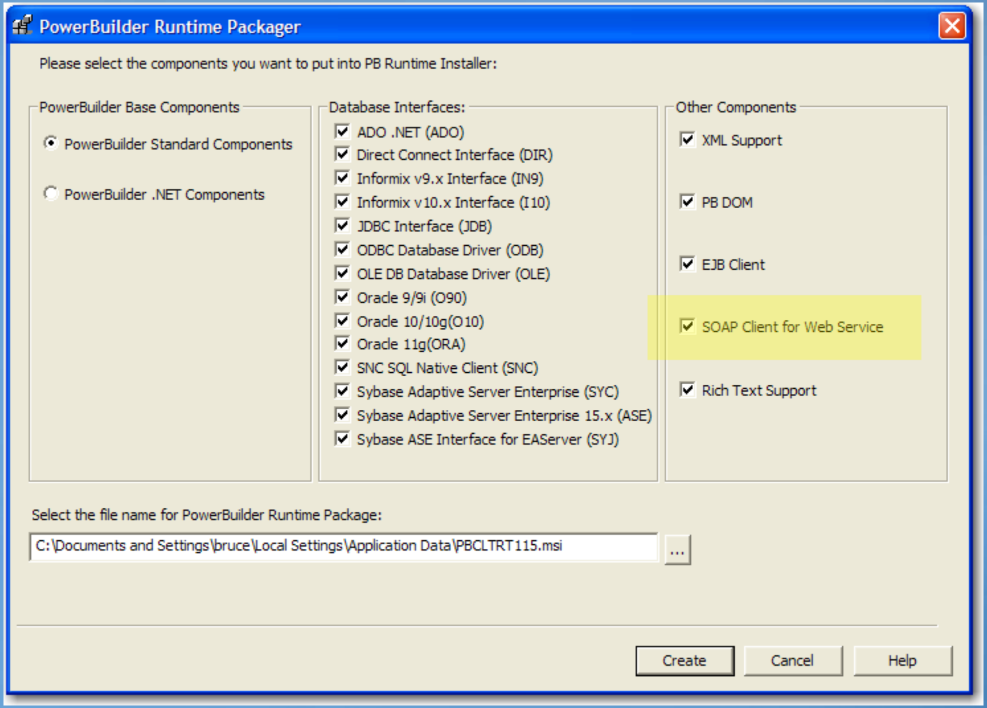
Files of type: PB Extension Files (*.pbx;*.dll)

Important Points About These Imports

- Using *pbwsclientnnn.pbx* requires the .NET 2.0, 3.5 4.0 or 4.5 Framework on design-time and runtime machines. **Note**: .Net 4.5 can <u>not</u> be used with PB 12.5.x or lower!
- Both extension files contain the same objects, and you use these objects and their methods in similar ways
- The *Sybase\Shared\PowerBuilder* directory contains PBD versions of the extension files that may still be used instead of importing the extensions (add PBDs to library list instead)
- When you create a Web service client application, you must deploy the extension file that you use along with the client executable to a directory in the application's search path
  - The Runtime Packager tool automatically includes the extension files required by your Web service applications

PowerBuilder Runtime Packager

- Will help to ensure PBNI extensions are deployed to your end users:



PowerBuilder Runtime Packager

Please select the components you want to put into PB Runtime Installer:

PowerBuilder Base Components
- [✓ PowerBuilder Standard Components]
- PowerBuilder .NET Components

Database Interfaces:
- ✓ ADO .NET (ADO)
- ✓ Direct Connect Interface (DIR)
- ✓ Informix v9.x Interface (IN9)
- ✓ Informix v10.x Interface (I10)
- ✓ JDBC Interface (JDB)
- ✓ ODBC Database Driver (ODB)
- ✓ OLE DB Database Driver (OLE)
- ✓ Oracle 9/9i (O90)
- ✓ Oracle 10/10g(O10)
- ✓ Oracle 11g(ORA)
- ✓ SNC SQL Native Client (SNC)
- ✓ Sybase Adaptive Server Enterprise (SYC)
- ✓ Sybase Adaptive Server Enterprise 15.x (ASE)
- ✓ Sybase ASE Interface for EAServer (SYJ)

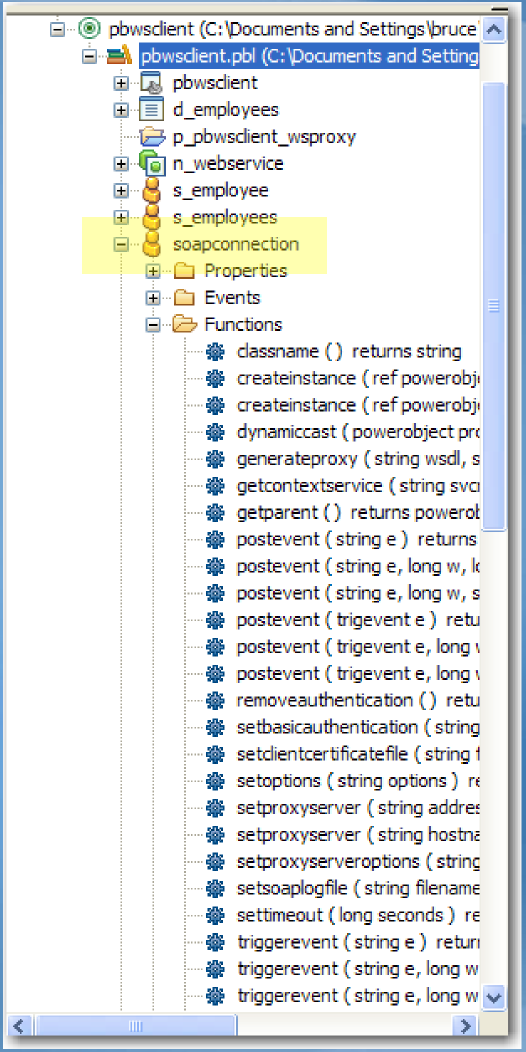
Other Components
- ✓ XML Support
- ✓ PB DOM
- ✓ EJB Client
- ✓ SOAP Client for Web Service
- ✓ Rich Text Support

