

# Team Developer™

.NET Projects

Product Version 7.0

Team Developer™: .NET Projects

### **Open Text Corporation**

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111 Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440

Fax: +1-519-888-0677

Support: <http://support.opentext.com>

For more information, visit <https://www.opentext.com>

Copyright © 2016 Open Text. All rights reserved. OpenText is a trademark or registered trademark of Open Text. The list of trademarks is not exhaustive of other trademarks, registered trademarks, product names, company names, brands and service names mentioned herein are property of Open Text or other respective owners.

### **Disclaimer**

No Warranties and Limitation of Liability. Every effort has been made to ensure the accuracy of the features and techniques presented in this publication. However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Warning: This software is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

## Table of Contents

<b>Chapter 1 – Introduction to .NET Projects.....</b>	<b>5</b>
.NET Build Settings.....	5
.NET Target Types.....	5
Other Settings .....	6
64-bit Applications.....	6
Signing Assembly .....	6
Compiling Process Overview.....	7
What Team Developer Does.....	7
Compiling from the Command Line .....	8
Data Type Matching.....	9
.NET Source Code.....	9
GUI Functions in DLLs .....	10
<b>Chapter 2 – Libraries and Functions.....</b>	<b>11</b>
QuickObjects.....	11
QuickTab2Tab Tool.....	11
Visual Toolchest.....	13
Stand-Alone Functions .....	13
VT Libraries.....	14
External Libraries and APIs in General .....	15
Datatypes in External DLLs .....	15
user32.dll Not Supported in .NET.....	15
Assembly Information for APLs.....	15
Unsupported SAL Functions .....	16
Unsupported APLs.....	18
Supported WM_* Messages .....	18
<b>Chapter 3 – WPF Applications.....</b>	<b>20</b>
WPF Applications.....	20
XAML Files .....	21
Customizing XAML files.....	22
Deploying WPF Desktop Applications.....	23
WPF Browser Applications in Firefox.....	23
WPF Browser Applications in Internet Explorer .....	24
<b>Chapter 4 – Publishing XBAP Applications .....</b>	<b>25</b>
Publishing XBAP Applications (WPF Browser).....	25

Build Settings for WPF Browser Application.....	25
Publishing a WPF Browser Application to IIS.....	27
IIS Troubleshooting .....	28
Database Proxy Service.....	29
<b>Chapter 5 – WPF Controls.....</b>	<b>31</b>
Custom WPF Control in Team Developer .....	31
WPF Functions.....	31
WPF Events.....	31
Debugging Applications with WPF Controls.....	32
Gauges.....	32
Examples.....	33
<b>Chapter 6 – Connectivity in .NET Applications.....</b>	<b>34</b>
Connecting to Oracle .....	34
Connecting to SQLServer .....	35
SQL Server TIMESTAMP.....	37
SQLBase.....	38
<b>Chapter 7 – .NET Explorer.....</b>	<b>39</b>
Using .NET Explorer.....	39
About AXLs, APLs, DLLs.....	41
Definitions.....	41
.NET Explorer Generates APL and AXL .....	41
.NET SAL Library Generates DLL and AXL .....	42
Variables Based on Imported .NET Classes .....	42
.NET Assemblies Created in Visual Studio .....	42
<b>Chapter 8 – Debugging DLLs.....</b>	<b>43</b>
.NET Class Libraries .....	43
.NET Web Services.....	46
.NET SAL Libraries.....	48

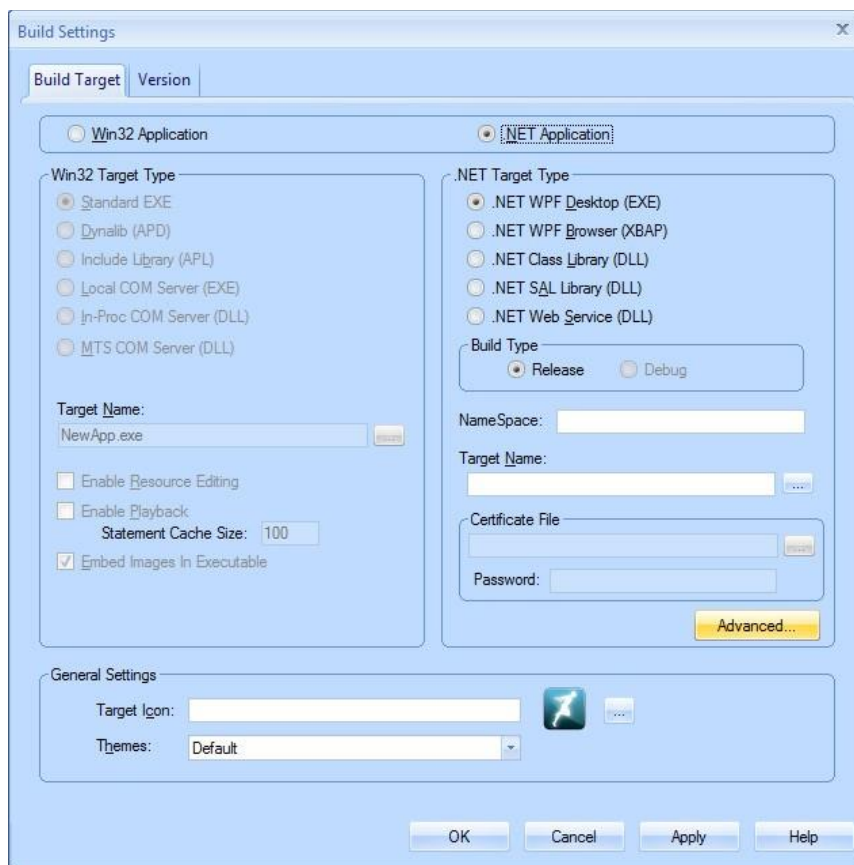
# Chapter 1 – Introduction to .NET Projects

This chapter provides an overview of the .NET portion of the build settings dialog box. It also contains a conceptual explanation of the compiling process that enables you to create a .NET application with Team Developer.

---

## .NET Build Settings

The Build Settings dialog box is accessed by selecting “Build Settings” from the Project menu (see the “Project Menu” section of the *Developing With SQLWindows* document). The right side of this dialog box contains options for .NET build settings. These options and settings are explained in the sections that follow.



## .NET Target Types

- **.NET WPF Desktop (EXE)**—A WPF application that runs from the desktop.
- **.NET WPF Browser (XBAP)**—A WPF application designed to run in a browser.
- **.NET Class Library (DLL)**—A class library used in developing other applications.
- **.NET SAL Library (DLL)**—The WPF equivalent for the dynalib compiler option in Win32 mode.

- **.NET Web Service (DLL)**—A .NET Web service that can be hosted on IIS.

You can also compile from the command line, including parameters to specify the build target. For examples, see “Compiling Process Overview” on page 7.

## Other Settings

**Target Name**—Used to indicate the name of the application, class library, or Web service you are going to build.

**Target Directory**—The directory where your build will be saved.

## 64-bit Applications

The Advanced button on the build settings dialog brings up a new Advanced Settings dialog box. In this dialog, the user can choose a value for the Target CPU:

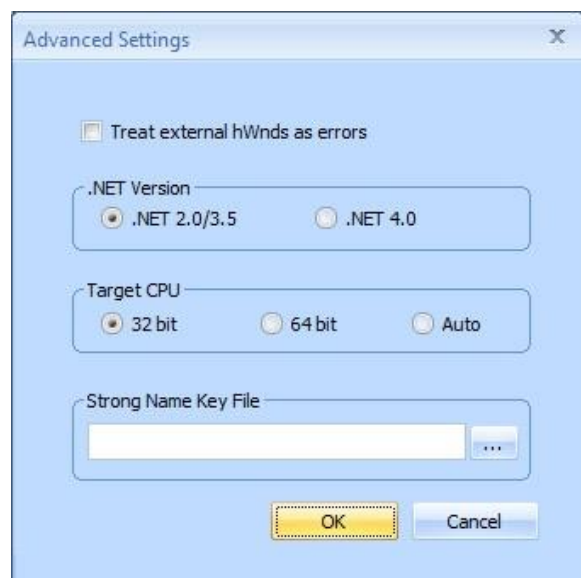
- 32bit: The application will always be 32-bit. On a 64-bit OS, applications run within the 32-bit subsystem.
- 64bit: The application will always be 64-bit and can only run on a 64-bit machine.
- Auto: The application will decide what bitness to use at launch time, depending on the OS. For DLLs, the bitness depends on the process loading them.

There is a new command-line argument to TD for specifying the target CPU when compiling from a DOS prompt. The syntax for the new flag is "-cpu:32bit", "-cpu:64bit", or "-cpu:auto", as follows:

```
cbi62.exe -net:WPFDesktop -cpu:64bit test.apt test.exe
cbi62.exe -net:WPFDesktop -cpu:32bit test.apt test.exe
cbi62.exe -net:WPFDesktop -cpu:auto test.apt test.exe
```

## Signing Assembly

The Advanced button on the build settings dialog brings up a new Advanced Settings dialog box. In this dialog, the user can specify the Strong Name Key file (\*.snk). The .NET compiler signs the assembly using the .snk file.



---

## Compiling Process Overview

.NET applications are compiled from Intermediate Language (IL). To create a .NET application from a Team Developer outline, your SAL code must be converted to IL and compiled into a .NET product. With Team Developer 6.0, Gupta introduced a new IL compiler that performs these functions for you.

To create a .NET product with Team Developer, execute the following steps:

1. Write your application in Team Developer as usual, using SAL code.
2. In the Project menu, select **Build Settings** to access the Advanced Settings dialog box.
3. Choose a .NET target type, and specify a target name and directory.

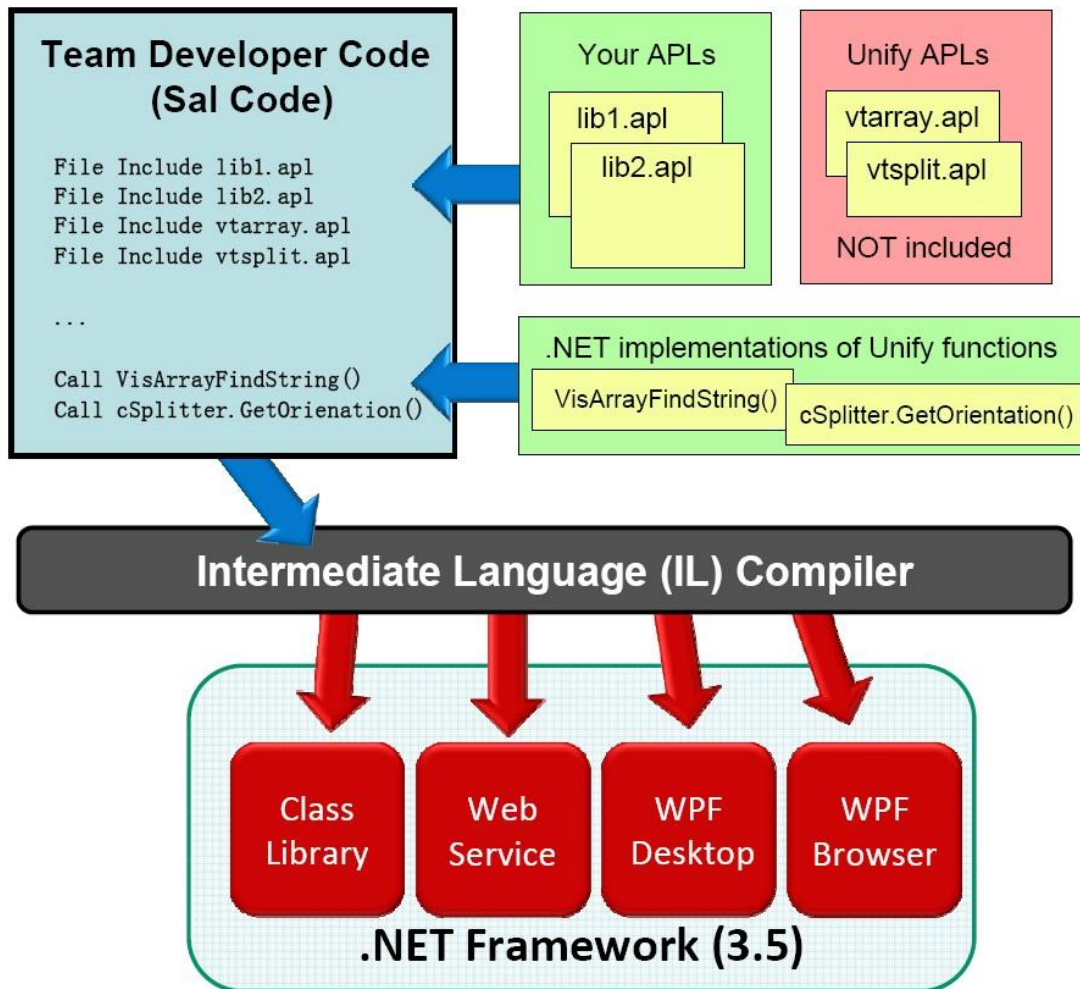
You can also compile from the command line. See “Compiling from the Command Line” on page 8 for information and examples.

### What Team Developer Does

Once you select a .NET target type, build your project as usual. Team Developer turns your Team Developer outline into a .NET application, class library, or Web service.

- **Library inclusions:**  
Any libraries you created and included are made available to the compiler.
- **Functions from APLs:**  
Gupta-provided libraries are not included. Instead, .NET implementations of these libraries’ functions are created individually. Team Developer and the IL compiler ignore your include statements for these libraries and incorporates the .NET implementations where functions from these libraries are called.
- **IL compile:**  
The IL compiler receives your SAL code, included APLs, and the .NET implementations of Gupta functions you called in your application. These elements are converted to IL and compiled.
- **Finished product:**  
Your .NET project is built and saved with the directory and filename you specified in the Build Settings dialog box. Team Developer uses the standard output window to report any errors.

The following diagram illustrates this process:



## Compiling from the Command Line

The following are example commands for compiling from the command line.

- To compile using the standard TD native compiler (Win32 application):

```
cbi62.exe -b test.app test.exe
```

- To compile a WPF desktop application:

```
cbi62.exe -.net:WPFDesktop test.app test.exe
```

or

```
cbi62.exe -.net:AutoDesktop test.app
```

- To compile a WPF browser application:

```
cbi62.exe -.net:WPFBrowser test.app test.exe
```

or

```
cbi62.exe -.net:AutoBrowser test.app
```

- To compile a .NET SAL library:

```
cbi62.exe -.net:SALLibrary test.apl test.dll
```

- To compile a .NET Class library:

```
cbi62.exe -.net:ClassLibrary test.apl test.dll
```



For an application to compile correctly from the command line, the command must match the application's build target (from the Build Settings dialog box). The following table shows the results of each .NET command line switch when used on an application with the indicated build target.

.NET Command Line Switch	Application's Current Build Setting	Result
.net:WPFDesktop	win 32 EXE, .NET WPF Desktop, .NET WPF Browser	.NET WPF Desktop
	All others	.err (error file)
.net:WPFBrowser	win 32 EXE, .NET WPF Desktop, .NET WPF Browser	.NET WPF Browser
	All others	.err (error file)
.net:ClassLibrary	Include Library, .NET Class Library	.NET Class Library
	All others	.err (error file)
.net:SALLibrary	Dynalib, .NET SALLibrary	.NET SALLibrary
	All others	.err (error file)
.net:AutoDesktop	win 32 EXE, .NET WPF Desktop, .NET WPF Browser	.NET WPF Desktop
	Include Library, .NET Class Library	.NET Class Library
	Dynalib, .NET SALLibrary	.NET SALLibrary
	All others	.err (error file)
.net:AutoBrowser	win 32 EXE, .NET WPF Desktop, .NET WPF Browser	.NET WPF Browser
	Include Library, .NET Class Library	.NET Class Library
	Dynalib, .NET SALLibrary	.NET SALLibrary
	All others	.err (error file)

## Data Type Matching

Native Team Developer applications allow for some mixing of data types. For example, a number variable can contain a window handle. In .NET build targets, this type of mismatch causes a compiler error.

## .NET Source Code

The IL compiler does not generate C# or other Microsoft source code. Team Developer contains a true .NET compiler that compiles the SAL language directly to

.NET Intermediate Language. The only IDE that can edit and compile SAL code is Team Developer.

There is a size limitation to TD code when being compiled to .NET. This comes from the Microsoft .NET Assembler, whose memory usage can climb over 2GB if the application is large enough. In TD, it seems to be around 650,000 outline items. Using the 64-bit version of the assembler (which TD will do when possible) removes the size limitation.

Compiling to .NET4 seems to shrink the size of the IL, buying the user a little more time, perhaps allowing a 700,000-800,000 outline item application before the memory limit is reached.

## **GUI Functions in DLLs**

.NET class libraries, .NET SAL libraries and .NET Web services do not support GUI objects. Team Developer applications that use GUI functions report warnings when the application is built as a .NET class library or a .NET SAL library. Team Developer applications that use GUI functions report errors when built as a .NET Web service dll.

# Chapter 2 – Libraries and Functions

This chapter provides information about which libraries and functions are supported by Team Developer for .NET projects for the following:

- QuickObjects
- Visual Toolchest
- External libraries and APIs in general
- Unsupported SAL functions
- Unsupported APLs
- Supported WM\_\* messages

---

## QuickObjects

With the exception of QuickGraphs, QuickObjects are not supported for .NET projects. However, the QuickTab2Tab tool enables you to convert your QuickTabs to native Team Developer Tab Controls. The native Tab Control is supported for .NET projects.

## QuickTab2Tab Tool

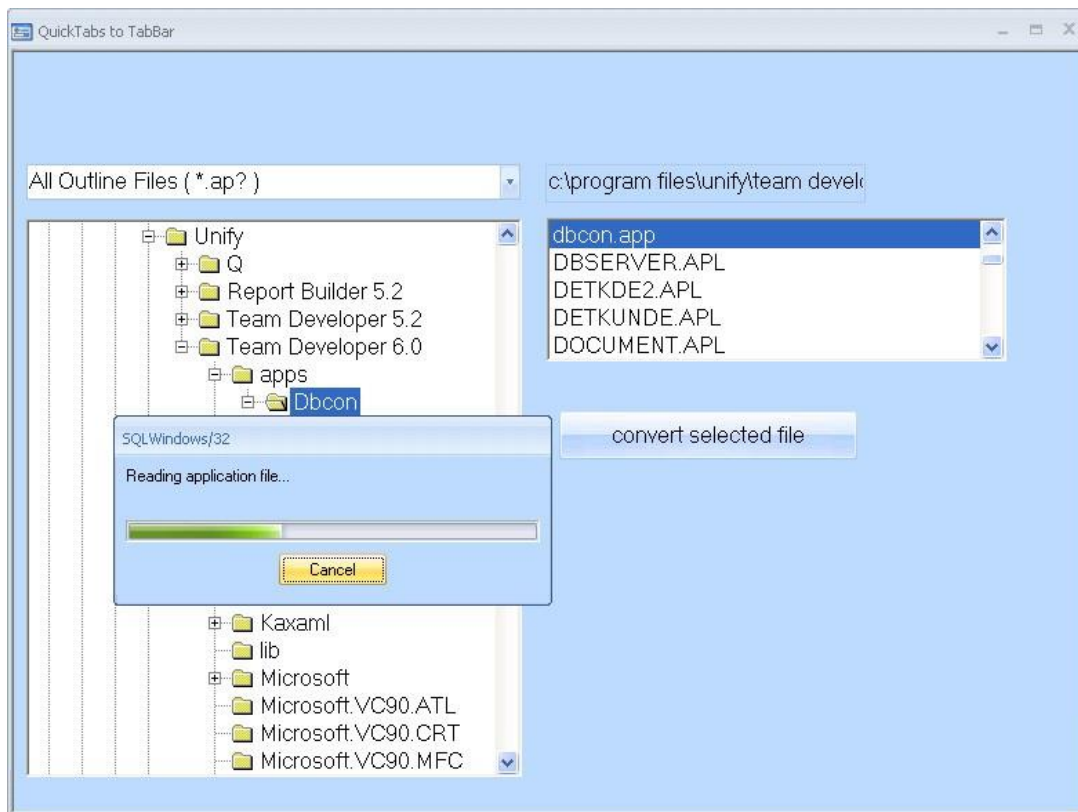
Use this tool to prepare an application for .NET deployment. Conversion from QuickTabs to native Tab Controls is not required for Win32 applications.

The QuickTab2Tab tool (qtab2tab.exe) is located in the root directory of your Team Developer installation. To convert an application's QuickTabs to native Team Developer tab controls, execute the following steps:

1. Make a copy of the target application and included libraries (qtab2tab.exe overwrites files; it does not create copies).
2. Find qtab2tab.exe in your Team Developer installation directory and double-click it to launch the QuickTab2Tab conversion tool.
3. Using the explorer tree at the left, find the directory that contains the application file you want to convert.
4. (*Optional*) Use the "All Outline Files" dropdown box to filter the displayed files by file type.
5. In the list at the right, select the file you want to convert and click **Convert Selected File**.

Application conversion cannot be done in pieces. To convert an application and its included libraries, convert the top-level .app file. QuickTab2Tab finds the necessary libraries and converts them as well.

The QuickTab2Tab tool reads and converts the required files. During the process, you will see progress bars like the one pictured below. A message box informs you when conversion is complete.



When conversion is complete, your application file and any included libraries that contained QuickTabs will have been overwritten. The updated files will contain native Team Developer tab bar controls, and any files that previously included qcktabs.apl and pagelist.apl will now include qtab2tab.apl.

QuickTab2Tab does not migrate classes derived from cSWTabs or cTabPageList. These must be done manually.

The following QuickTab functions are not supported for QuickTab2Tab conversion:

CancelMode	Delete	FindName
GetContentsBorderRect	GetContentsRect	GetContentsRectPixels
GetDrawStyle	GetMarginRect	GetMinimumWidth
GetName	GetRowCount	IndexFromPoint
InitFromProps	Insert	SetDrawStyle
SetMinimumResizeSize	SetName	SetRowCount
SetWorkspaceBoundary	ShowSiblings	

---

## Visual Toolchest

The Visual Toolchest consists of several class libraries and many stand-alone functions. Most of the stand-alone functions are implemented for .NET compatibility, but some are not. Visual Toolchest class libraries are mostly unsupported, with a couple of exceptions.

Even when a Visual Toolchest class library is supported, there is no mechanism for accessing variables in imported .NET assemblies. Thus, it is not possible to directly access member variables of a Visual Toolchest class when building to .NET.

### Stand-Alone Functions

The following stand-alone Visual Toolchest functions are supported for .NET projects:

#### **vtarray.apl**

VisArrayAppend	VisArrayCopy	VisArrayFillNumber	VisArrayFindString
VisArrayDeleteItem		VisArrayFillString	VisArrayInsertItem
VisArrayFillDateTime		VisArrayFindDateTime	VisArraySort
		VisArrayFindNumber	

#### **vtdebug.apl**

VisDebugBeginTime	VisDebugSetFlags	VisDebugSetTime
VisDebugEndTime	VisDebugSetLevel	VisDebugString
VisDebugGetFlags		

#### **vtDOS.apl**

VisDosBuildFullName	VisDosIsParent	VisDosGetVersion
VisDosEnumDirInfo	VisDosGetCurDir	VisDosGetVolumeLabel
VisDosEnumDirs	VisDosGetDriveSize	VisDosMakeAllDir
VisDosEnumDrives	VisDosGetDriveType	VisDosMakePath
VisDosEnumFileInfo	VisDosGetEnvString	VisDosSetFlags
VisDosEnumFiles	VisDosGetFlags	VisDosSplitPath
VisDosEnumPath	VisDosGetNetName	VisDosSetVolumeLabel
VisDosExist		

#### **vtfile.apl**

VisFileAppend	VisFileGetSize	VisFileSetDateTime
VisFileClose	VisFileGetType	VisFileSeek
VisFileCopy	VisFileOpen	VisFileTell
VisFileCreateTemp	VisFileRead	VisFileWrite
VisFileDelete	VisFileReadBinary	VisFileWriteBinary
VisFileExpand	VisFileReadString	VisFileWriteString
VisFileFind	VisFileRename	
VisFileGetAttribute	VisFileSetAttribute	

#### **vtmenu.apl**

VisMenuCheck	VisMenuGetPopupHandle	VisMenuInsertPicture
VisMenuDelete	VisMenuGetText	VisMenuSetFont
VisMenuDisable	VisMenuIsChecked	VisMenuSetPicture
VisMenuEnable	VisMenuIsEnabled	VisMenuSetText
VisMenuGetCount	VisMenuInsert	VisMenuUncheck
VisMenuGetHandle	VisMenuInsertFont	

**vtmisc.apl**

VisGetVersion	VisGetWinVersion	VisNumberChoose
VisGetWinFlags	VisSendMsgStrin	VisProfileEnumStrings

**vtstr.apl**

VisStrChoose	VisStrLoadTable	VisStrScanReverse
VisStrExpand	VisStrPad	VisStrSubstitute
VisStrFind	VisStrProper	VisStrTrim
VisStrFreeTable	VisStrRightTrim	VisStrPadB
VisStrLeftTrim		

**vttblwin.apl**

VisTblClearColumnSelection	VisTblFindDateTime
----------------------------	--------------------

**vtwin.apl**

VisWinClearAllFields	VisWinIsChild	VisWinLoadAccelerator
VisWinClearAllEditFlags	VisWinIsMaximized	VisWinMove
VisWinFreeAccelerator	VisWinIsMinimized	VisWinSetFlags
VisWinGetFlags	VisWinIsRequiredFieldNull	VisWinSetStyle
VisWinGetHandle	VisWinIsRestored	VisWinSetTabOrder
VisWinGetStyle	VisWinIsWindow	VisWinShow
VisWinGetText		

**VT Libraries**

The following Visual Toolchest class libraries and associated base classes are supported for .NET projects. Other Visual Toolchest class libraries are not supported.