Select the file name for PowerBuilder Runtime Package:

C:\Documents and Settings\bruce\Local Settings\Application Data\PBCLTRT115.msi

Result of PBX Import

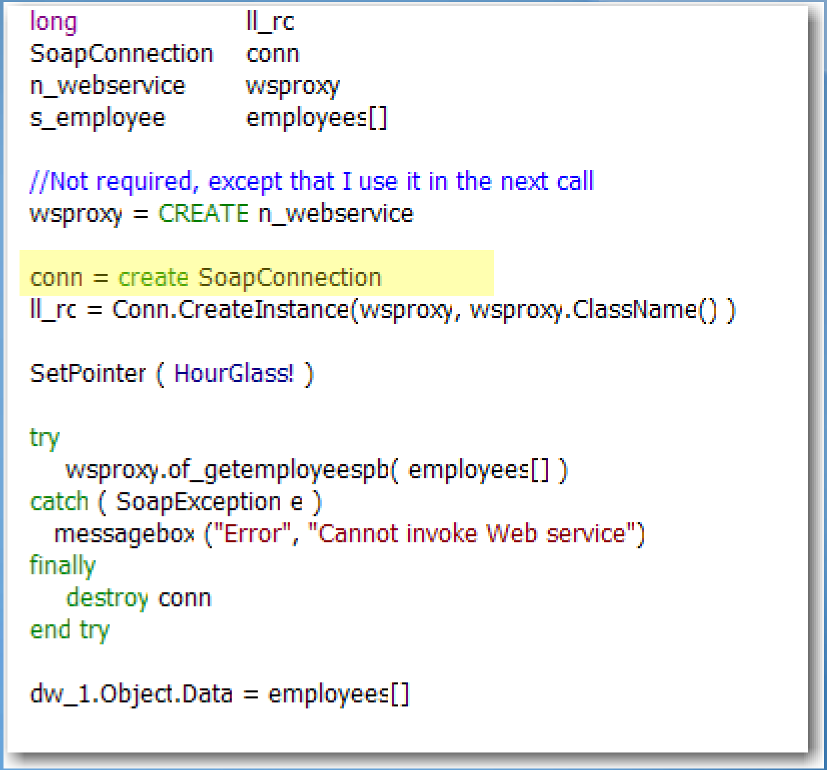
- Following the import of the .NET extension, you will see two new objects in the System Tree:
  - SoapConnection
  - SoapException
- Notice the CreateInstance method in soapconnection



```
pbwsclient (C:\Documents and Settings\bruce
  pbwsclient.pbl (C:\Documents and Setting
    pbwsclient
    d_employees
    p_pbwsclient_wsproxy
    n_webservice
    s_employee
    s_employees
    soapconnection
      Properties
      Events
      Functions
        classname ( ) returns string
        createinstance ( ref powerobj
        createinstance ( ref powerobj
        dynamiccast ( powerobject pr
        generateproxy ( string wsdl, s
        getcontextservice ( string svc
        getparent ( ) returns powerol
        postevent ( string e ) returns
        postevent ( string e, long w, l
        postevent ( string e, long w, s
        postevent ( trigevent e ) retu
        postevent ( trigevent e, long
        postevent ( trigevent e, long
        removeauthentication ( ) retu
        setbasicauthentication ( string
        setclientcertificatefile ( string
        setoptions ( string options ) re
        setproxyserver ( string addres
        setproxyserver ( string hostna
        setproxyserveroptions ( string
        setsoaplogfile ( string filename
        settimeout ( long seconds ) re
        triggerevent ( string e ) return
        triggerevent ( string e, long w
        triggerevent ( string e, long w
```

Connection Code

- After importing the SoapConnection object, you are ready to write code to communicate with the Web Service
- Begin by instantiating the soapconnection object:



```
long                ll_rc
SoapConnection      conn
n_webservice        wsproxy
s_employee          employees[]

//Not required, except that I use it in the next call
wsproxy = CREATE n_webservice

conn = create SoapConnection
ll_rc = Conn.CreateInstance(wsproxy, wsproxy.ClassName() )

SetPointer ( HourGlass! )

try
    wsproxy.of_getemployeespb( employees[] )
catch ( SoapException e )
   messagebox ("Error", "Cannot invoke Web service")
finally
    destroy conn
end try

dw_1.Object.Data = employees[]
```

SoapConnection Methods

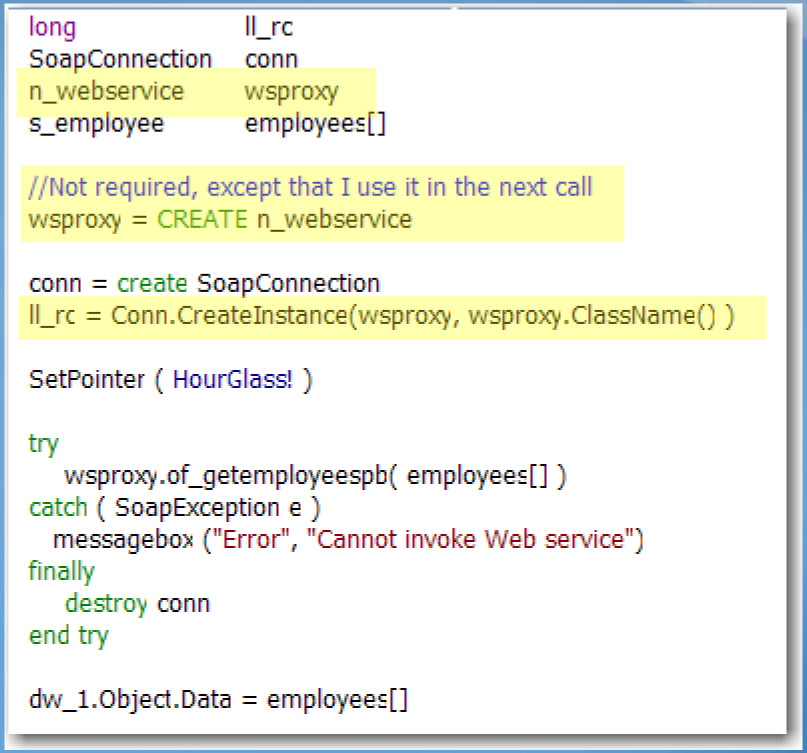
- New methods that were added to SoapConnection in PowerBuilder v10.5 & higher
- *Prior* to PB v10.5, most connection options were passed in as arguments to the SetOptions( ) method of SoapConnection
- Now, there are individual methods you may call
- For EasySoap use:
  - SetSoapLogFile( )
  - SetTimeout( )
  - UseConnectionCache( )

Securing Web Services

- Securing Web Services has been secondary from the beginning of the specification
- However, you have seen some security measures are in place
  - The ability to secure a Web Service:
    - Basic authentication (user id and password)
    - Use of digital certificates
- You may also secure a Web Service through the use of SOAP Headers
- This section will show you how to use SOAP Header authentication

Making the Web Service Call

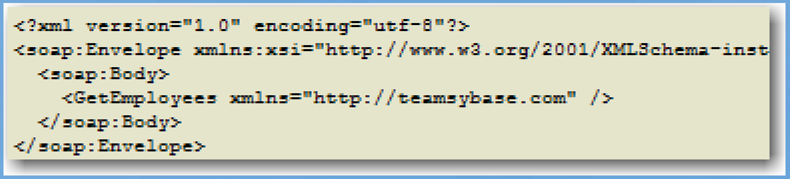
- Declare a reference variable of type Web Service proxy
- Create an instance of the Web Service proxy



```
long              ll_rc
SoapConnection    conn
n_webservice      wsproxy
s_employee        employees[]

//Not required, except that I use it in the next call
wsproxy = CREATE n_webservice

conn = create SoapConnection
ll_rc = Conn.CreateInstance(wsproxy, wsproxy.ClassName() )

SetPointer ( HourGlass! )

try
    wsproxy.of_getemployeespb( employees[] )
catch ( SoapException e )
   messagebox ("Error", "Cannot invoke Web service")
finally
    destroy conn
end try

dw_1.Object.Data = employees[]
```

Sample SOAP Message

- Use of SOAP Headers is optional
- Below is an example of calling a Web Service method named *GetEmployees*



```xml
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
  <soap:Body>
    <GetEmployees xmlns="http://teamsybase.com" />
  </soap:Body>
</soap:Envelope>
```

Note about SOAP Headers

- Be aware that authenticating callers by encoding plaintext user names and passwords in SOAP Headers is not secure
- To secure SOAP Header information you could:
  - Encrypt SOAP messages by writing a SOAP extension that unencrypts requests and encrypts responses
  - Use SSL / HTTPS to publish the Web Service

Web Service DATAWINDOWS

Web Service as a DataWindow Data Source

- In PowerBuilder v11.0 and higher, you can use a Web Service as the data source for a DataWindow object
  - Supports a disconnected client model
  - Eliminates requirement that database vendor’s client software reside on end-user machine
  - Web Service ‘result set’ support

Web Service DataWindows

- Are an extension of the Web Services support that has been in PowerBuilder since Version 9.0
  - Uses the .NET Web Service engine
  - Creates a .NET assembly to do the work behind the scenes
- Web Service DataWindows are modeled on the way the Stored Procedure DataWindow works
- Two components:
  - Design-time component that allows you to browse, select a Web Service, then a specific method
  - Run-time component that
    - Retrieves data and maps to DataWindow columns
    - Updates data mapping columns to Web Service method inputs

Restrictions on Web Service Methods

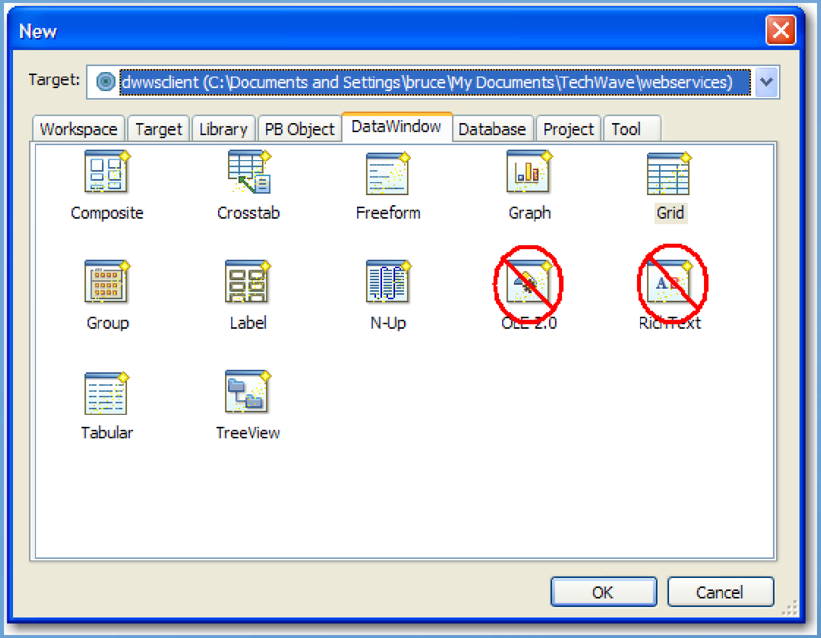
- The return of the Web Service method must be:
  - **Simple** data types such as Integer, String, Date, Time, Double, Blob (base64Binary), Boolean, Decimal, Float, Long, DateTime, Char (byte), etc
    - DWO will have a single column/row
  - Array of simple types
    - DWO will nave *n* rows of a single column depending on the size of the array
  - Structure of simple types
    - DWO will have 1 row with *n* columns depending on the number of variables in the structure
  - Array of structure
    - DWO will have *n* rows, *n* columns
- Some Web Service methods *will not* work with the DataWindow

Other Web Service DataWindow Notes

- Web Service DataWindows will allow *Retrieval Arguments* (If the Web Service method has input parameters)
- Query Mode is **not** supported
- The Web Service method metadata is used to create the actual DataWindow object
- You will use the Retrieve( ) & Update ( ) methods just as you do today!

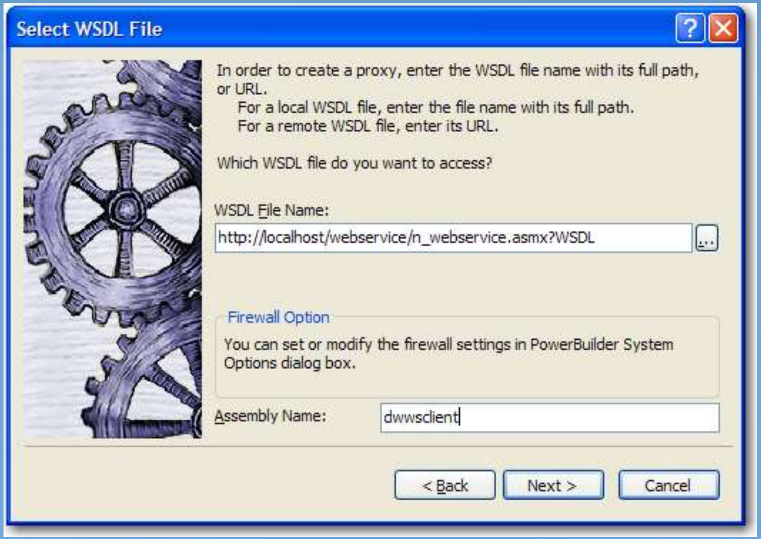
Supported Presentation Styles

- Most Presentation Styles are supported:
- RichText and OLE are **not** supported



Selecting a WSDL File …

- First, select a WSDL file describing the Web Service



In order to create a proxy, enter the WSDL file name with its full path, or URL.
For a local WSDL file, enter the file name with its full path.
For a remote WSDL file, enter its URL.

Which WSDL file do you want to access?

WSDL File Name: http://localhost/webservice/n_webservice.asmx?WSDL

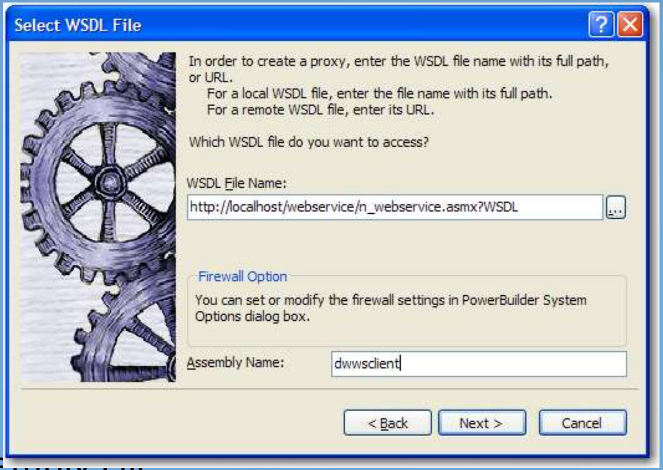
Firewall Option
You can set or modify the firewall settings in PowerBuilder System Options dialog box.

Assembly Name: dwwsclient

- Enter the URL to a WSDL, ASMX, or XML file, or browse a mapped drive for a WSDL file
  – The file selected should be in a publicly accessible location for all members of the development team

Provide a .Net Assembly Name …

- The Assembly File serves as an interface between the DataWindow and the Web Service



Select WSDL File

In order to create a proxy, enter the WSDL file name with its full path, or URL.
For a local WSDL file, enter the file name with its full path.
For a remote WSDL file, enter its URL.

Which WSDL file do you want to access?

WSDL File Name: http://localhost/webservice/n_webservice.asmx?WSDL

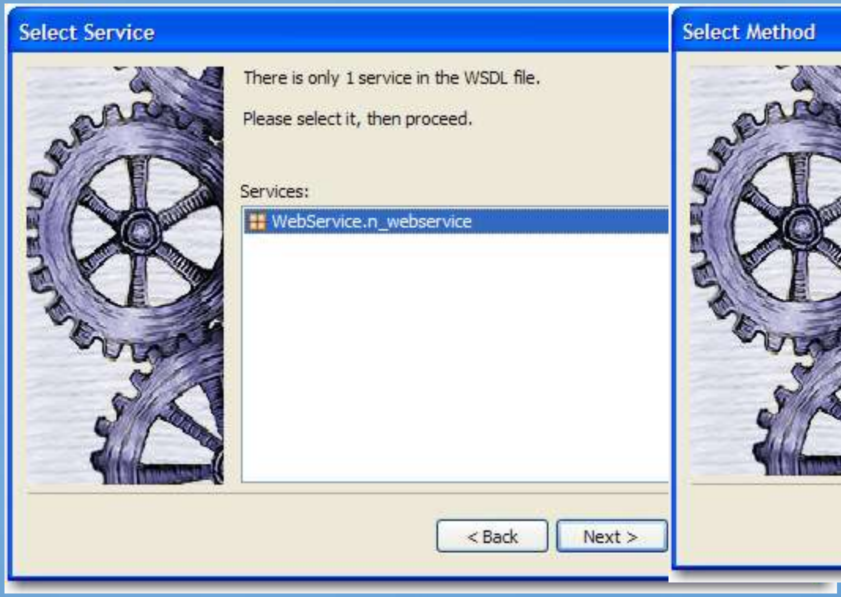
Firewall Option
You can set or modify the firewall settings in PowerBuilder System Options dialog box.

Assembly Name: dwwsclient

- Name the Assembly File
  - If you do not name the Assembly file, the wizard will select a name based on the name of the WSDL file entry

Select Web Service / Web Service Method

- Next, you must select a service described in the WSDL and then one of its public methods

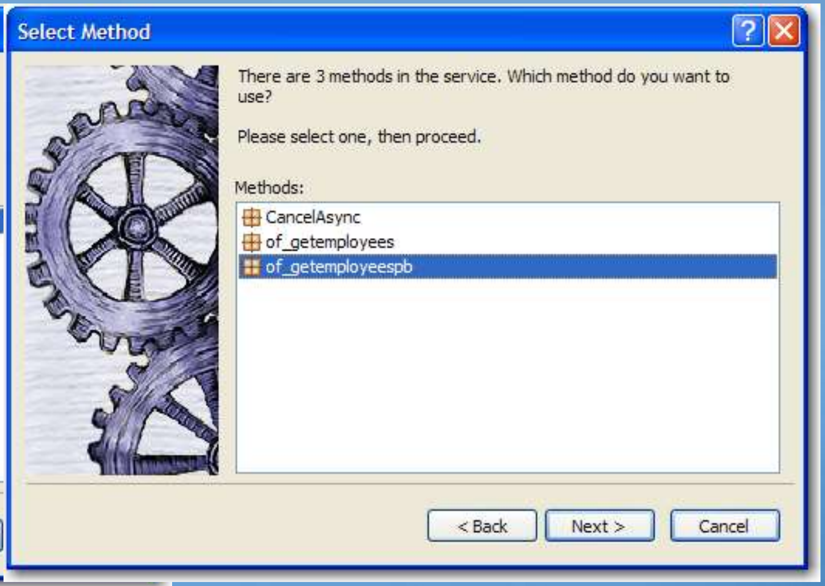


There is only 1 service in the WSDL file.

Please select it, then proceed.

Services:

WebService.n_webservice



There are 3 methods in the service. Which method do you want to use?

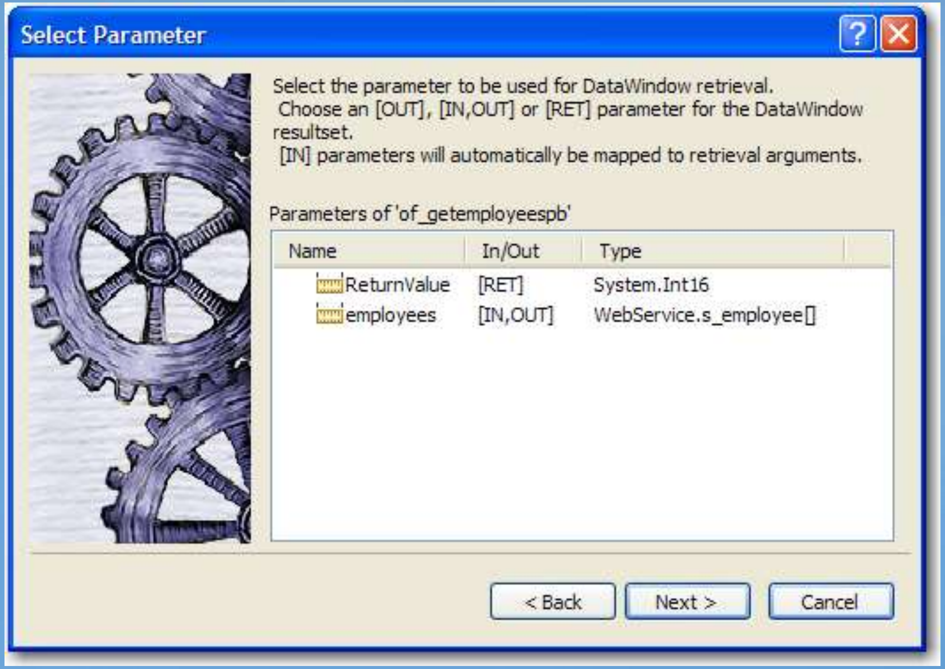
Please select one, then proceed.

Methods:

CancelAsync
of_getemployees
of_getemployeespb

Select the Web Service Method Output

- Select which of the methods arguments or its return value


Select Parameter

Select the parameter to be used for DataWindow retrieval.
Choose an [OUT], [IN,OUT] or [RET] parameter for the DataWindow resultset.
[IN] parameters will automatically be mapped to retrieval arguments.

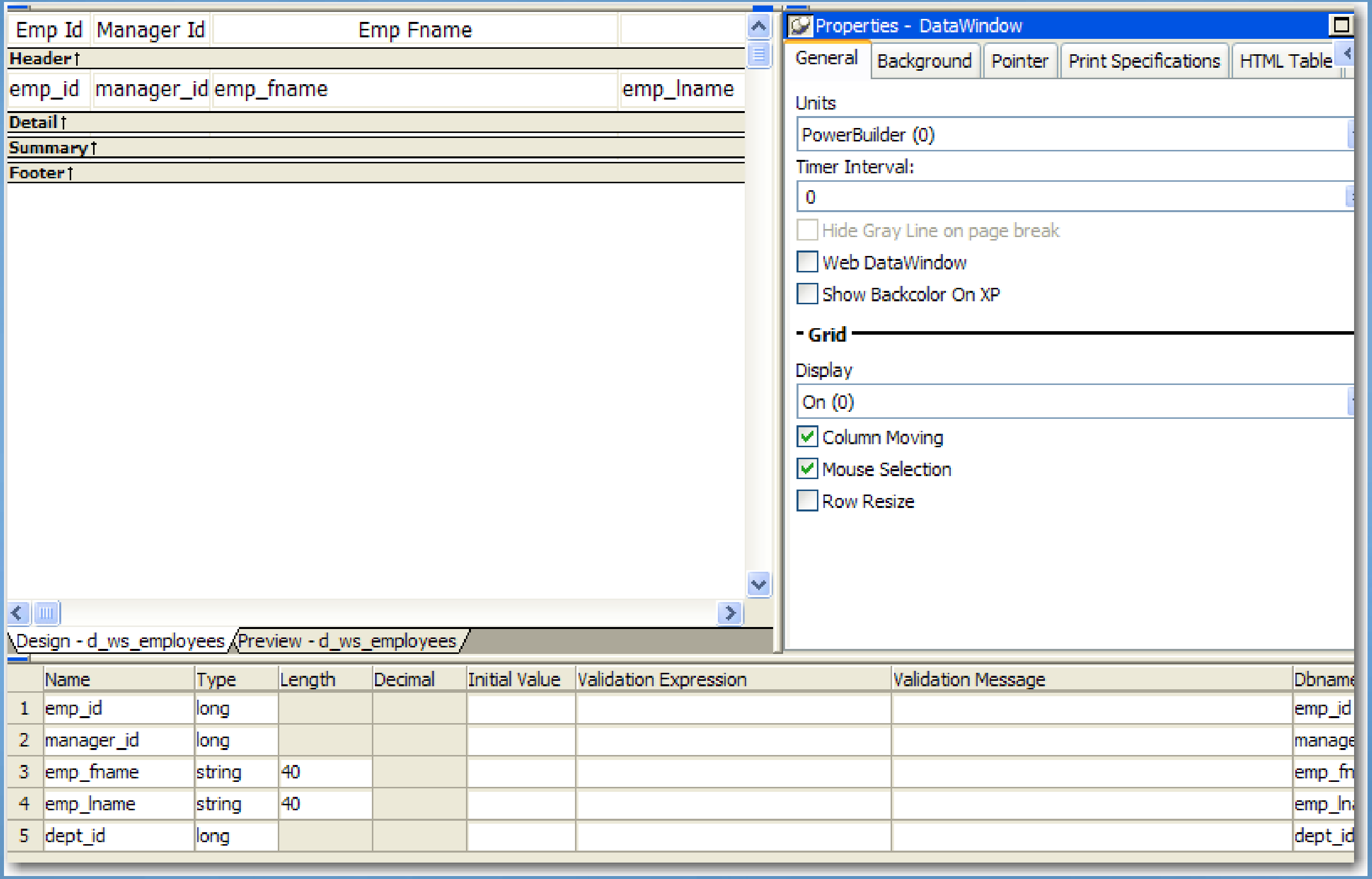
Parameters of 'of_getemployeespb'

Name	In/Out	Type
ReturnValue	[RET]	System.Int16
employees	[IN,OUT]	WebService.s_employee[]

Continued …

Finished Web Service DataWindow

- After completing the wizard the DataWindow is displayed



Emp Id	Manager Id	Emp Fname	
Header↑			
emp_id	manager_id	emp_fname	emp_lname
Detail↑			
Summary↑			
Footer↑			

Design - d_ws_employees / Preview - d_ws_employees

Properties - DataWindow

General | Background | Pointer | Print Specifications | HTML Table

Units: PowerBuilder (0)

Timer Interval: 0

☐ Hide Gray Line on page break
☐ Web DataWindow
☐ Show Backcolor On XP

Grid

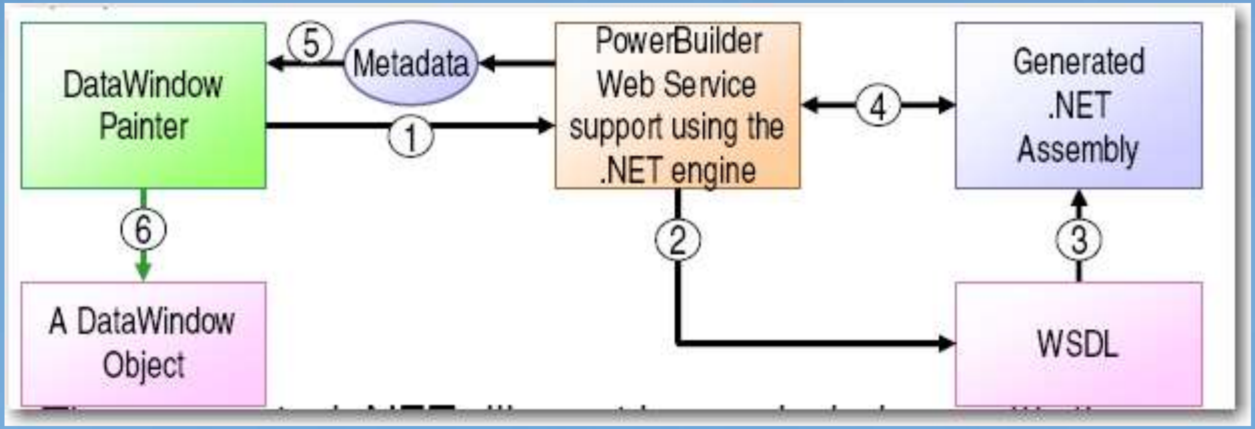
Display: On (0)

☑ Column Moving
☑ Mouse Selection
☐ Row Resize

	Name	Type	Length	Decimal	Initial Value	Validation Expression	Validation Message	Dbname
1	emp_id	long						emp_id
2	manager_id	long						manage
3	emp_fname	string	40					emp_fn
4	emp_lname	string	40					emp_ln
5	dept_id	long						dept_id

Interaction with the Web Service

- PowerBuilder automatically generates a .NET assembly (dll) used to interact with the Web Service at runtime

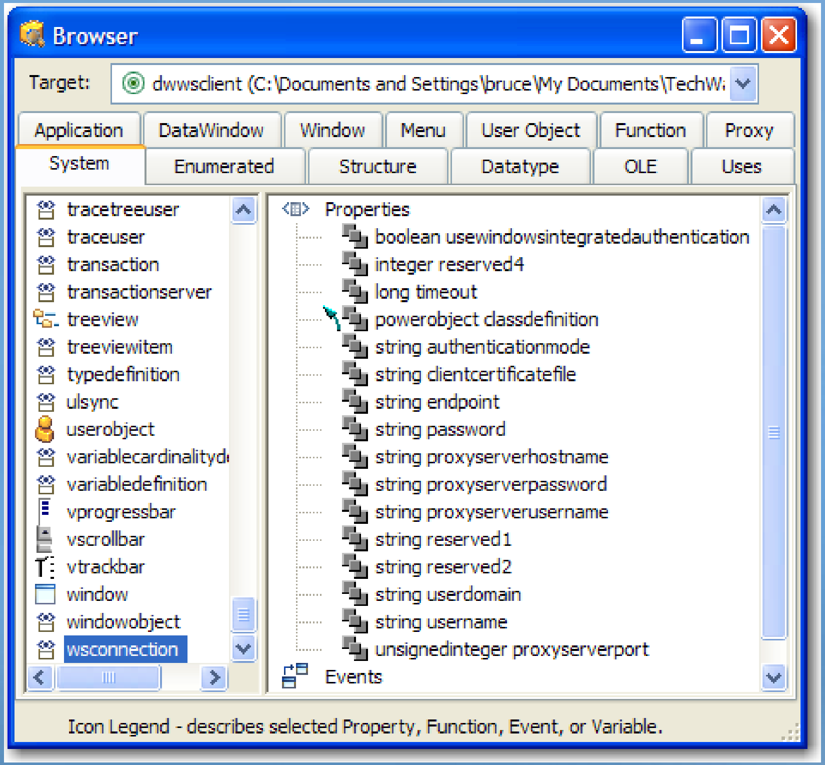


- The generated .NET dll must be copied along with the application executable and required PowerBuilder runtime DLLs for Web Service applications

New WS Connection Object

- Some Web services support or require a user ID and password, and other session-related properties
- The **wsconnection** can provide

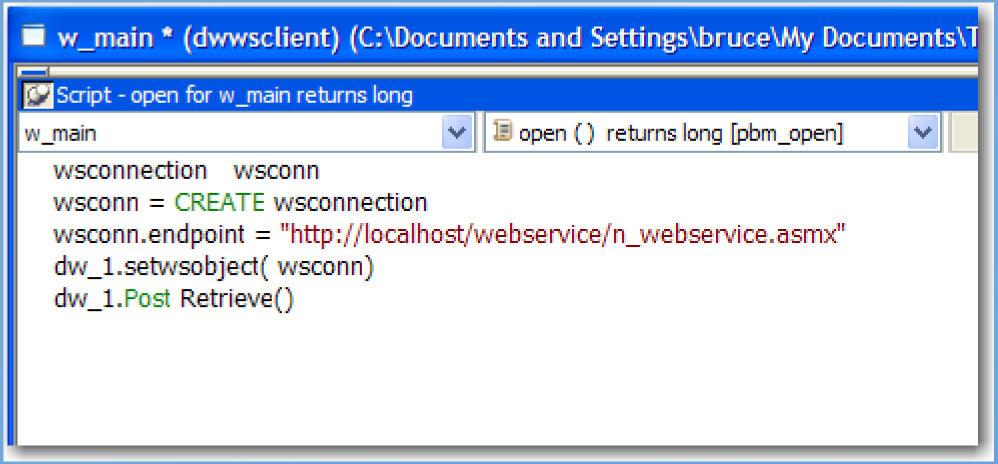
  this information:



```
tracetreeuser            Properties
traceuser                  boolean usewindowsintegratedauthentication
transaction                integer reserved4
transactionserver          long timeout
treeview                   powerobject classdefinition
treeviewitem               string authenticationmode
typedefinition             string clientcertificatefile
ulsync                     string endpoint
userobject                 string password
variablecardinalityd       string proxyserverhostname
variabledefinition         string proxyserverpassword
vprogressbar               string proxyserverusername
vscrollbar                 string reserved1
vtrackbar                  string reserved2
window                     string userdomain
windowobject               string username
wsconnection               unsignedinteger proxyserverport
                         Events
```

Icon Legend - describes selected Property, Function, Event, or Variable.

Sample WSConnection Code


w_main * (dwwsclient) (C:\Documents and Settings\bruce\My Documents\T
Script - open for w_main returns long
w_main
open ( ) returns long [pbm_open]

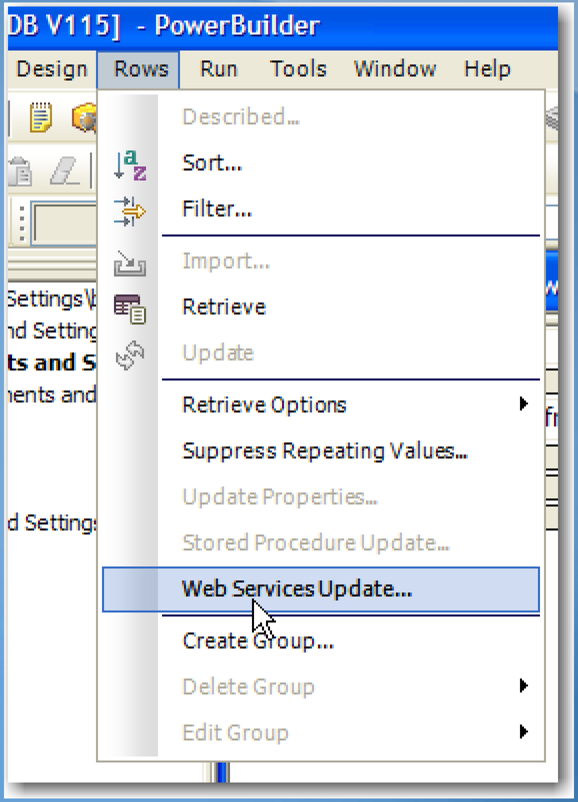
```
wsconnection   wsconn
wsconn = CREATE wsconnection
wsconn.endpoint = "http://localhost/webservice/n_webservice.asmx"
dw_1.setwsobject( wsconn)
dw_1.Post Retrieve()
```

Updates on WS DataWindows

- There are **no** transaction standards provided with Web Services
- Web Services are inherently stateless
  - Call a method, get a response, finished
- Given the above limitations, if updating data via a Web Service DataWindow, you will use the "Trust" methodology
  - Basically, you are throwing the data "over the fence" to the Web Service and trusting he will do the right thing
  - For example, if you have a DataWindow doing an insert, update and delete, and the call to the Web Service method for the delete fails, the Web Service DataWindow doesn't retain knowledge of the other two operations

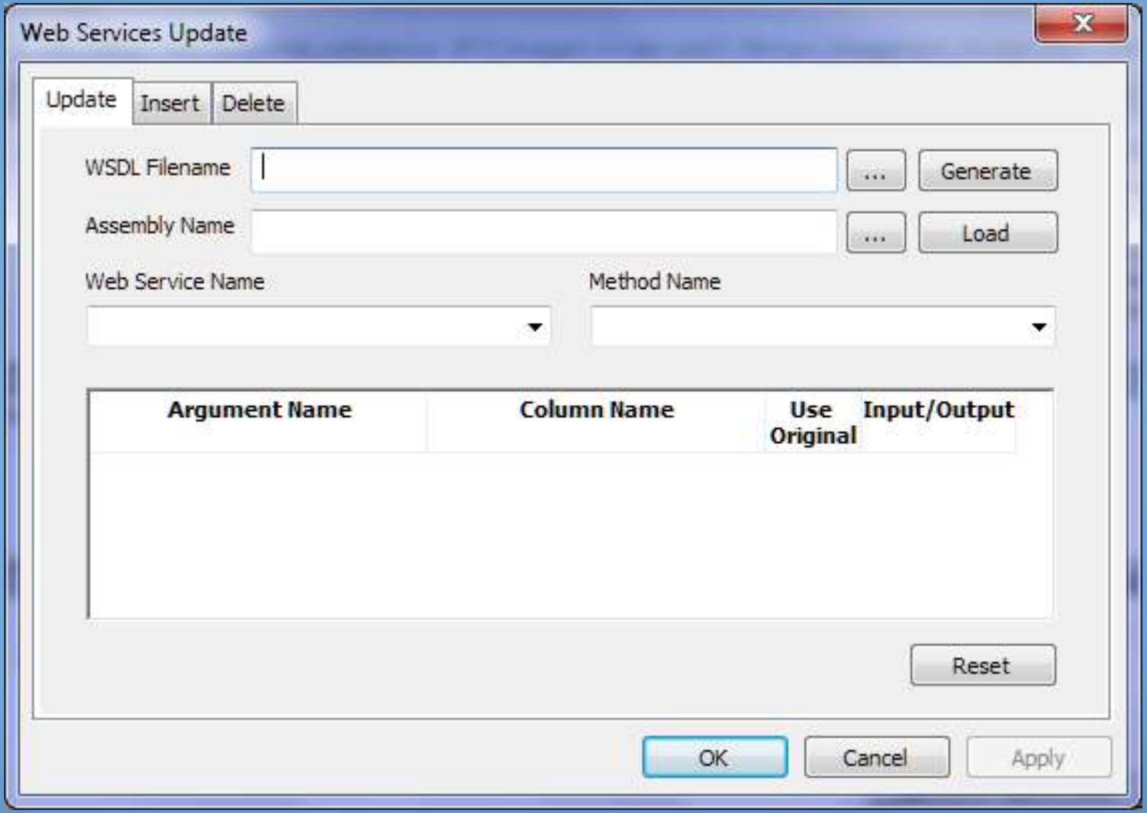
Defining Update Properties

- As mentioned before, the Web Service DataWindow was modeled from the Stored Procedure DataWindow
- The DataWindows Rows menu item now has a new item for Web Services Updates…
- Instead of mapping the DataWindow to a particular Stored Procedure, you will map the DataWindow (columns) to a particular Web Service method input parameter(s)



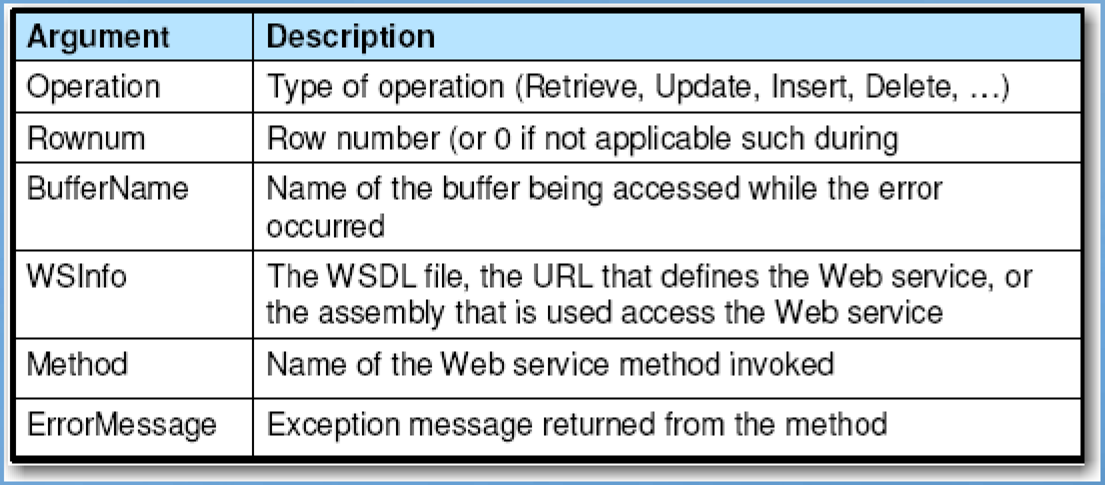
Web Service DataWindow Updates

- Similar to Stored Procedure update options



Web Service Error Handling

- New **WSError** event is analogous to the existing DataWindow DbError event when using a Web Service data source



Argument	Description
Operation	Type of operation (Retrieve, Update, Insert, Delete, …)
Rownum	Row number (or 0 if not applicable such during
BufferName	Name of the buffer being accessed while the error occurred
WSInfo	The WSDL file, the URL that defines the Web service, or the assembly that is used access the Web service
Method	Name of the Web service method invoked
ErrorMessage	Exception message returned from the method

Web Services Tracing

- You can also perform *limited* tracing of the Web Service DataWindow
- Do so by adding a key-value pair to PB.INI

```
[DataWindow] section
debug_ws_metadata = 1
```

Q&A Session

Questions?

Have you hugged your DataWindow today?