**vtcomm.apl**

Base classes: sPoint, sRect, sSize.

**vtsplit.apl**

Base classes: cSplitter, cSplitterWindow.

**vtmeter.apl**

Base classes: cMeter

**vtcal.apl**

Base classes: cCalendar, cCalendarDropDown

Currently no methods are supported for vtcal.apl.

**vtlbox.apl**

Baseclasses: cOutlineListBox, cPictureListBox, cRadioListBox, cColorListBox

---

## External Libraries and APIs in General

### Datatypes in External DLLs

When using an external DLL in a .NET project and calling its functions, remember that native TD/SALC datatypes are not supported as return values, parameters or receive parameters. These datatypes include:

- DATETIME
- HFILE
- HUDV
- HWND \*see note below
- HSESSIONHANDLE
- HSQLHANDLE
- LPDATETIME
- LPHFILE
- LPNUMBER
- LPSESSIONHANDLE
- LPHSQLHANDLE

Only primitive C/C++ types are supported, with the exception of HARRAY, HUDV, and STRUCTPOINTER, which are not supported.

WPF applications use object references rather than window handles. Modernizing your application to .NET and moving to WPF means moving away from manipulating windows via API calls. Fortunately, the functionality you acquired via external APIs in the past is probably available in native SAL code now.

### user32.dll Not Supported in .NET

This is a Win32 DLL and cannot be made to work in .NET.

### Assembly Information for APLs

Team Developer contains .NET assembly information for the following APLs. This is not a list of fully-supported APLs in .NET. Rather, these are APLs that will not cause compiler errors when included in a .NET project. Because assembly information is available, the project will compile. You will also see the symbol information in the External Assemblies section of the outline. In most cases, however, the assembly information will not include every function in the APL.

You can check if an APL has assembly information by doing the following:

1. Open the APL.
2. Right-click its name at the top of the tree view and select **Properties**.
3. Click the **AssemblyFile** tab. If text appears in the field under “Assembly Import File”, Team Developer has assembly information for this APL.

Automation.apl

VTCComm.apl

cgmail.apl	VTDebug.apl
cdk.apl	VTDOs.apl
CStructL.apl	VTFile.apl
GTableX.apl	VTLbx.apl
ODBSal32.apl	VTlstvw.apl
QckDVC.apl	VTMenu.apl
QckMail.apl	vtmeter.apl
QckTabs.apl	VTMisc.apl
SalMail.apl	vtsplit.apl
sqlnwkc.n.apl	VTStr.apl
VArray.apl	VTTblWin.apl
VTComDlg.apl	xmlib.apl

---

## Unsupported SAL Functions

The following SAL functions are not supported for .NET build targets:

SalActiveXAutoErrorMode	SalActiveXCreateFromData
SalActiveXCreateFromFile	SalActiveXDelete
SalActiveXGetData	SalActiveXGetFileName
SalActiveXInsertObjectDlg	SalActiveXOLEType
SalAppFind	SalContextBreak
SalCreateWindowExFromStr	SalCreateWindowFromStr
SalDDEAddAtom	SalDDEAlloc
SalDDEDeleteAtom	SalDDEExtract
SalDDEExtractCmd	SalDDEExtractDataText
SalDDEExtractOptions	SalDDEFindAtom
SalDDEFree	SalDDEGetAtomName
SalDDEGetExecuteString	SalDDEPost
SalDDERequest	SalDDESend
SalDDESendAll	SalDDESendExecute
SalDDESendToClient	SalDDESetCmd
SalDDESetDataText	SalDDESetOptions
SalDDEStartServer	SalDDEStartSession
SalDDEStopServer	SalDDEStopSession
SalDropFilesQueryPoint	SalEditCanPasteLink
SalEditCanPasteSpecial	SalFontGetSizes
SalGetTypeEx	SalHBinaryToNumber
SalHtmlHelp	SalIdleKick



SalIdleRegisterWindow	SalIdleUnregisterWindow
SalModalDialogFromStr	SalMTSCreateInstance
SalMTSDisableCommit	SalMTSEnableCommit
SalMTSGetObjectContext	SalMTSIsCallerInRole
SalMTSIsInTransaction	SalMTSIsSecurityEnabled
SalMTSSetAbort	SalMTSSetComplete
SalNumberToHBinary	SalObjIsNull
SalOutlineChildOfType	SalOutlineCurrent
SalOutlineItemOfWindow	SalOutlineItemTypeText
SalReportClose	SalReportCreate
SalReportGetRichTextVar	SalReportSetRichTextVar
SalRibbonSetItemTransparentColor	SalStaticFirst
SalStaticGetSize	SalStaticSetSize
SalStrToMultiByte	SalStrToWideChar
SalTabSetPageTransparentColor	SalTblSetColumnXMLAttributes
SalTblSetView	SalTblQueryView
SalWindowGetDockSetting	SalWinGetStyle
SalXMLDeserializeUDV	SalXMLGetLastError
SalXMLSerializeUDV	SalYieldEnable
SalYieldQueryState	SalYieldStopMessages
SalYieldStartMessages	SqlClose
SqlCloseAllSPResultSets	SqlCloseResultSet
SqlConnectTransaction	SqlContextSetToForm
SqlDeleteConnectionString	SqlDropStoredCmd
SqlFindIniFile	SqlGetConnectionStrings
SqlGetCmdOrRowsetPtr	SqlGetCursor
SqlGetErrorPosition	SqlGetDSOrSessionPtr
SqlGetLastStatement	SqlGetRollbackFlag
SqlGetStatementErrorInfo	SqlImmediateContext
SqlListConnections	SqlOpen
SqlOraPLSQLStringBindType	SqlRetrieve
SqlStore	SqlWriteConnectionString

---

## Unsupported APLs

The following APLs are not supported in Team Developer 6.2.

axtmpl.apl	pagelist.apl
cdk.apl	swbidi32.apl
cdkfwrk.apl	TINotify.apl
cgmail.apl	ttmngr.apl
gtablex.apl multitbl.apl	

---

## Supported WM\_\* Messages

In WPF applications, the following WM\_\* messages are supported for the window types listed.

- WM\_LBUTTONDOWN  
Button Check Box Option Data Field  
Rich Text Box List Box Picture Box  
Child Grid/Child Table/Table Window/Grid Window
- WM\_RBUTTONDOWN  
Form/Dialog  
All controls except:  
Data Fields  
Multi-line text Tab bar Navigation Bar
- WM\_LBUTTONUP  
Button Check Box Data Field  
Rich Text Box Combo Box
- WM\_RBUTTONUP  
Child Grid Child Table Grid Window Table Window
- WM\_CHAR  
All field controls (Data Field, Combo Box, Multiline, Rich Text, Column) Child Grid  
Child Table Table Window Grid Window
- WM\_KEYUP  
All controls except:  
Pushbutton  
Picture Tab bar  
Navigation bar  
Top Level Grid Window Top Level Table Window

- WM\_KEYDOWN  
Data Field Rich Text Box Child Grid  
Child Table List Box Combo Box  
Column
- WM\_LBUTTONDOWNBLCLK  
Data Field Rich Text Box Combo Box
- WM\_RBUTTONDOWNBLCLK  
Not supported in WPF.
- WM\_MOUSEMOVE  
Not supported in WPF.
- WM\_KILLFOCUS  
Data Field Rich Text Box
- WM\_SIZE  
Form/Dialog MDI

# Chapter 3 – WPF Applications

This chapter discusses WPF applications and their associated files, describes WPF desktop and browser applications, and explains XAML files and how to customize them.

---

## WPF Applications

The Build Settings dialog box provides two WPF options: WPF Desktop and WPF Browser. These two options are virtually the same. The resulting applications are identical except that one can be deployed on the desktop and the other is hosted in a browser.

Best practices for WPF applications include the following:

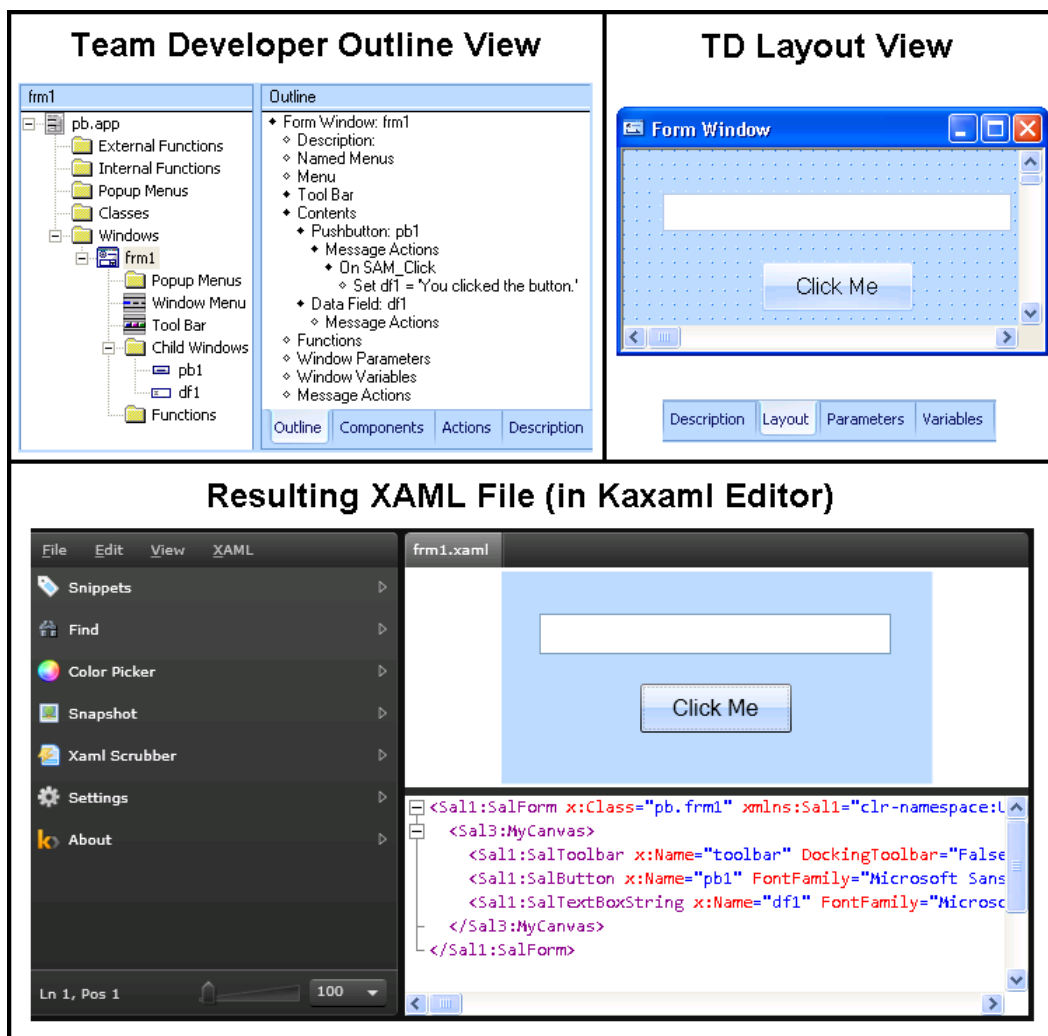
- Names of WPF Browser applications cannot contain some punctuation characters, such as [ (open bracket) and ] (close bracket).
- Certain keywords in your application file name will cause a “User Account Control” warning in Windows when you try to run the application. Avoid words like “install,” “setup,” and “update” in your file name.
- WPF applications cannot have two top-level items with the same name. This includes an MDI window and its child form window, which are both considered top-level when compiling to .NET.
- WPF applications cannot include a library with the same name as the application. For example, you cannot call a library called `foo.dll` from an application called `foo.exe`.
- WPF does not have a notion of a cache, so `SAM_CacheFull` will never fire in a WPF application.
- Because WPF Browser applications run inside a browser window, `SAM_Close` and `WM_Close` messages do not succeed. Instead, use `SalDestroyWindow`.
- In WPF applications, `SalReportView` launches an independent instance of Report Builder (rather than a runtime preview window).
- If the .NET 4.0 framework is installed, you will not be able to debug WPF browser applications. To avoid this problem, use the .NET 3.5 framework or debug your applications using the WPF desktop build target. (Microsoft will probably release a fix for this bug soon.)
- When debugging WPF applications, Step Into and Step Over work differently than in Win32. You will need to set a distinct break point for each message or event in WPF, or else an attempt to “Step Into” might result in a “Step Over.”
- To run TD60 WPF applications using ADOProxy running on 64-bit machine, you must enable 32-bit Applications mode on IIS. To do this, follow these steps:
  - a. Click **View Application Pools**.
  - b. Set Application pool defaults.
  - c. Set Enable 32-bit Applications to **TRUE**.

## XAML Files

XAML is an XML-based markup language developed by Microsoft. It is the language behind the visual presentation of WPF applications. XAML is similar to HTML in that it is text based and tag based, and it determines the look and structure of a project. In the case of HTML, the project is a webpage. In the case of XAML, the project is an application. In TD, XAML files determine the visual aspects of a TD application deployed as WPF.

When you compile a TD application with a WPF build target, XAML data is automatically created, although it is not exposed as independent files. If you decide to customize the XAML, files are created in your project directory for you to view and edit (see Customizing XAML files, which follows). These XAML files contain code that specifies the layout, structure, size, colors, etc. of the various windows, buttons, and other visual elements in your application. A .xaml file is created for each top-level dialog box, form window, MDI window, grid window, and table window.

The following screenshots show a simple form window as it appears in the Team Developer outline and layout, and the resulting XAML file as viewed in the editor provided with Team Developer.



## Customizing XAML files

The previous screenshot of the XAML file shows the file being viewed in the Kaxaml-based editor that ships with Team Developer. You can access Kaxaml through Team Developer by right-clicking on a parent window of any type in the outline or layout view. In the right-click menu that appears, select Custom XAML and then Edit custom XAML...

When you select “Edit custom XAML,” a new .xaml folder is created in your project directory, and it is populated with .xaml files for your project. Kaxaml launches, and you can use it to view and edit the .xaml files (select “Open” from the file menu, find the new .xaml directory, and select the file you want to view/edit).

Best practices for customizing XAML files include the following:

- When customizing XAML files for Team Developer applications, use the XAML editor provided with Team Developer.
- Use the method mentioned above to access custom XAML files (right-click/ Custom XAML/Edit custom XAML). Running Kaxaml.exe independently or opening .xaml files via Windows Explorer bypasses necessary information exchange between Team Developer and Kaxaml; for example, location of internally-stored application images.
- If you make significant user interface changes in your Team Developer outline after you have customized your XAML files, update your XAML files by following these steps:
  - a. Rename your .xaml directory so that the IL compiler will not recognize it.
  - b. Compile your project.
  - c. Right-click a parent window in your outline, select Custom XAML and then Edit custom XAML. This will create new .xaml files.
  - d. Using a comparison tool, compare the new .xaml files with your old customized files (in the directory you renamed). You will see differences that correspond with the changes you made in Team Developer, and you will also see differences where you customized your old files.
  - e. Copy the desired customizations from your old customized files to the new files. The files in the .xaml directory are now customized.
  - f. Compile your project. The IL compiler uses the files in your .xaml directory.
  - g. (Optional) Delete the directory you renamed in step a.
- Do not make the following types of changes in your custom XAML files:
  - ▶ Do not add new controls that do not correspond to controls in your Team Developer application outline.
  - ▶ Do not rearrange the order of controls.
  - ▶ Do not rename controls.
  - ▶ Do not change event handlers.

### More Information about XAML

For more information about XAML in WPF, see Microsoft’s documentation at <http://msdn.microsoft.com/en-us/library/ms747122.aspx>

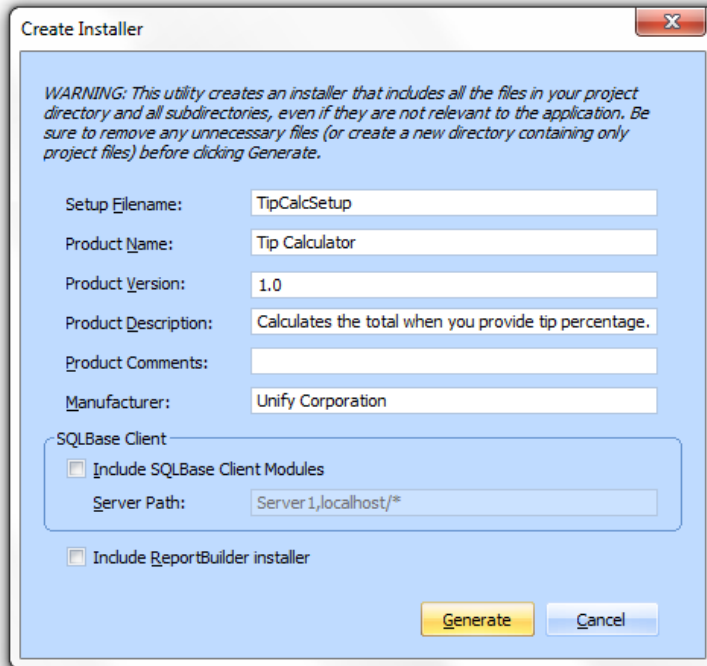
---

## Deploying WPF Desktop Applications

To deploy your WPF desktop applications to users, you must create an installer by following these steps:

1. Select **Create Installer** from the Project menu. This menu option is disabled unless your build target is set to WPF Desktop.

The Create Installer dialog box opens.



The fields in this dialog box are automatically populated with information you entered in the Version tab of the Build Settings dialog box. Most of these fields are optional. Add, delete, or edit text as you see fit.

2. If your application requires SQLBase connectivity, check the “Include SQLBase Client Modules” check box. Otherwise, leave this box unchecked.

Your project’s sql.ini file is packaged in the installer. Therefore, if your sql.ini uses “localhost” in its serverpath setting, you must change this setting to point to the SQLBase database you plan to use; for example, an IP address.

3. If your application requires Report Builder functionality, check the “Include ReportBuilder installer” check box. This allows your users to install a limited version of Report Builder. They can access all the functionality required for your application, but cannot create new reports or run Report Builder independently. If your application does not require Report Builder functionality, leave this check box unchecked.
4. Click **Generate** and wait a few moments. The “Generate Finished” message appears, and the installer is placed in your application directory.

---

## WPF Browser Applications in Firefox

To use the Firefox browser to run WPF Browser applications, you must install a Firefox plugin. To check if you have the plugin, use the script at <http://msdn.microsoft.com/en-us/library/bb909867.aspx>.

If you do not have the script, do the following:

- Windows XP/Vista Users

Install the latest .NET 3.5 framework (with available service packs). Try the script from the link again. If the script still indicates a problem, check the following directory:

C:\Windows\Microsoft.NET\Framework\v3.5\Windows Presentation Foundation

The .NET framework installation should have placed a file named NPWPF.dll in that directory. Copy that file to C:\Program Files (x86)\Mozilla Firefox\plugins.

- Windows 7 Users

The .NET 3.5 SP1 framework cannot be installed on a Windows 7 machine. To obtain the necessary plugin, you must acquire NPWPF.dll—for example, from a Windows XP or Vista machine—and copy it to the appropriate directory.

---

## WPF Browser Applications in Internet Explorer

Internet Explorer 8 provides an excellent host for WPF browser applications. However, an issue can arise when using tabbed browsing and closing the tab that is running the application. To avoid this problem, close Internet Explorer completely when you want to close the application.

On a 64-bit Windows machine, 64-bit Internet Explorer is not the default version. Users need to navigate to Internet Explorer 64-bit to run Team Developer WPF 64-bit applications. 64-bit Internet Explorer is located at %ProgramFiles%\Internet Explorer.



# Chapter 4 – Publishing XBAP Applications

This chapter contains instructions for publishing XBAP applications in WPF Browser.

---

## Publishing XBAP Applications (WPF Browser)

A successfully published application has a starting/index page that looks like the following:



To publish a WPF Browser application, execute the following steps.

### Build Settings for WPF Browser Application

If you are running Team Developer in Windows Vista or later, run as Administrator so that Team Developer has directory sharing permission.

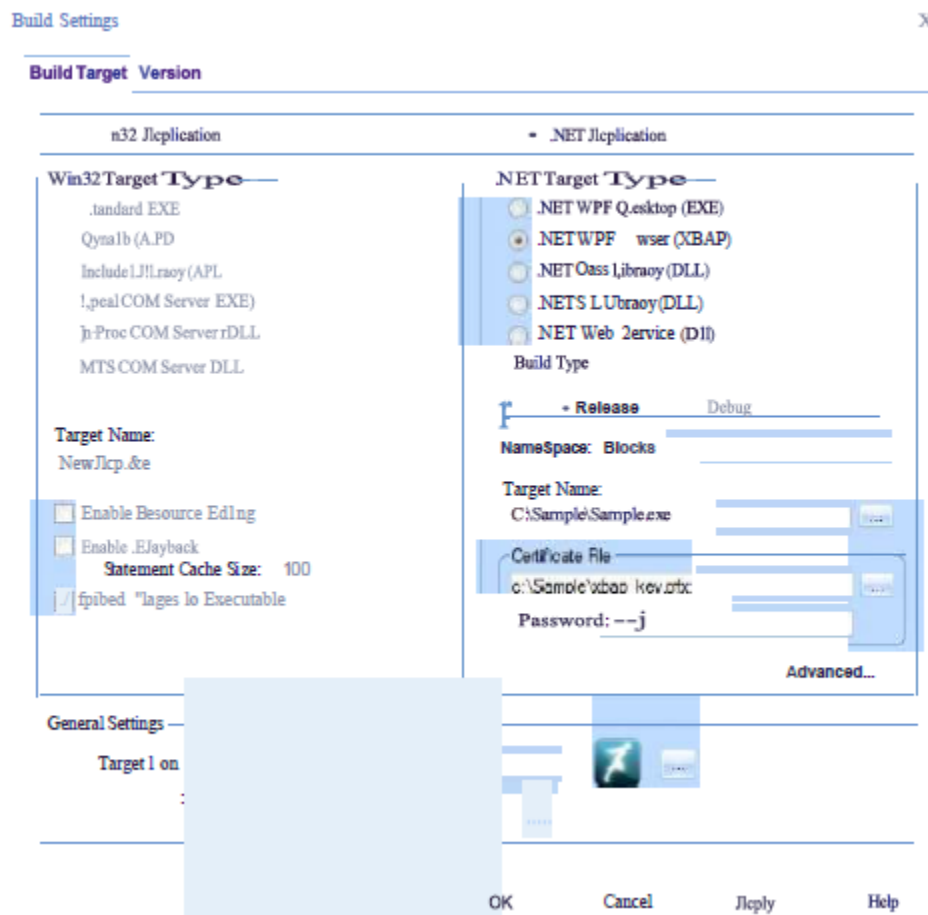
Certificate File settings are established in the Build Target tab in the Build Settings dialog. The certificate file fields are enabled when you choose .NET WPF Browser as the target type. By default, the compiler uses a private certificate. This certificate is saved in the build directory specified in the Target Name field, and has the file name xbp\_key.pfx. It is necessary to import this certificate for a WPF browser application to run from IIS. A WPF application will not download unless you import the certificate that is used to sign the application.

To import the certificate, follow these steps:

1. Open the Build Settings dialog by selecting **Project > Build Settings** from the menu bar.
2. Select **.NET Application**.
3. Select **.NET WPF Browser (XBAP)**.
4. Under **Target Name**, enter the directory where you want to build the application and click **OK**.

Update the Product Version (in the Version tab) each time you rebuild an application.

5. Build the application by selecting **Project > Build** from the menu bar.
  6. Open your application directory in Explorer, open the **xbap\_key.pfx** file, and configure the private certificate as follows:
    - a. On the initial screen click **Next**.
    - b. Click **Next** again.
    - c. Enter the password **12345** (see note below) and click **Next**.
    - d. Select **Place all certificates in the following store** and click **Browse**.
    - e. Choose **Trusted Root Certification Authorities** and click **OK**.
    - f. Click **Next**.
    - g. Click **Finish**.
    - h. Import the same certificate to Trusted Publishers by repeating steps a-h and selecting **Trusted Publishers** in step e.
- Note:** The private certificate provided with Team Developer has a set password of 12345. If you use a different certificate, you must also use the password of that certificate.
7. Go back to the Build Setting dialog, enter your certificate file's location and password, and click **OK**.



## Publishing a WPF Browser Application to IIS

You can deploy a WPF browser application to an IIS 6.0/7.0/7.5 server by selecting **Project > Publish** from the menu bar.

**Publish WPF Browser Application**

**IIS Settings**

Server Name:

User Name:

Password:

☒ Deploy Database Proxy Server

**Database Proxy Server**

URL:

**Publish**

You can specify the folder to publish on IIS server or local/remote file system.

Publish Location:

**Sign Application**

Certificate File:

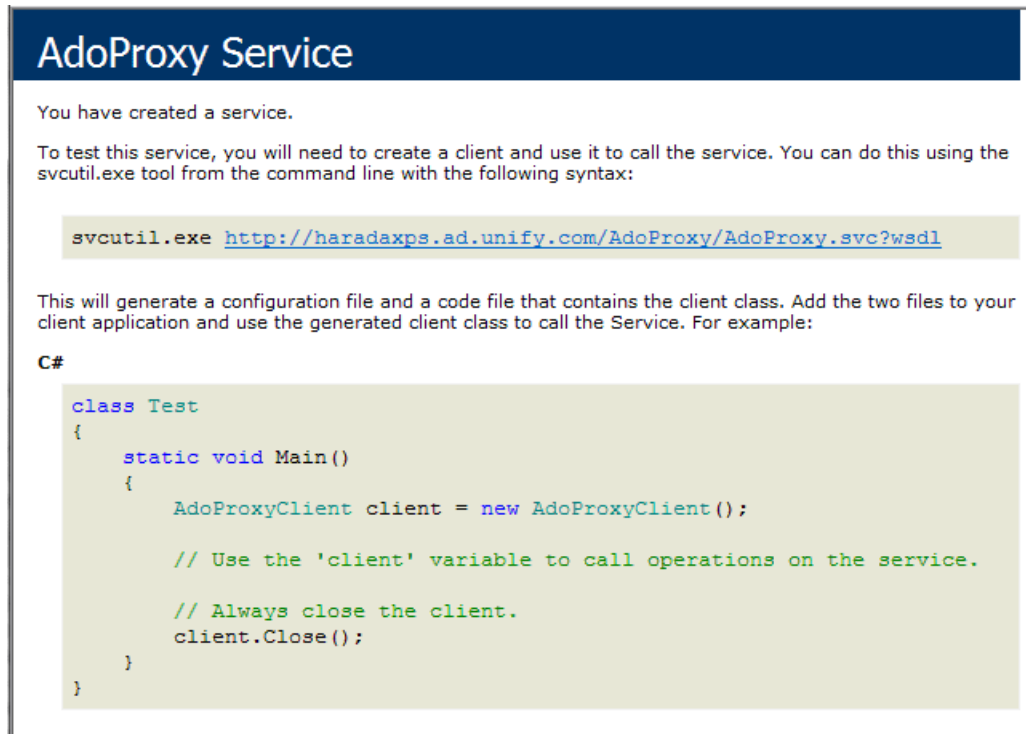
Password:

The fields have the following functions:

Field	Explanation
Server Name	IIS server name to deploy the WPF application (IIS 6, IIS 7)
User Name	User name who can log into specified IIS server. Must be a member of the Administrators group.
Password	Password for the user
Deploy Database Proxy Server	Check this to deploy the database proxy server to the same IIS server along with the application. The URL of the proxy service is <code>http://&lt;IIS_Server&gt;/DBPipeServer/DBPipeServer.svc</code> If you are using SQLBase through a remote Server, before publishing the XBAP application, edit <code>sql.ini</code> ([win32client.ws32] section) by commenting out "autostartserverpath".
URL (Database Proxy Server)	Specify the database proxy service URL that is already running.
Publish Location	You can specify the location on the IIS server to deploy the application. If this field is blank, the application is deployed to <code>&lt;IIS Root&gt;/&lt;Application Name&gt;</code> . In most cases <code>&lt;IIS Root&gt;</code> is <code>C:\InetPub\wwwroot</code> . You can also specify a local or remote folder when you do not specify an IIS server. This is useful when the WPF application has been deployed and you know its physical location.

Field	Explanation
Sign Application	Specify your own certificate here. Changes made in this field are reflected in the Build Settings dialog.

If the proxy server is running, the following page appears when you access the specified URL:



**AdoProxy Service**

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://haradaxps.ad.unify.com/AdoProxy/AdoProxy.svc?wsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

**C#**

```
class Test
{
    static void Main()
    {
        AdoProxyClient client = new AdoProxyClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

## IIS Troubleshooting

**Problem:** After installing the certificate and launching the application, an error similar to the following is reported:



**Solution:** Activate 32-bit applications by following these steps:

1. In the IIS GUI, navigate to the **Application Pool**.
2. Access **Advanced Settings**.
3. Set Activate 32-bit Applications to **TRUE**.

**Problem:** AdoProxy server is not working as a WCF (Windows Communication Foundation) service.

**Solution:** Because the default IIS configuration does not enable WCF, you must run the following command as administrator to change this configuration:

```
cmd /C %FrameworkDir%\v3.0\Windows Communication  
Foundation\ServiceModelReg.exe -iru
```

## Database Proxy Service

Database Proxy Server is a Web service that runs on IIS. To run database proxy service correctly, all necessary database client tools must be installed on the database proxy server machine.

### SQLBase Client Setup on IIS Server

Follow these steps:

1. Install Microsoft Visual C++ 2008 SP1 Redistributable Package (x86). This program is required to run SQLBase Client.
2. Install SQLBase Client .NET Data Provider ((Applicable for Team Developer 6.1 SP2 and earlier versions) as follows:
  - a. Download SQLBase 11.6 Standard for Windows from <http://www.guptatechnologies.com/Services/productDownloads.aspx>
  - b. Run **setup.exe**.
  - c. Select **Custom**.
  - d. Select the Feature **SQLBase 32bit Drivers > .NET Data Provider**.
  - e. Browse to the directory you want to install and finish installation.
  - f. Edit **sql.ini** (on IIS Server). Under [win32client.ws32] section, change localhost to the server address where SQLBase resides.
3. Create a SQLBase System Variable (Windows 7/Vista) as follows:
  - a. Right-click **Computer** and select **Properties**.
  - b. Select **Advanced system settings**.
  - c. Select **Environment Variables**.
  - d. Under System Variables, select **New**.
  - e. Under Variable name, enter **SQLBASE**.
  - f. Under Variable value: Path to the installed SQLBase Client, select **OK**.
4. Add SQLBase Client to the Path as follows:
  - a. Repeat steps 3a. thru 3c.
  - b. Under System variables, select **Path > Edit**.
  - c. Under Variable value, enter **%SQLBase%**; to the front of the Path and select **OK**.
  - d. Restart your IIS Server.

### Oracle Client Setup on IIS Server

Follow these steps:

1. Install oracle 11gR2 32bit client or server on IIS Server (database proxy server ).

2. Configure Oracle connection(s) as follows:
  - a. Go to C:\app\<user>\product\11.2.0\client\_1\network\admin.
  - b. Edit the tnsnames.ora file and add your Oracle connection(s).

### SQL Server Client Setup on IIS Server

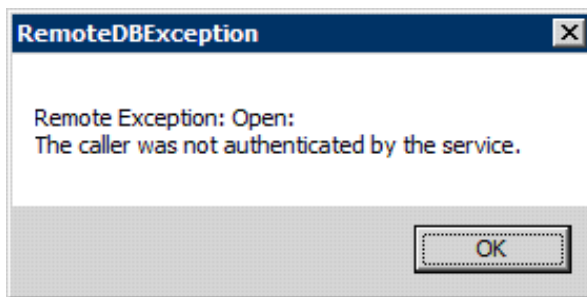
Refer to "Configuring an ODBC Data Source" in the *Connecting SQLWindows Objects to Databases* document.

Follow these steps:

1. Launch odbcad32 on Proxy Server machine.  
On a 64-bit Windows OS, go to C:\Windows\SysWOW64 and launch odbcad32.
2. Add system DSN(s) needed by the application.

### Notes regarding the proxy server

- Both the database proxy server and the client should be members of the same domain. Access from different domains or workgroups is rejected. The following error message will appear:



- XBAP applications are stored in a local cache on the client. If you try to run an xba application from IIS or local file system and get the following error, you must clear the local application cache.

"Unable to install this application because an application with the same identity is already installed. To install this application, either modify the manifest version for this application or uninstall the preexisting application."

If you make changes in your code/XAML that are not displayed when you run the app, clear the local application cache.

```
rundll32 %windir%\system32\dfshim.dll CleanOnlineAppCache
```

- The IIS Server machine must have .NET Framework (version 3.5 SP1) installed.

# Chapter 5 – WPF Controls

This chapter explains how to implement WPF controls in your applications, change their properties, and invoke their methods.

In many cases, WPF controls serve as replacements for visual ActiveX controls. Most ActiveX controls are old implementations of gauges, HTML viewers, and other controls. These controls usually have an old-fashioned look and feel. Many third-party WPF controls, however, implement similar functionality with a modern look and feel. WPF controls make it easy to replace ActiveX controls rather than preserve the outdated ActiveX controls.

In other words, WPF controls are often available online, they look great, and their implementation is a preferable alternative to transferring an outdated ActiveX control to a .NET application.

---

## Custom WPF Control in Team Developer

Team Developer contains a control called a Custom WPF Control. This control is much like a frame with an added, important attribute: Xaml. To use a WPF control (of any kind) in your application, simply add a “Custom WPF Control” and then use the Xaml attribute to specify the type of WPF control you will be implementing. The Xaml attribute is a string containing a xaml fragment for a single WPF control. This control could be a container that in turn contains one or more additional controls. The xaml fragment must contain xml namespace declarations for any namespaces that are referenced.

### WPF Functions

The following functions are provided for retrieving and setting the values of properties in a WPF control:

SalWPFGetBoolProperty	SalWPFSetBoolProperty
SalWPFGetDateProperty	SalWPFSetDateProperty
SalWPFGetNumericProperty	SalWPFSetNumericProperty
SalWPFGetStrProperty	SalWPFSetStrProperty

In addition, SalWPFInvokeMethod enables you to call methods for WPF controls from within your Team Developer code.

See the online help for documentation on each of these functions.

### WPF Events

The SAM\_WPFEvent message appears whenever an event occurs in any WPF control. Thus, any WPF control can use SAM\_WPFEvent and the associated wpf\*keywords.

This event has three keywords:

- wpfEventName – A string that indicates which event occurred.
- wpfEventCancelled – Boolean that can be set to cancel the event.

- `wpfEventHandled` – Boolean that marks the event as “handled” in order to inhibit unwanted interference.

The following is an example of the usage of this event and its `wpfEventName` keyword:

```
WPF Custom: xSlider Message Actions
On SAM_WPFEvent
    If wpfEventName ="ValueChanged"
        Call SalWPFGetNumericProperty( xSlider, "Value", nSkX ) Set
            sArgsTransform[0] =SalNumberToStrX(nSkX, 0 )
        Call SalWPFInvokeMethod(wpfTransformMyTD, "setTransform",
            sArgsTransform, sReturn)
```

## Debugging Applications with WPF Controls

Some WPF controls, such as media players based on Media Element, do not honor the “Current Directory” when the application is in debug mode. Thus, for WPF controls to work consistently at debug time, you must provide proper URLs for source files, including a full path to the file required.

---

## Gauges

One type of WPF control is the gauge. Gauges provide a visually interesting display of numerical data, and the wide variety of WPF gauges available makes them a versatile option.

Gauges have the following properties. All of these properties have string representations, which means they can be set in the control’s XAML property or via `SalWPFSetStrProperty`.

### Properties for all gauges

Property	Type	Comments
Min	Double	Minimum Value
Max	Double	Maximum Value
IsInteractive	Bool	Indicates whether the user can change the value by dragging the needle, bar, etc.
IsLogarithmic	Bool	Indicates that the gauge should be logarithmic
IsReversed	Bool	Reverse the gauge
Location		ScaleObjectLocation
LogarithmicBase	Double	
MajorTicks	Double	The number of major ticks in the gauge. "Major Ticks" are those specifying the largest size grouping. Thus, if the entire gauge range consists of 0 through 100, and MajorTicks is set to 4, there will be four major groups inside that range of 0-100, which will result in tick marks at 25, 50, and 75. In most gauges, the major tick marks are the locations where an actual numeric value is drawn on the scale.



Property	Type	Comments
MiddleTicks	Double	Each major tick section can be subdivided into Middle Ticks. The MiddleTicks property specifies how many subgroups are displayed inside each major tick group. A MiddleTicks setting of 2 results in one medium-sized tick mark drawn between each major-sized tick mark (one medium tick dividing the two medium tick groups).
MinorTicks	Double	Similar to MiddleTicks. The MinorTicks property indicates how many subgroups are displayed within each middle tick group.
ShowFirstLabel	Bool	Indicates whether the first label should be shown
ShowLastLabel	Bool	Indicates whether the last label should be shown
Value		Double The value of the gauge
IsFlat	Bool	Use a flat, 2D style

### Properties for Radial Gauges, Range Gauges, and Flat Range Gauges

Property	Type	Comments
Radius	Double	The radius
StartAngle	Double	Start angle in degrees (0 = right, 90 = down)
SweepAngle	Double	Number of degrees from the start to the end

### Properties for Range Gauges and Flat Range Gauges

Property	Type	Comments
RangeList	RangeList	List of ranges; for example: "Red-0-300;Yellow-301-600;Green-601-1000")

The RangeList property describes the color ranges for a graph.

The string representation has the following format, where <brushn> is any string that can be used to describe a brush in xaml, and <Minn> and <maxn> are the minimum and maximum values for the range:

```
<brush1>-<min1>-<max1>;<brush2>-<min2>-<max2>
```

---

## Examples

For examples of how to implement WPF controls in your applications, change their properties, invoke their methods, and so on, see the sample applications in your [Team Developer]\Samples\ directory.

# Chapter 6 – Connectivity in .NET Applications

This chapter explains how to connect to Oracle, SQL Server, and SQLBase in .NET applications.

---

## Connecting to Oracle

- When developing WPF applications that go against an Oracle database, you must use the Oracle 11gR2 32-bit client.
- The Oracle11gR2 32-bit client can be used to go against an oracle10g or oracle11g database.
- There is no need to configure an Oracle .NET connection because Team Developer automatically maps your native and OLEDB connections to equivalent .NET connections.

To configure an oracle OLEDB UDL file, follow these steps:

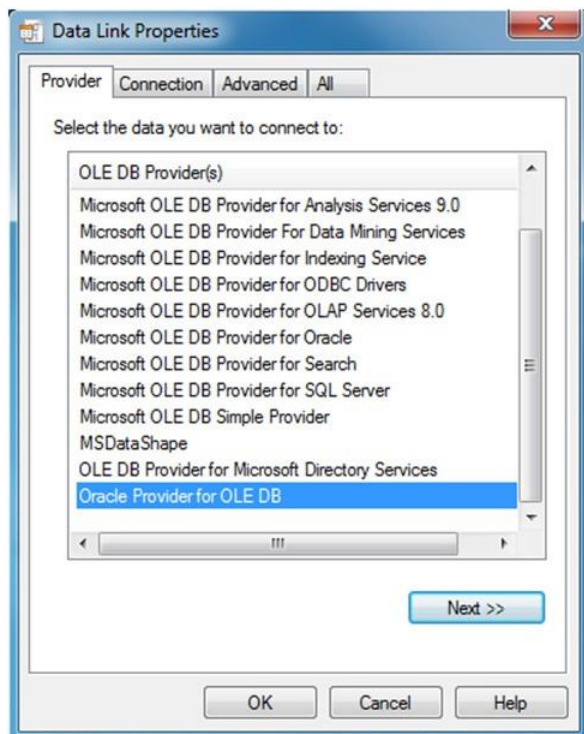
1. Open the command prompt and register the Oracle 11gR2 32bit OLEDB driver:

```
regsvr32 C:\app\<users>\product\11.2.0\client_1\BIN\OraOLEDB11.dll
```

2. Launch the "Data Link Properties" dialog for the udl file you want to configure:

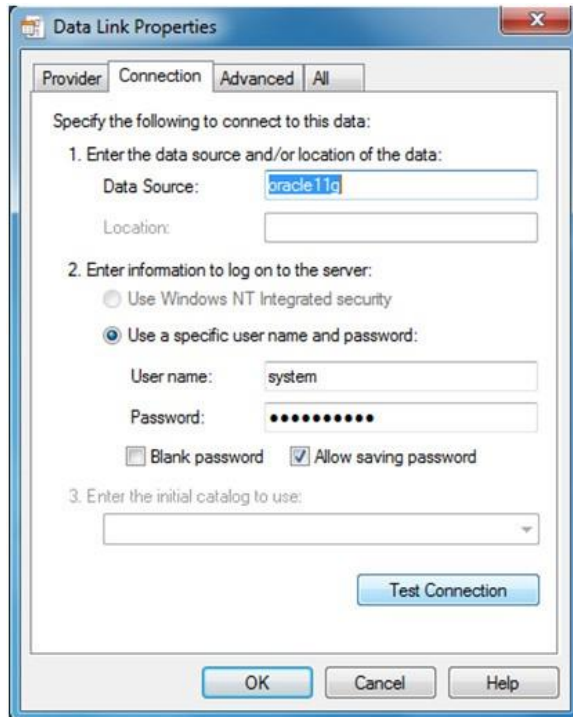
```
rundll32.exe "%ProgramFiles(x86)%\Common Files\System\OLEDB\oledb32.dll",  
OpenDSLFile "<path to udl file>\<udlfile>.udl"
```

3. Under the Provider tab, select **Oracle Provider for OLE DB**.



4. Click the Connection tab and enter your Data Source name, username, and password.

5. Click **Test Connection**.



On a 64-bit Windows OS, run regsvr32 and rundll32 from C:\Windows\syswow64.

---

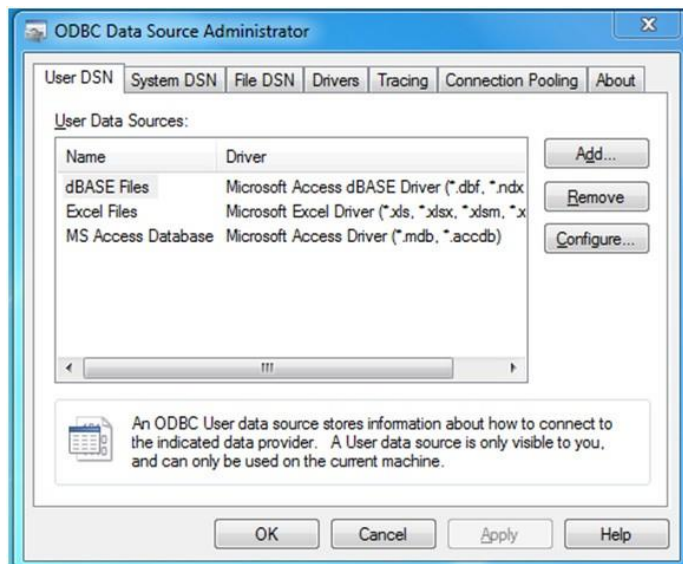
## Connecting to SQLServer

Your SQLServer standard/Native ODBC and OLEDB connections automatically map to equivalent .NET connections.

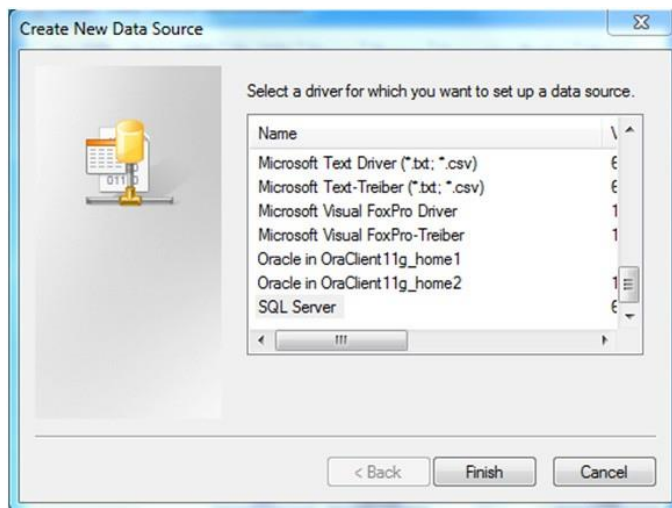
### ODBC

To set up an odbc connection, follow these steps:

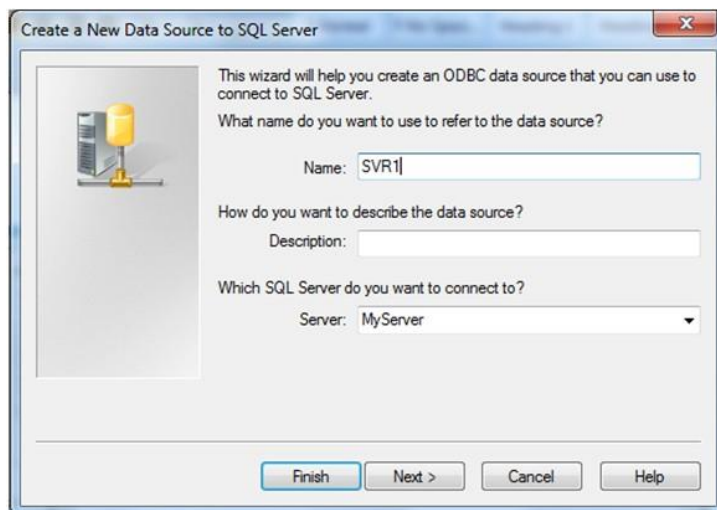
1. Open the command prompt and run **odbcad32**.  
On a 64-bit windows OS, run odbcad32 from the syswow64 directory.
2. In the ODBC Data Source Administrator window, click the **Add** button.



3. In the resulting Create New Data Source dialog, scroll down to SQL Server (for standard odbc connection) or Native SQLServer (for native odbc connection) and select **Finish**.



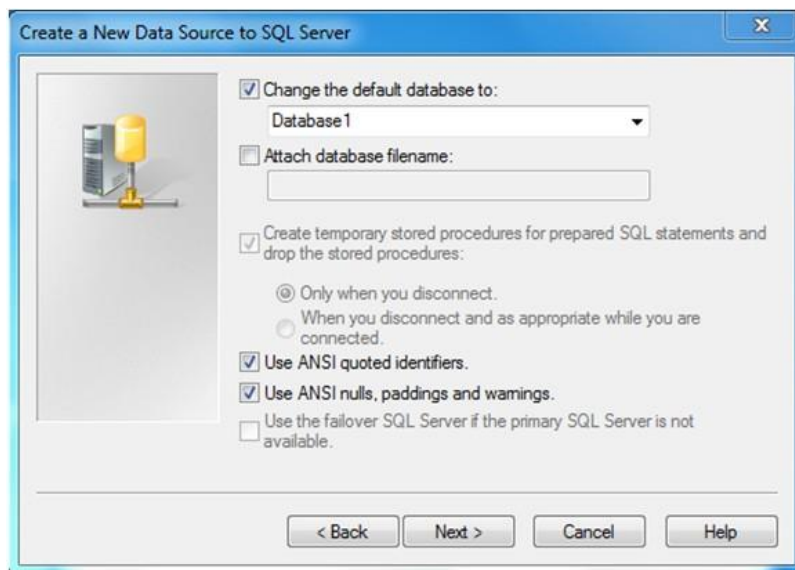
4. In the Create New Data Source to SQL Server dialog, enter a reference name for the server and a description, select the server you want to connect to, and select **Next**.



5. Select **With SQL Server authentication using a login ID and password entered by the user**, enter the Login ID and Password, and click **Next**.



6. Check the "Change the default database to" check box, select your database from the dropdown, and click **Next**.



7. Click **Finish**.
8. Select **Test Data Source** and, if the test is successful, click **OK**.
9. Exit the ODBC Data Source Administrator and open up the sql.ini in your Team Developer directory.
10. Uncomment the line of code "comdll=sqlodb32" and insert the following line of code under "[odbcctr]" using the reference name you gave your server:

```
remotedbname=[RefName],dns=[RefName] (for example,  
remotedbname=[SVR1],dns=[SVR1])
```

## SQL Server TIMESTAMP

Due to strict Data Type Matching in WPF applications, SqlServer TIMESTAMPS (which are often used as Row IDs) are handled somewhat differently than they are in WIN32. Because TIMESTAMPS are binary data types,

to bind a timestamp to a string you must call "SetLongBindDatatype" after running SqlPrepare and before running SqlExecute, as in the following example:

```
Call SqlPrepare(hSql,"SELECT ROWID INTO :dfString FROM TEST") Call  
SqlSetLongBindDatatype(1, 23)  
Call SqlExecute(hSql)
```

The resulting string will be a string of bytes packed into a double-byte Unicode string. It will not be human readable, but can be used to compare the contents of one row to another.

---

## SQLBase

.NET connectivity in SQLBase is a mirror of native. No additional actions are required on the part of the user.

# Chapter 7 – .NET Explorer

This chapter describes .NET Explorer, the assemblies you can access, and the libraries you can build with it.

.NET Explorer is a powerful tool that allows you to generate include libraries (APLs) from existing .NET assemblies. The resulting APLs can be included in your Team Developer application whether your build target is Win32 or .NET.

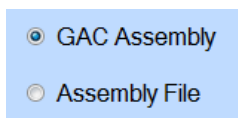
## Using .NET Explorer

Follow these steps:

1. In the Tools menu, select **.NET Explorer**. The following window opens:



2. Click **Next**.
3. The next screen asks you to select GAC Assembly or Assembly File.

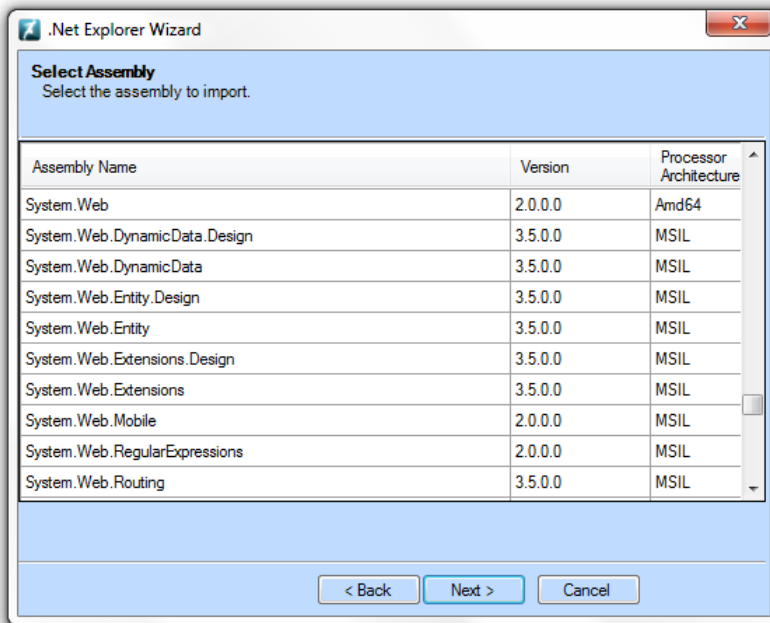


- ▶ GAC Assembly – Choose this option to import an assembly from the Global Assembly Cache (a collection of .NET assemblies that exist on your machine).
- ▶ Assembly File – Choose this option to import a custom assembly.

Make your selection and click **Next**.

4. The next window differs according to the option you selected in the previous step.

If you chose GAC Assembly, you will see a screen similar to the following:



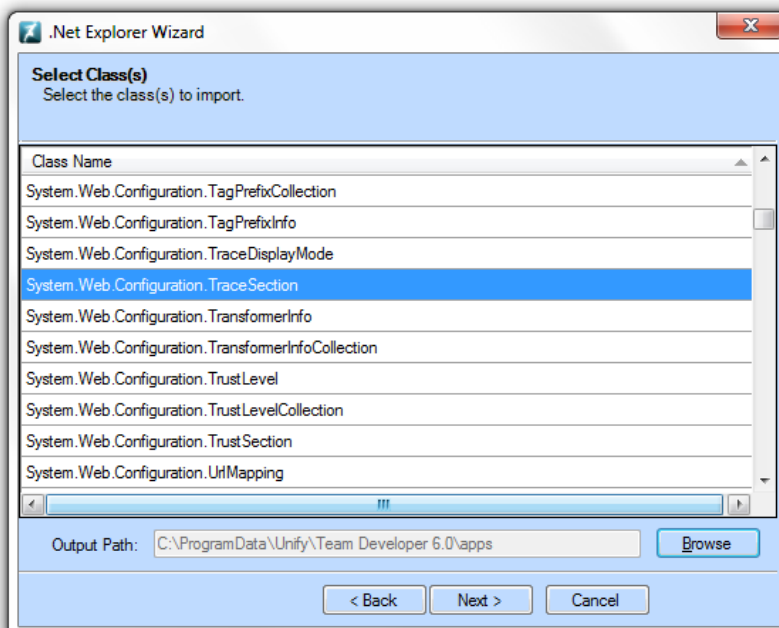
If you chose Assembly File, you will see a data field and Browse button where you can indicate the location of the assembly file.

This step and the next step assume that you chose GAC Assembly. See steps 6 and 7 for information that applies to either option.

Select the assembly you want to import and click **Next**.

5. The next screen lists classes contained in the assembly you selected.

Select the class(es) you want to import. Use Shift+click, Ctrl+click, or click and drag to select multiple classes.



You must also indicate the output path for the APL that will be generated. Type the path in the Output Path field or use the Browse button to navigate to the desired directory.



Click **Next**.

- When you see the following message, click **Finish**.

.NET proxy has been generated successfully.

.NET Explorer closes and, assuming Team Developer is still the most recently active application, returns you to your application outline.

The generated APL is not automatically included in your outline. Therefore, repeat steps 1-6 to generate as many APLs as you want.

**Note:** .NET assemblies (DLLs) created in Visual Studio must be generated with target framework set to .NET Framework 3.5 or .NET Framework 4.0. Otherwise, .NET Explorer displays the following error message when you try to import the DLL: "Could not load the selected assembly because its CPU architecture is not supported."

- To include the generated APL, right-click the Libraries section of your outline, select **Add Next Level** and **File Include**.
- Browse to the location you indicated in the Output Path (step 5), and select the new APL file. If it does not reside in the same directory as your application outline, click the "Append Path To File Name" check box, and click **Open**.

---

## About AXLs, APLs, DLLs

### Definitions

In this chapter, AXL, DLL, and APL files are defined as follows:

- AXL.** An AXL file is a description (written in XML) of the symbols being imported from the external assembly. It must be included for a .NET project.
- DLL.** A DLL is a library that contains compiled code for functions that will be called from the application.
- APL.** Like a DLL, an APL is a library that contains code for functions that will be called from the application.

### .NET Explorer Generates APL and AXL

When you import a .NET assembly and create an APL via .NET Explorer, an AXL file of the same name is also generated. As explained previously, the APL file is the one you include in your outline. When you include the APL, Team Developer automatically imports the AXL file as well. You can see this in the External Assemblies section of the outline.

Technically, the AXL is only needed if your finished product will be a .NET application. The APL is only needed if your finished product will be a Win32 application. Both files are included in the outline for two reasons:

- With both files included, you can compile your application to .NET or Win32 without changing your file inclusions.
- The Coding Assistant draws upon the APL for function names, parameters, etc. Thus, if you only include the AXL file, the coding assistant's features will be limited.

## **.NET SAL Library Generates DLL and AXL**

When you create a .NET SAL Library with Team Developer, the compiled project consists of two files: a DLL and an AXL. In this case, you will include the AXL file (in the External Assemblies section of your outline). Team Developer looks in the same directory for the DLL and automatically includes it.

---

## **Variables Based on Imported .NET Classes**

When you create a variable based on one of the imported .NET classes, the instantiation of the variable is not always automatic.

When using a Win32 build setting, you need to explicitly instantiate your user-defined variable (UDV) by calling one of the constructor methods in the code generated by .NET Explorer. There will be one method for each version of the constructor available.

When using a .NET build setting, the generated code will automatically instantiate the UDV if there is a default constructor available. If there is no default constructor available, then you will have to explicitly call one of the constructor methods, as in Win32.

There is no harm in calling the method for the default constructor, even if it was already called automatically. In this case, the UDV will simply be re-instantiated unnecessarily. However, since you must call the default constructor explicitly in order to compile to Win32, calling the default constructor method explicitly will allow you to compile your code in both build settings with no change in the code.

Users wanting to resize a native .NET array must do so within their external code. With the exception of `SalArrayGetUpperBound`, all `SalArray` functions reports an error when compiling to .NET applications.

---

## **.NET Assemblies Created in Visual Studio**

Using Visual Studio 2008 and Visual Studio 2010, users can generate Class Libraries in any CLR language (C++, C#, VB.NET). .NET Explorer can be used to generate equivalent APLs and AXLs that can be used in Team Developer applications.

# Chapter 8 – Debugging DLLs

This chapter describes various ways to debug .NET Class Libraries and Web services from with Team Developer .NET applications. SQLWindows allows customers to build .NET applications or libraries as 64-bit, but, for debugging purposes, you should choose 32-bit as the target CPU type.

---

## .NET Class Libraries

Classes provide reusable code in the form of objects. A class library contains one or more classes that can be called on to perform specific actions. Users can develop .NET class libraries using Visual Studio or SQLWindows. Team Developer supports importing class libraries that are targeted to either the 3.5 or 4.0 version of .NET Framework. For more details on how to make class libraries using Visual Studio, visit:

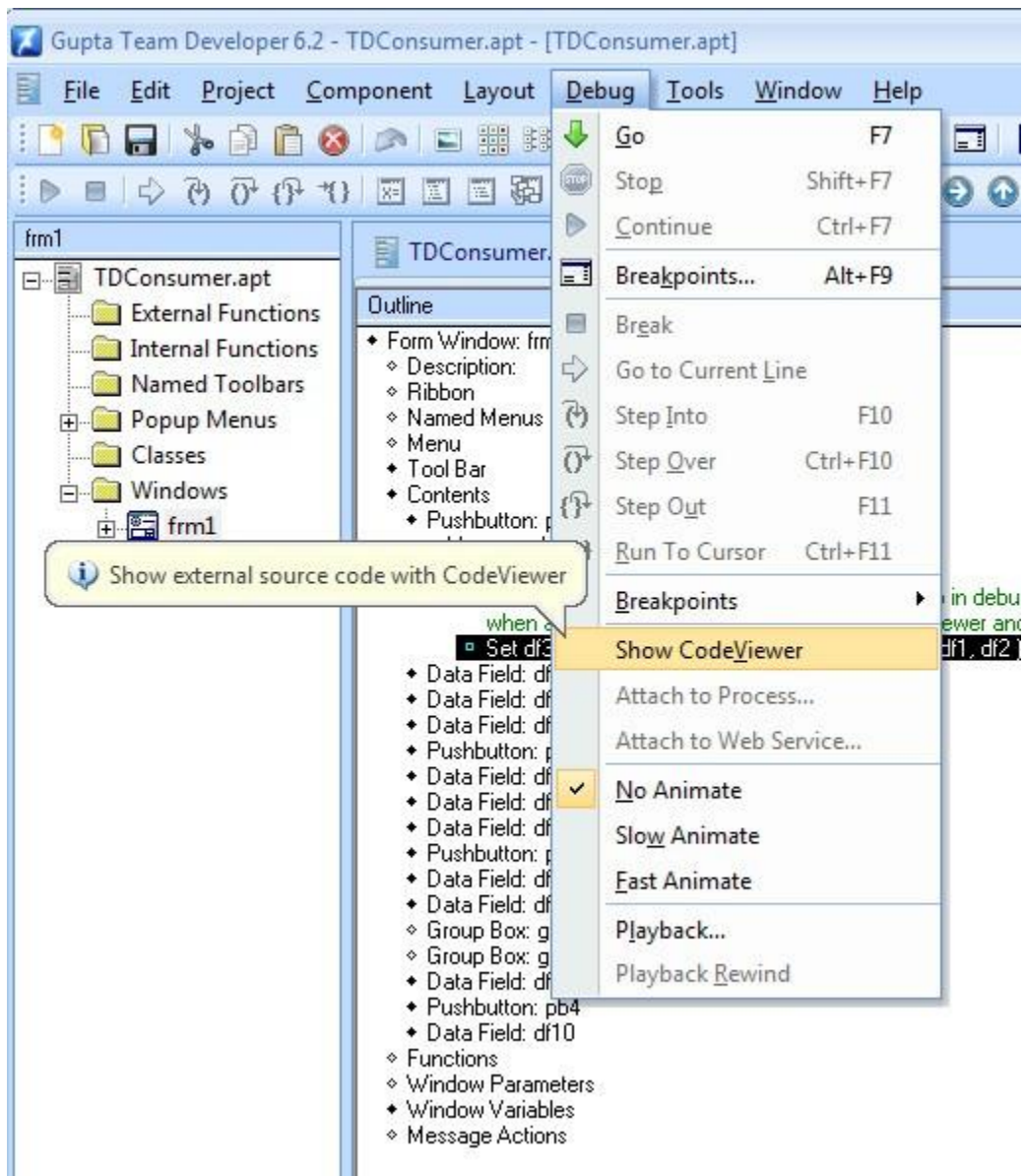
[http://msdn.microsoft.com/en-us/library/f3cye135\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/f3cye135(v=vs.90).aspx)

For more information on how to make class libraries using SQLWindows, check Team Developer IDE .NET build settings.

To use and debug .NET class libraries in Team Developer .NET applications, follow these steps:

1. Build the class library using Visual Studio. Make sure to generate a PDB file for the library.
2. Use .NET Explorer and import the class library generated in Step 1 to create corresponding APL and AXL files.
3. Open the Team Developer application that is using the library and set the breakpoint at the function call. Make sure build settings for the application are set to either .NET WPF Desktop or .NET WPF Browser.
4. Press F7 from TD IDE and run the application in Debug mode. When the breakpoint is hit, click **Step-In** and observe that SQLWindows opens the corresponding library source file in a separate code viewer.
5. Keep clicking **Step-In** and observe how the focus in the separate code viewer moves as it executes the code functions.

Alternatively, the user can open a separate code viewer by selecting the “Show CodeViewer” menu item under the popup menu “Debug” in the IDE as the following shows:



The following shows the result of when SQLWindows steps into other sources:

```

0001 ?b1ic Class Class1
0002     ?ub t.c !"-;Lnq:lon ;;co ($yVq:l n-;m);:=hc;c;::,
0003         syva:nu.. z ;;s ne c;::);;s lnē e:::
0004         C: l ;;neq:::
0005     De
0006         Add = num1 + num2
0007         i = i + 1
0008     Loop Until i > 10
0009
0010     Re: ampe'-dd
0011     :!d. tme e"!l.
0012
0013     ?ub !e Fu"!et!ot! Co:"et n-. H;S'!:-!nQ(3yValX' ;,s    s := :!Q', ayva.1 y
0014     Re"Cum:-n x - y
0015     :!d :tme! , !\
0016     :id Class
0017

```

GUPTA TEARN DEVELOPER 62 [Break] - TDCONSUMER.APT - (TDCONSUMER.APT)

9... TDC on summer apt  
 ....E:J External Functions  
 !.....12:1 Internal Functions  
 ...12:1 Named Toolbars  
 © ..12:1 Popup Menus  
 !:!! Classes  
 El Cl Windows  
 © \$fmr1

- Application Description: Unify SQLWindows  
Standard Application Template
- Libraries
- GlobalDeclarations
- Form Window: frm1
  - \* Description:
  - Ribbon
  - \* Named Menus
  - \* Menu
  - ToolBar
  - Contents
    - Pushbutton: pb1
    - Data Field: df1
    - Data Field: df2
    - Data Field: df3
    - Pushbutton: pb2
      - Message Actions
        - On SAM Click
          - Setdl6 M•DotNetClass .6.ddl dl4 dl5
    - Data Field: df4

---

## .NET Web Services

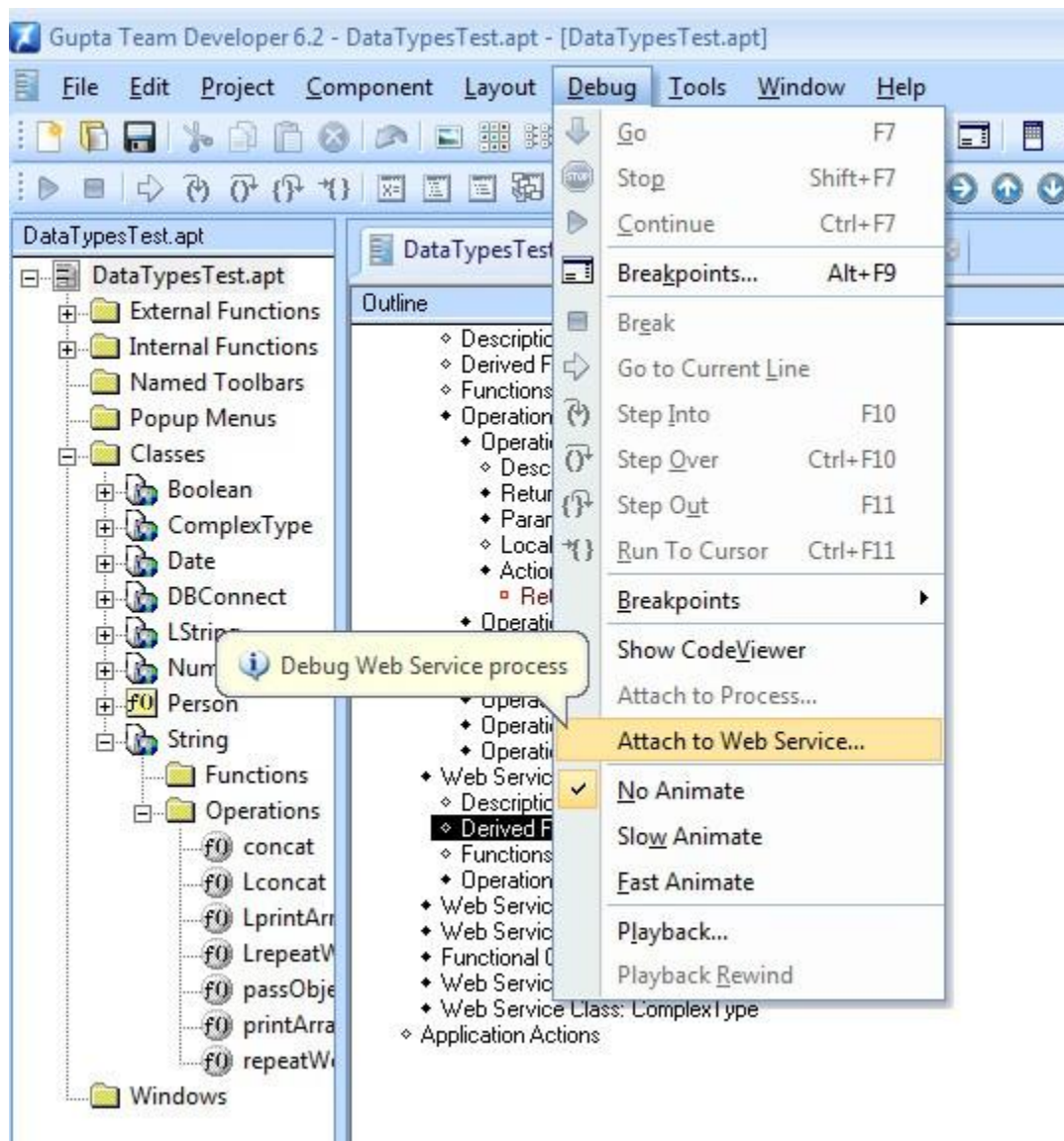
SQLWindows allows debugging of .NET Web services that are created with SQLWindows and hosted onto IIS. Refer to WebServicesPart1.PDF and WebServicesPart2.PDF for more information on how to create and use .NET Web services with Team Developer.

To debug .NET Web services hosted on a local machine, follow these steps:

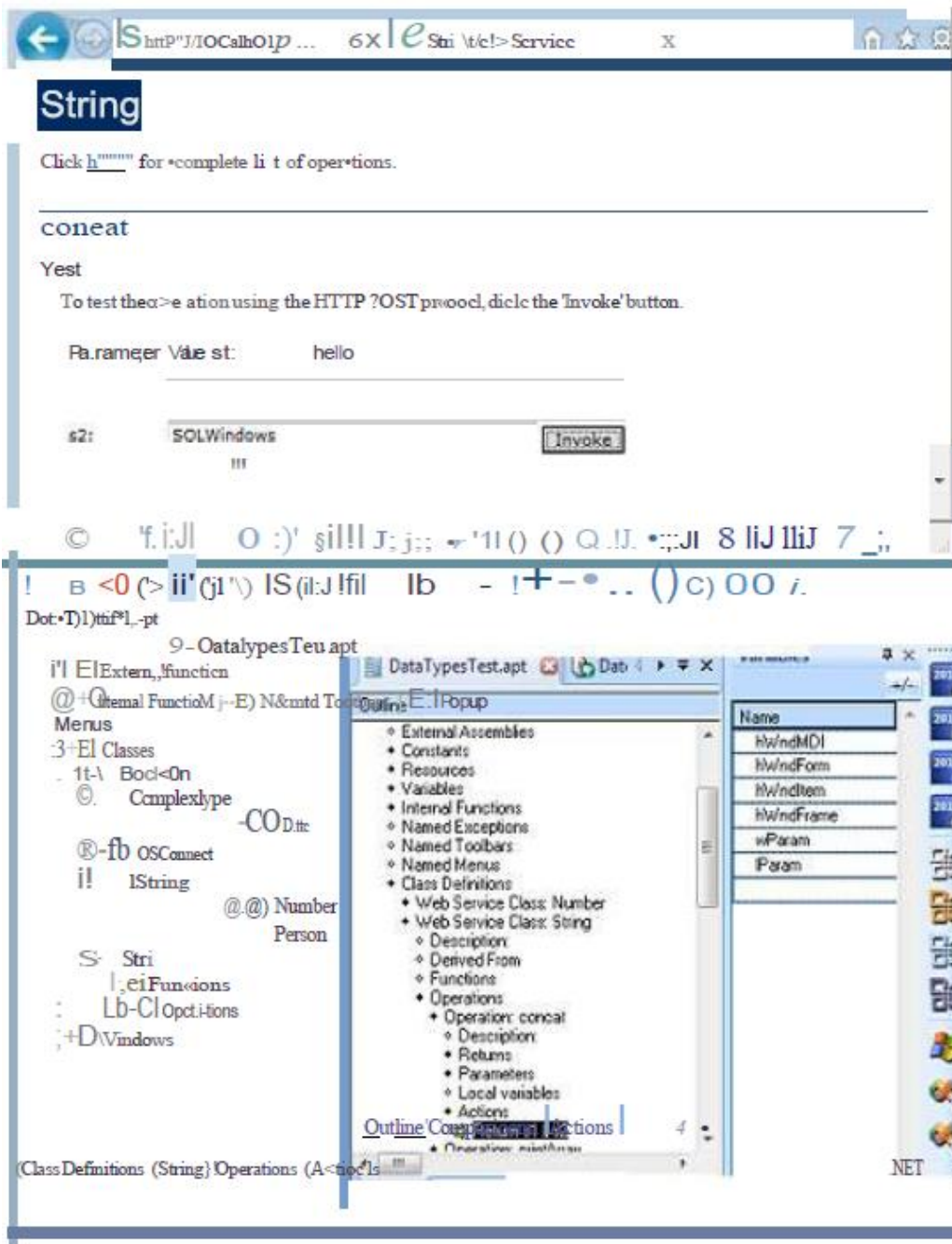
1. Create a Web service dll and corresponding asmx file using SQLWindow's .NET WebServices build setting (DataTypesTest.apl).
2. Host the generated Web service onto IIS .
3. Open Team Developer as Administrator.
4. Open the Web service source DataTypesTest.apl from the IDE.
5. Select the popup menu Debug and the "Attach to web service" menu item. This command attaches the current code to the Web service process (w3wp).
6. Access the Web service from Internet Explorer using the corresponding asmx file and observe that Team Developer steps into the current code.

Currently, the new debugger can attach/debug only in 32-bit because Team Developer is a 32-bit application. SQLWindows allows customers to build .NET applications or libraries as 64-bit. However, for debugging purposes, you should choose 32-bit as the target CPU type. The following is an illustration.





The following illustrates a Web Service debug:



IV OS3

## .NET SAL Libraries

.NET SAL libraries are equivalent to Dynamic Libraries (.APD) in WIN32. To debug a SAL library, users need a consumer application that includes .NET SAL libraries.

1. Run the consumer application (WPF EXE).
2. Open .NET SAL library source in Team Developer IDE as administrator.
3. From Team Developer , select the "Debug" popup menu, the "Attach to Process" menu item, and the consumer application EXE.



4. Step into the application when the breakpoint is hit.

The menu item “Attach to the process” can be seen in the following:

