TEAM DEVELOPER™

.NET Projects

Product Version 6.3

GUPTA

Team Developer™: .NET Projects, Product Version 6.3

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.

Last updated: November 11, 2014.

Legal Notice

Gupta Technologies, Inc.
1420 Rocky Ridge Drive, Suite 380
Roseville, CA 95661

Gupta Technologies.com

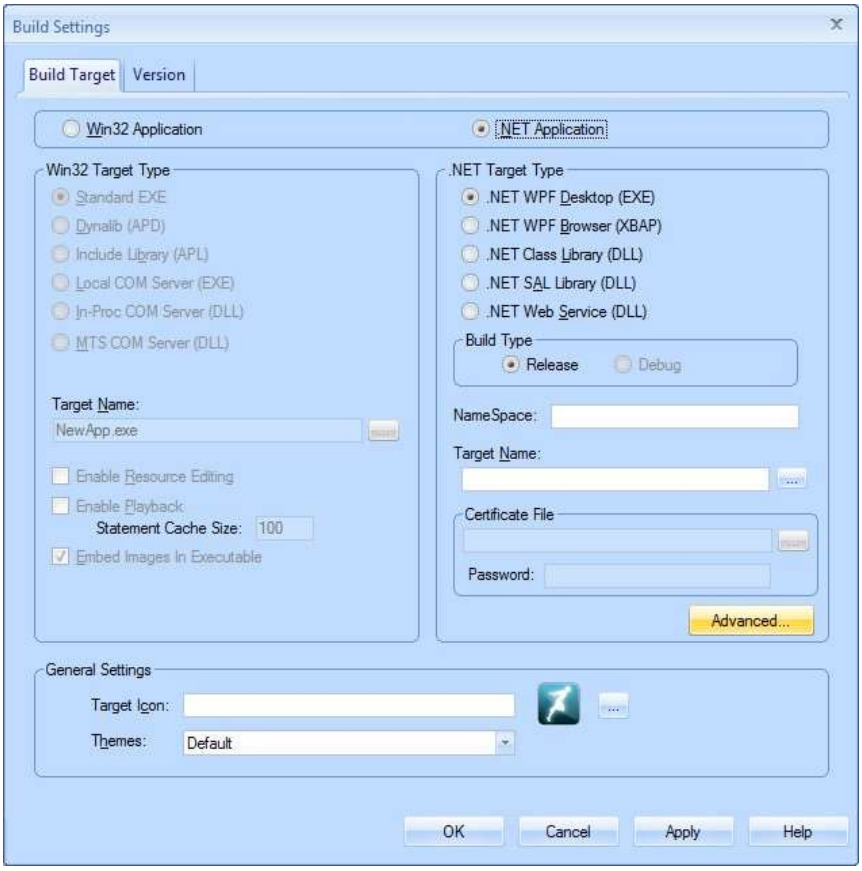
Table of Contents

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 *Project menu* on page *4-14* of the document entitled *Developing With SQLWindows*). The right side of this dialog box contains options for .NET build settings. These options and settings are explained below.



.Net Target Types

NET WPF Desktop - A WPF application that runs from the desktop.

NET WPF Browser - A WPF application that is designed to run in a browser.

NET Class Library - A class library that will be available for use in developing other applications.

NET SAL Library - This is the WPF equivalent for the dynalib compiler option in win32 mode.

NET Web Service- A .NET Web Service that can be hosted on IIS.

Note: You can also compile from the command line, including parameters to specify the build target. See the section entitled "What You Do" below for examples.

Other settings

Target Name - Use this field 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 will bring up a new Advanced Settings dialog box. In this dialog, the user can choose one of three values for Target CPU.

32 bit: The app will always be 32-bit. On a 64-bit OS, application run within the 32- bit subsystem.

64 bit: The app will always be 64-bit and can only run on a 64-bit machine.

Auto: The app 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". cbi62.exe -.net:WPFDesktop -

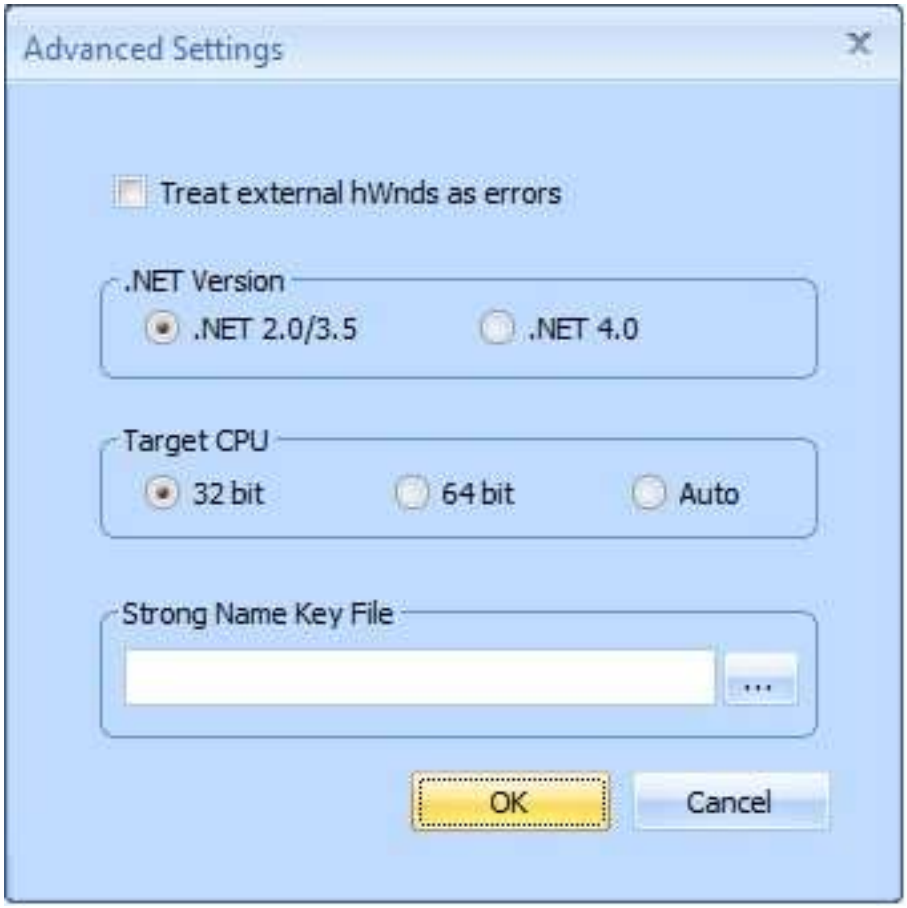
cpu:64bit test.apt test.exe

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 will bring up a new Advanced Settings dialog box. In this dialog, user can specify the Strong Name Key file (*.snk). The .NET compiler will sign the assembly using the .snk file..



Compiling Process Overview

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

What You Do

To create a .NET product with team developer, follow these steps:

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

Note: You can also compile from the command line. See *Compiling from the Command Line* for information and examples.

What Team Developer Does

Once you have selected a .NET target type, simply build your project as usual. Team Developer will execute the following steps in order to make your Team Developer outline into a .NET application, class library, or web service:

1. **Library Inclusions**

   Any libraries you have created and included are made available to the compiler.

2. **Functions from APLs.**

   Gupta-provided libraries are not included. Instead, .NET implementations of these libraries' functions have been created individually. Team Developer and the IL compiler ignore your include statements for these libraries and incorporate the .NET implementations where functions from these libraries are called.

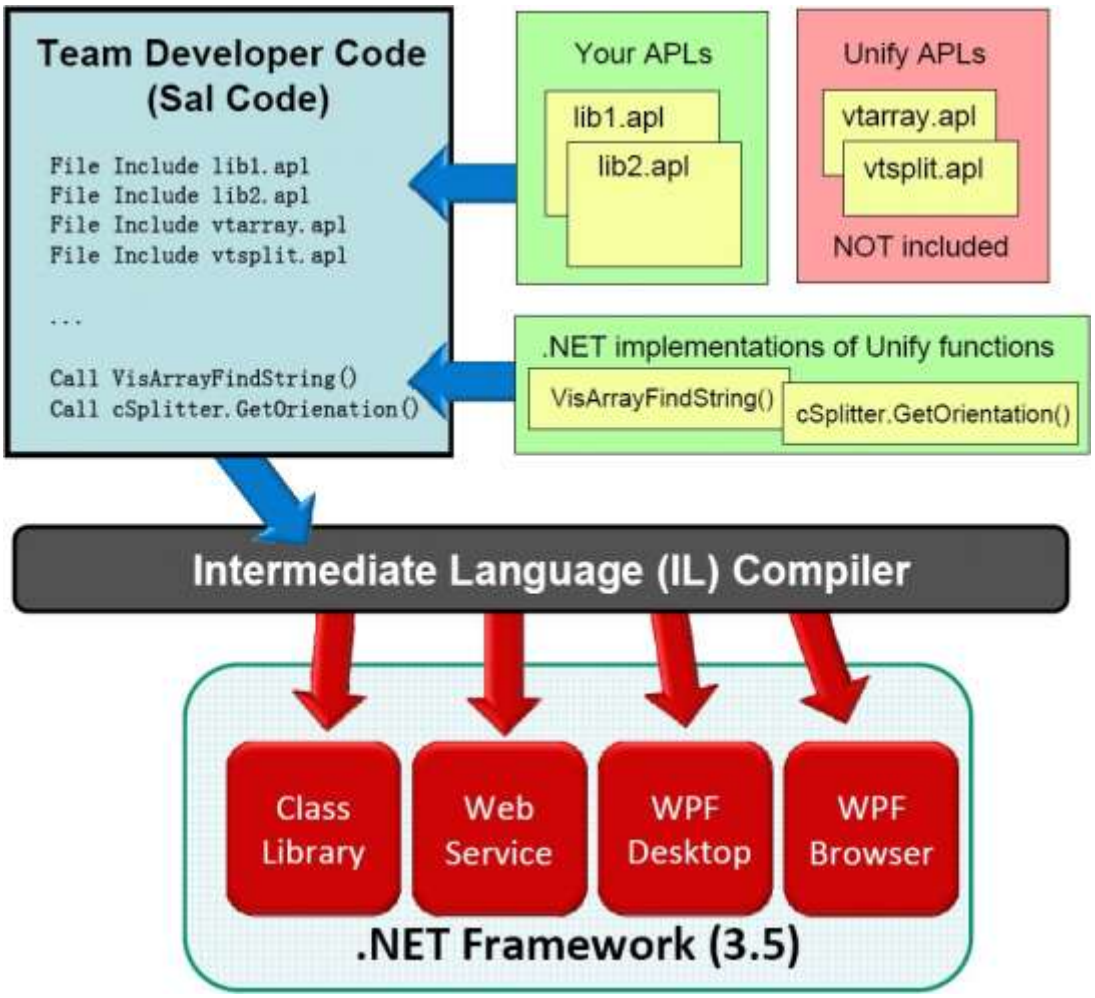
3. **Intermediate Language (IL) Compiler**

   The IL compiler receives your SAL code, your included APLs, and the .NET implementations of Gupta functions you have called in your application. These elements are converted to Intermediate Language and compiled.

4. **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 the process described above:



Compiling from the Command Line

Here are some sample commands for compiling from the command line.

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

  Compiles using the standard TD native compiler (i.e. Win32 application).

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

  Compiles a WPF Desktop application.

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

  Compiles a WPF Browser application.

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

  Compiles a .Net SAL Library.

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

  Compiles a .Net Class Library.

In order 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 table below 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 SAL Library	.Net SAL Library
	All others	.err (error file)

.NET Command Line Switch	Application's Current Build Setting	Result
.net:AutoDeskTop	win 32 EXE, .Net WPF Desktop, .Net WPF Browser	.Net WPF Desktop
	Include Library, .Net Class Library	.Net Class Library
	Dynalib, .Net SAL Library	.Net SAL Library
	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 SAL Library	.Net SAL Library
	All others	.err (error file)

Notes About .NET Projects

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 will cause 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.

Note that there is a size limitation to TD code when being compiled to .NET. This is coming from the Microsoft .NET Assembler, whose memory usage can climb over 2GB if the application is large enough. In TD, it seems to be roughly around 650,000 outline items.

Using the 64 bit version of the assembler (which TD will now do when possible) removes the size limitation.

Also, 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 hit.

GUI Functions in DLLs

.NET Class libraries, .NET Sal Libraries and .NET Web Services does not support GUI objects. Team Developer applications using GUI functions, report warnings when the application is built as .NET Class Library or .NET SAL Library. Team Developer applications using GUI functions, report errors when built as .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. Specifically, this chapter covers:

- QuickObjects
- Visual Toolchest
- External APIs in general
- Unsupported SAL Functions

QuickObjects

With the exception of QuickGraphs (see below), QuickObjects are not currently 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

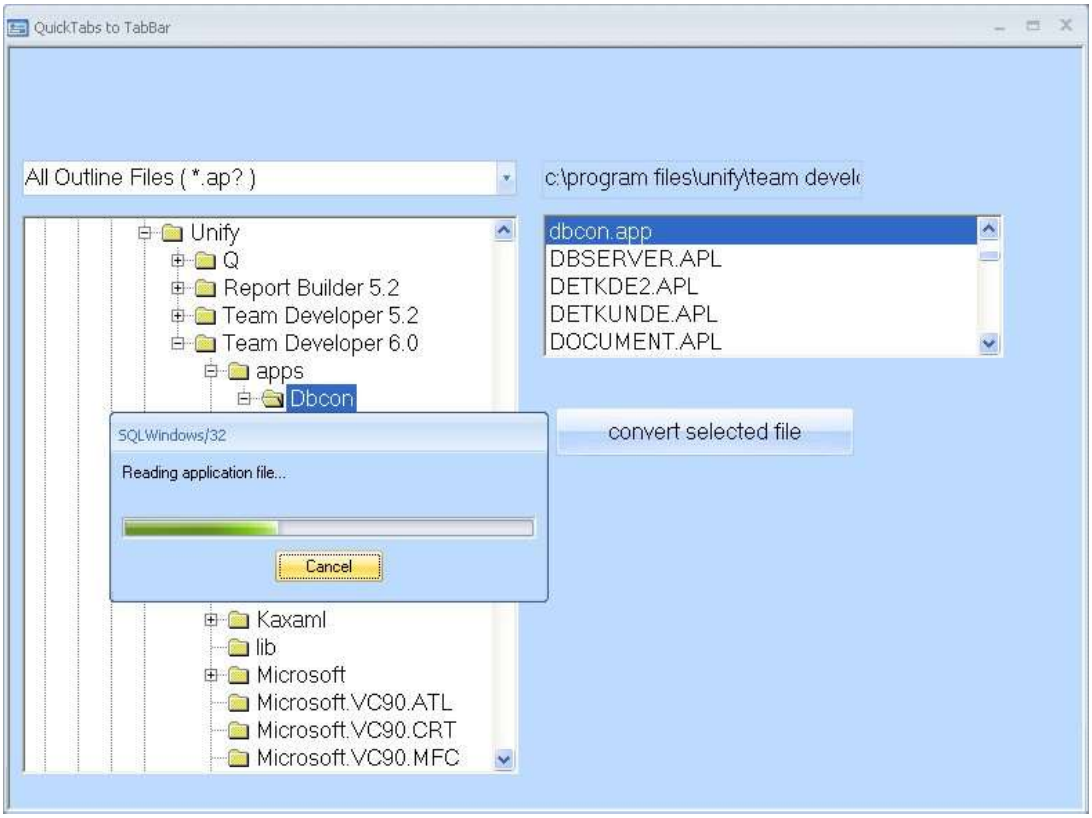
Note: Use this tool to prepare an application for **.NET deployment**. Conversion from QuickTabs to native Tab Controls is not required for Win32 applications.

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, do the following:

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 on 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 filetype.
5. In the list at the right, select the file you want to convert, and click **Convert Selected File**.

Note: Application conversion cannot be done in pieces. If you want to convert an application and its included libraries, convert the **top-level .app file**. QuickTab2Tab will find the necessary libraries and convert them, too.

The QuickTab2Tab tool will read and convert the required files. During the process, you will see progress bars like the one pictured below. A message box will inform 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**.

Note: QuickTab2Tab does not migrate classes derived from cSWTabs or cTabPageList. These will need to be done manually.

Bear in mind that 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 is comprised of several class libraries, as well as many stand- alone functions. Most of the stand-alone functions have been implemented for .NET compatibility, but some have not. Visual Toolchest class libraries are mostly unsupported, with a couple of exceptions.

Note: 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	VisArrayFillNumber	VisArrayFindString
VisArrayCopy	VisArrayFillString	VisArrayInsertItem
VisArrayDeleteItem	VisArrayFindDateTime	VisArraySort
VisArrayFillDateTime	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
VisDosEnumPath
VisDosExist
VisDosGetFlags
VisDosGetNetName
VisDosSplitPath
VisDosSetVolumeLabel

vtfile.apl

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

vtmenu.apl

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

vtmisc.apl

VisGetVersion
VisGetWinFlags
VisGetWinVersion
VisSendMsgString
VisNumberChoose
VisProfileEnumStrings

vtstr.apl

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

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		

VTLibraries

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

Note: Currently no methods supported for vtcal.apl.

---

vtlbx.apl

Base classes: 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/SAL C 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 will be supported, with the exception of HARRAY, HUDV, and STRUCTPOINTER which are **not supported**.

Note: 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 have 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 as such cannot be made to work in .NET.

Assembly Information for APLs

Team Developer contains .NET assembly information for the following APLs. Please note that 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.

Note: You can check if an APL has assembly information by doing the following: Open the APL, right click on its name at the top of the tree view, and select Properties. Then Click on the "Assembly File" tab. If text appears in the field under "Assembly Import File," then Team Developer has assembly information for this APL.

Automation.apl
cgmail.apl
cdk.apl
CStructL.apl
GTableX.apl
ODBSal32.apl
QckDVC.apl
QckMail.apl
QckTabs.apl
SalMail.apl
sqlnwkcn.apl
VTArray.apl
VTCal.apl
VTComDlg.apl
VTComm.apl
VTDebug.apl
VTDOs.apl
VTFile.apl
VTLbx.apl
VTlstvw.apl
VTMenu.apl
vtmeter.apl
VTMisc.apl
vtsplit.apl
VTStr.apl
VTTblWin.apl
VTWin.apl
xmllib.apl

Unsupported SAL Functions

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

SalActiveXAutoErrorMode
SalActiveXCreateFromFile
SalActiveXGetData
SalActiveXInsertObjectDlg
SalAppFind
SalCreateWindowExFromStr
SalDDEAddAtom
SalDDEDeleteAtom
SalDDEExtractCmd
SalDDEExtractOptions
SalDDEFree
SalDDEGetExecuteString
SalDDERequest
SalDDESendAll
SalDDESendToClient
SalDDESetDataText
SalDDEStartServer
SalDDEStopServer
SalDropFilesQueryPoint
SalEditCanPasteSpecial
SalGetTypeEx
SalActiveXCreateFromData
SalActiveXDelete
SalActiveXGetFileName
SalActiveXOLEType
SalContextBreak
SalCreateWindowFromStr
SalDDEAlloc
SalDDEExtract
SalDDEExtractDataText
SalDDEFindAtom
SalDDEGetAtomName
SalDDEPost
SalDDESend
SalDDESendExecute
SalDDESetCmd
SalDDESetOptions
SalDDEStartSession
SalDDEStopSession
SalEditCanPasteLink
SalFontGetSizes
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
SalStaticFirst
SalStaticGetSize
SalStaticSetSize
SalStrToMultiByte
SalStrToWideChar
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
cdk.apl
cdkfwrk.apl
cgmail.apl
gtablex.apl multitbl.apl
pagelist.apl
swbidi32.apl
TlNotify.apl
ttmngr.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 (e.g 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_LBUTTONDBLCLK

Data Field Rich Text Box Combo Box

WM_RBUTTONDBLCLK

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 introduces WPF applications and their associated files. Consult this chapter for information about WPF desktop and browser applications, an explanation of XAML files, and instructions on customizing XAML files.

WPF Applications

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

Some notes about WPF applications:

- Names of WPF Browser applications cannot contain some punctuation characters, such as '**[**' and '**]**'.
- Certain keywords in your application file name will cause a "User Account Control" warning in Windows when you try to run the application. Thus, 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 (these 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.
- Since 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."
- In order to to run TD60 WPF applications using ADOProxy running on 64- bit machine, you need to enable 32-bit Applications mode on IIS:

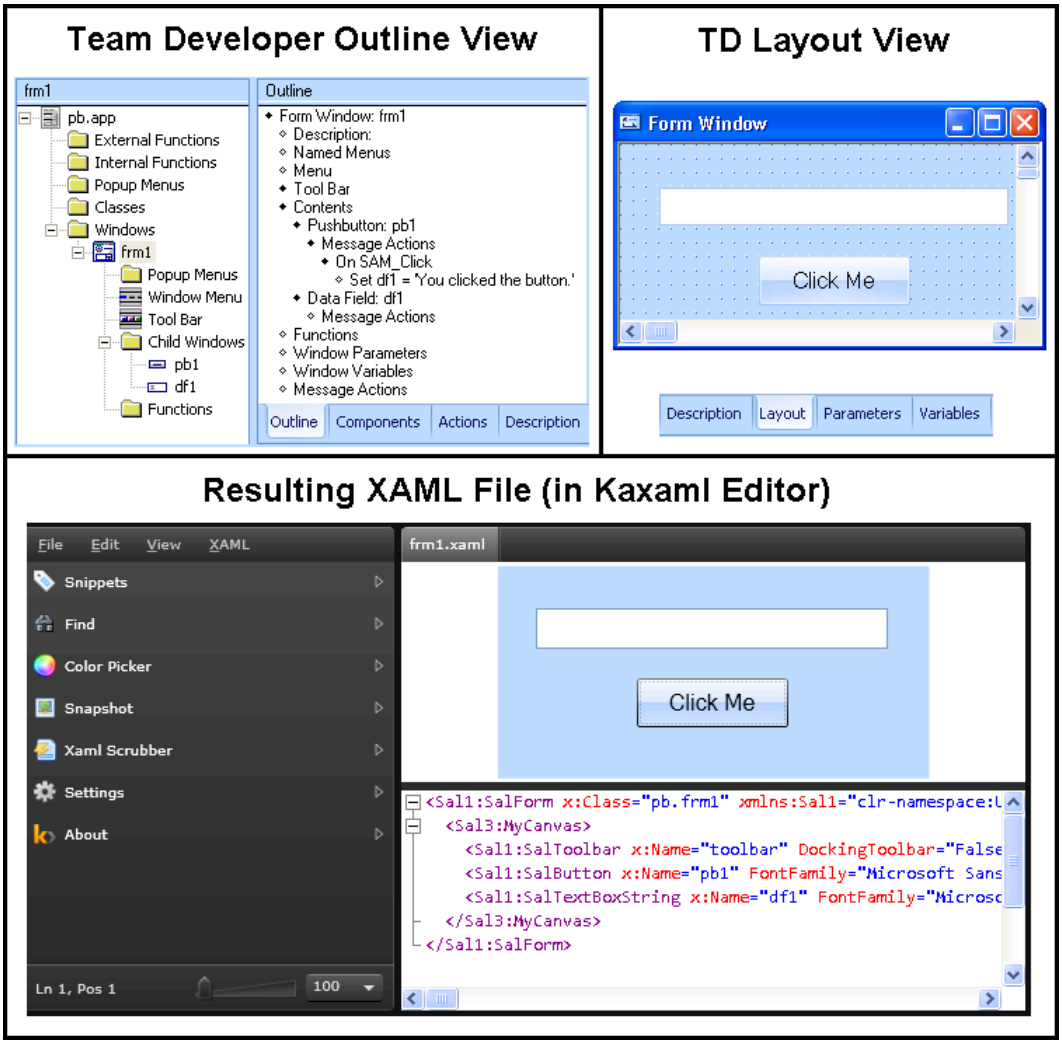
  1. Click on **View Application Pools**
  2. Set Application pool defaults
  3. Set **Enable 32-bit Applications** to TRUE

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 our case, 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, then files are created in your project directory for you to view and edit (see *Customizing XAML files* below). 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 screenshots below 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 screenshot of the XAML file above 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 (just select “Open” from the file menu, find the new .xaml directory, and then select the file you want to view/edit).

Please note the following best practices for customizing XAML files.

XAML Code Customization - Best Practices

1. When customizing XAML files for Team Developer applications, use the XAML editor that is provided with Team Developer.
2. Use the method mentioned above to access custom XAML files (i.e. right-click/ Custom XAML/Edit custom XAML). Running Kaxaml.exe independently or opening .xaml files via Windows Explorer will bypass necessary information exchange between Team Developer and Kaxaml (for example, location of internally-stored application images).
3. If you make significant user interface changes in your Team Developer outline after you have customized your XAML files, update your XAML files using these steps:
    a. Rename your .xaml directory so that the IL compiler will not recognize it.
    b. Compile your project.
    c. Right-click on 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 will use the files in your .xaml directory.
    g. (Optional) Delete the directory you renamed in step a.
8. Finally, do not make the following types of changes in your custom XAML files:
    a. Do not add new controls that do not correspond to controls in your Team Developer application outline.
    b. Do not rearrange the order of controls.
    c. Do not rename controls.
    d. Do not change event handlers.

More Information about XAML

For more information about XAML in WPF, see Microsoft's documentation:

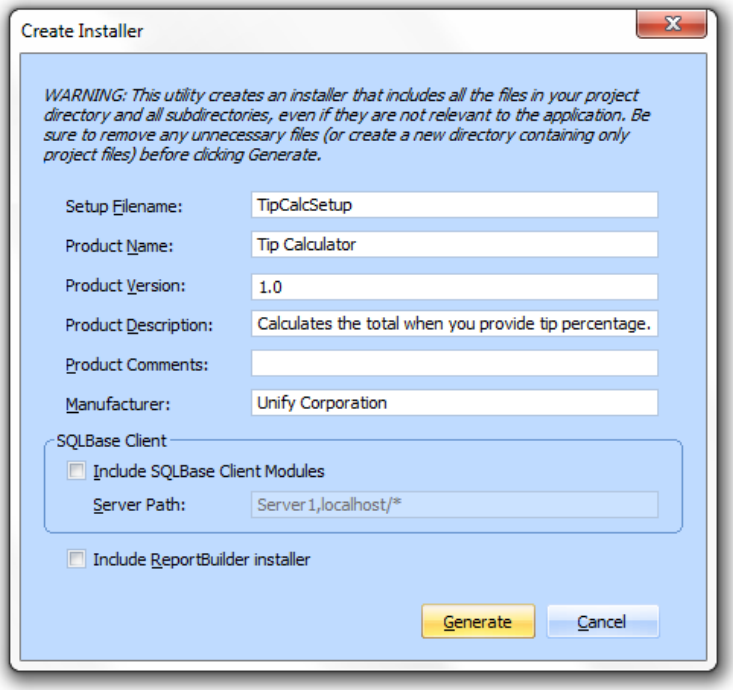
http://msdn.microsoft.com/en-us/library/ms747122.aspx

Deploying WPF Desktop Applications

In order to deploy your WPF desktop applications to users, you will need to create an installer by doing the following:

1. Select "Create Installer" from the Project menu. Note that this menu option will be disabled unless your build target is set to WPF Desktop.

   The Create Installer dialog box displays.



   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" checkbox. Otherwise, leave this box unchecked.

Note: Your project's sql.ini file will be packaged in the installer. Thus, if your sql.ini uses "localhost" in its serverpath setting, you will need to change this setting to point to the SQLBase database you plan to use (e.g. use an IP address).

3. If your application requires Report Builder functionality, check the "Include ReportBuilder installer" checkbox. This will allow your users to install a limited version of Report Builder. They will be able to access all the functionality required for your application, but they will not be able to create new reports or run Report Builder independently. If your application does not require Report Builder functionality, leave this checkbox unchecked.
4. Click Generate, and wait a few moments. You will see a "Generate Finished" message, and the installer will be placed in your application directory.

WPF Browser Applications in Firefox

In order to use the Firefox browser to run WPF Browser applications, you need to install a Firefox plugin. To check if you have the plugin, use the script here:

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 above link again. If the script still indicates a problem, check this 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** and you should be set.

Windows 7 Users

The .NET 3.5 SP1 framework cannot be installed on a Windows 7 machine. In order to obtain the necessary plugin, you will need to 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 in order 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)

You can publish a WPF Browser application by following the steps outlined below. A successfully published application will have a starting/index page like the one pictured here.

To publish your application, do the following:

1. Build settings for WPF Browser application

Note: If you are running Team Developer in Windows Vista or later, be sure to **Run as Administrator** so that Team Developer will have 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 xbap_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 which is used to sign the application.

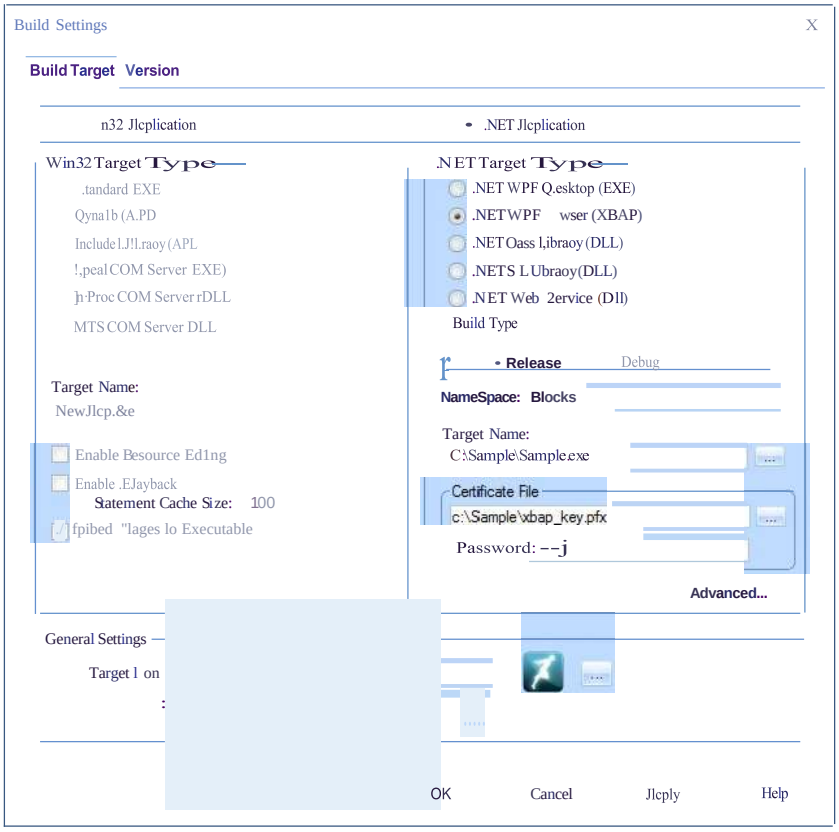
1. Open the Build Settings dialog by selecting "Project" and "Build Settings" from the menu bar. Select ".Net Application". Select ".NET WPF Browser (XBAP)". Under "Target Name:" enter the directory where you want to build the application. Click "OK."

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

2. Build the application by selecting "Project" and then "Build:" from the menu bar.
3. Open your application directory in explorer. Open the “xbap_key.pfx” file; configure the private certificate with the following steps:
    a. On the initial screen click “Next”
    b. Click “Next” again
    c. The password will be ‘12345’ (see note below). 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 the above steps and selecting “Trusted Publishers” in step e.

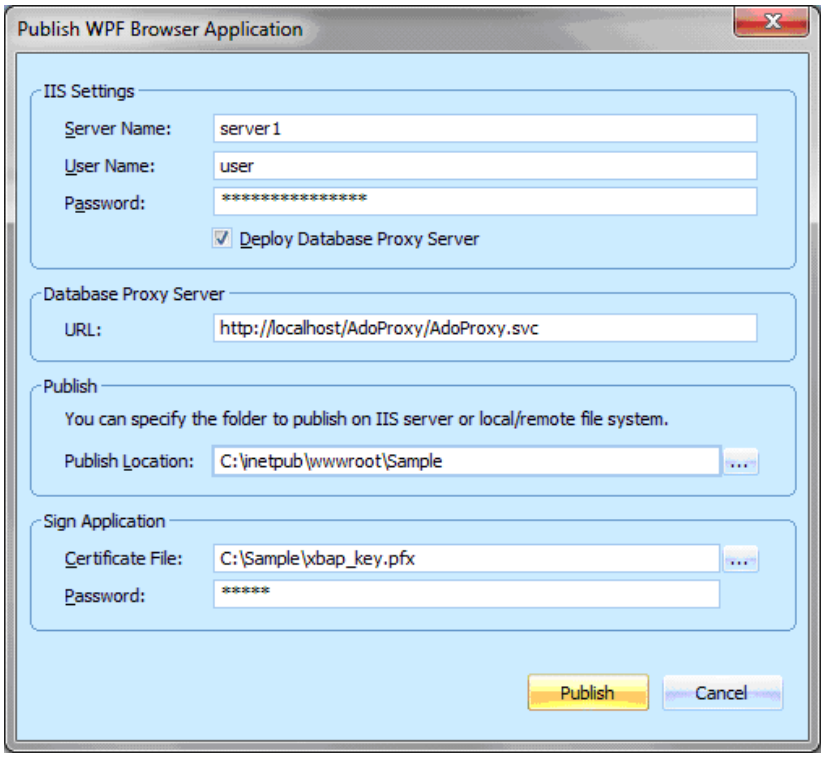
Note: The private certificate provided with Team Developer has a set password of 12345. If you use a different certificate, you must also use the password of that certificate.

4. **Go back to the Build Setting dialog. Enter your certificate file's location and password. Then click OK.**



2. 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" then "Publish…" from the menu bar.

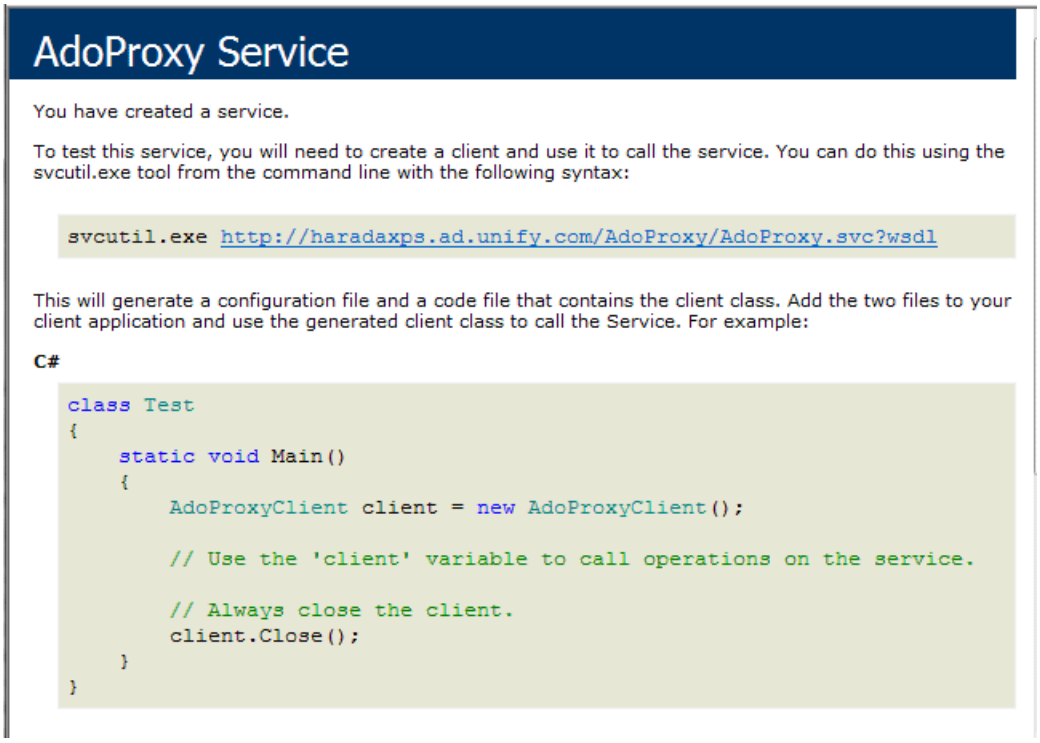


Here is an explanation of each field in this dialog:

Field	Explanation
Server Name	IIS server name to deploy WPF application (IIS 6, IIS 7)
User Name	User name who can log into specified IIS server. This user must be a member of the Administrators group.
Password	Password for the above user.
Deploy Database Proxy Server	Check this If you want to deploy database proxy server to the same IIS server along with the application. The URL of the proxy service will be http://<IIS_Server>/DBPipeServer/DBPipeServer.svc Note on using SQLBase through a remote Server: Prior to publishing the XBAP application, edit sql.ini, under "[win32client.ws32]" section, by commenting out "autostartserverpath".

Field	Explanation
URL (Database Proxy Server)	Specify the database proxy service URL which 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 will be deployed to <IIS Root>/<Application Name>. In most cases <IIS Root> is C:\InetPub\wwwroot. You can also specify a local or remote folder when you don't specify an IIS server. This is useful when the WPF application has been deployed and you know its physical location.
Sign Application	Specify your own certificate here. Changes made in this field will be reflected in the Build Settings dialog.

If the proxy server is running, the following page will display when accessing the specified URL:



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 doing the following:

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

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

Solution: Because the default IIS configuration does not enable WCF, you will need to run the following command as administrator in order to change this configuration:

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

3. Database Proxy Service

Database Proxy Server is a webservice 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

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.)
   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 wish to install and finish installation

f. Edit sql.ini (on IIS Server). Under "[win32client.ws32]" section, change "localhost" to Server address where SQLBase resides

3. Create a SQLBase System Variable (Windows 7/Vista)

a. Right Click on Computer. 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

a. Repeat steps a. thru c. (Step 3)

b. Under System variables, Select Path. Select Edit.

c. Under Variable value, enter "%SQLBase%;" to the front of the Path. Select OK.

d. Restart your IIS Server.

Oracle Client Setup on IIS Server

1. Install oracle 11gR2 32bit client or server on IIS Server (database proxy server).
2. Configure Oracle connection(s).

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 document entitled *Connecting SQLWindows Objects to Databases*.

1. Launch odbcad32 on Proxy Server machine.

Note: On a 64 bit Windows OS, you will need to go to **C:\Windows\SysWOW64** and then launch **odbcad32**.

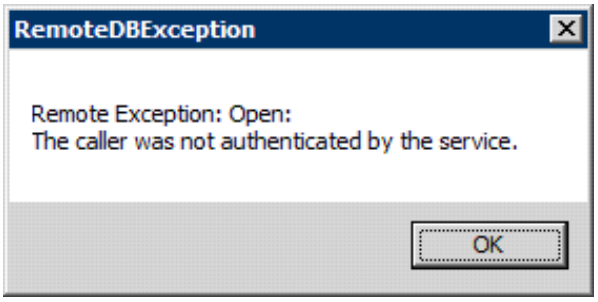
---

2. Add system DSN(s) needed by the application.

Notes About Proxy Server

Keep the following in mind as you set up the proxy server:

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



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

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

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

Chapter 5. WPF Controls

This chapter introduces WPF controls. Read this chapter for information on how to implement WPF controls in your applications, change their properties, invoke their methods, etc.

Why WPF Controls?

In many cases, WPF controls will serve as replacements for visual ActiveX controls. Most ActiveX controls are old implementations of controls like gauges, HTML viewers, etc. These controls usually have an old fashioned look and feel, and many third party WPF controls 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," 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 may be a container, which 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 have been provided for retrieving and setting the values of properties in a WPF control:

SalWPFGetBoolProperty	SalWPFSetBoolProperty
SalWPFGetDateProperty	SalWPFSetDateProperty
SalWPFGetNumericProperty	SalWPFSetNumericProperty
SalWPFGetStrProperty	SalWPFSetStrProperty

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

See the in-build help for documentation on each of these functions.

WPF Events

The **SAM_WPFEvent** message will fire 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 has 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.

Here 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.

Properties for gauges are listed below. 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

Name	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 the MajorTicks is set to "4", there will be 4 major groups inside that range of 0-100. That would result in tick marks at the 25, 50, and 75 on the scale. In most gauges, the major tick marks are the locations where an actual numeric value is drawn on the scale.
MiddleTicks	Double	Each major tick section can be subdivided into "Middle Ticks" - the MiddleTicks property will specify how many subgroups will be displayed inside each major tick group. Continuing the above example, a MiddleTicks setting of "2" would result in 1 medium-sized tick mark being drawn between each major-sized tick mark (that one medium tick dividing the 2 medium tick groups).
MinorTicks	Double	Similar to MiddleTicks - the MinorTicks property indicates how many subgroups will be 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

Radial Gauges, Range Gauges, and Flat Range Gauges

These properties are in addition to those listed in the “Properties for all Gauges” table.

Name	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

Range Gauges and Flat Range Gauges

These properties are in addition to those listed in the above table and the “Properties for all Gauges” table.

Name	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:

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

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.

Samples

For examples of how to implement WPF controls in your applications, change their properties, invoke their methods, etc., 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

1. When developing WPF applications which go against an oracle database you must use the Oracle 11gR2 32bit client.
2. The Oracle11gR2 32bit client can be used to go against an oracle10g or oracle11g database.
3. There is no need to configure an oracle .NET connection as that Team Developer will automatically map your native and/or OLEDB connections to equivalent .NET connection(s).

How to configure an oracle OLEDB UDL file:

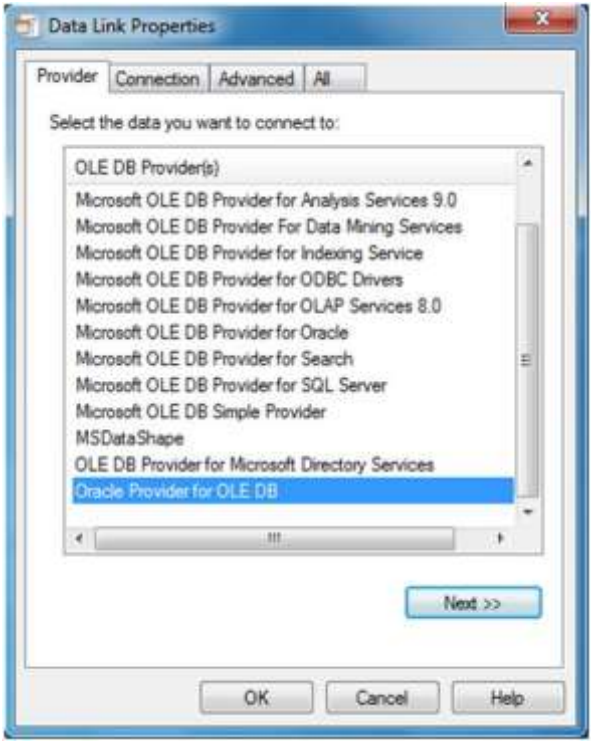
1. Open the command prompt and register the Oracle 11gR2 32bit OLEDB driver:

```
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:

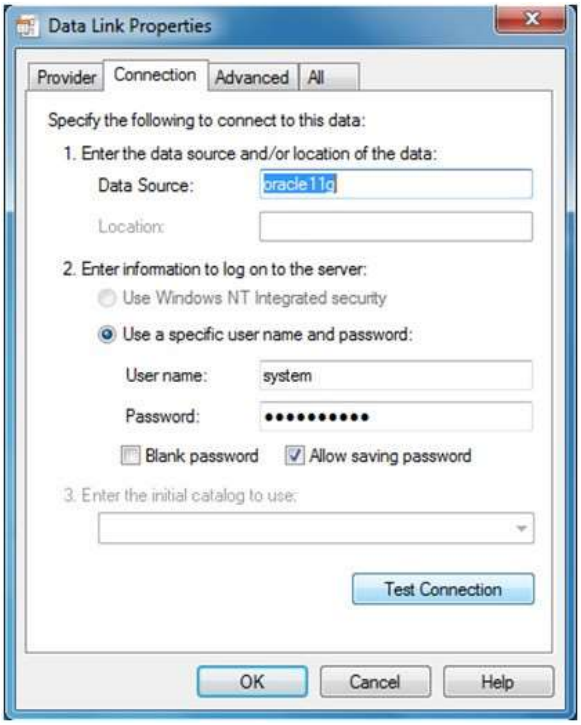
```
rundll32.exe "%ProgramFiles(x86)%\Common Files\System\OLEDB\oledb32.dll", OpenDSLFile
   "<path to udl file>\<udlfile>.udl"
```

3. Under the “Providers” tab select “Oracle Provider for OLE DB”



4. In the “Connection” tab enter your Data Source name along with the username and password.

5. Do a test connection



Note: 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 connection(s).

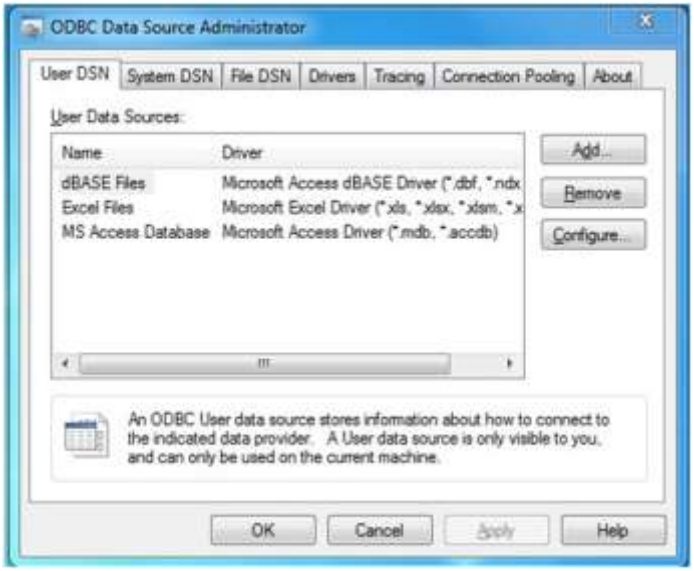
ODBC:

For an odbc connection setup like you have always done.

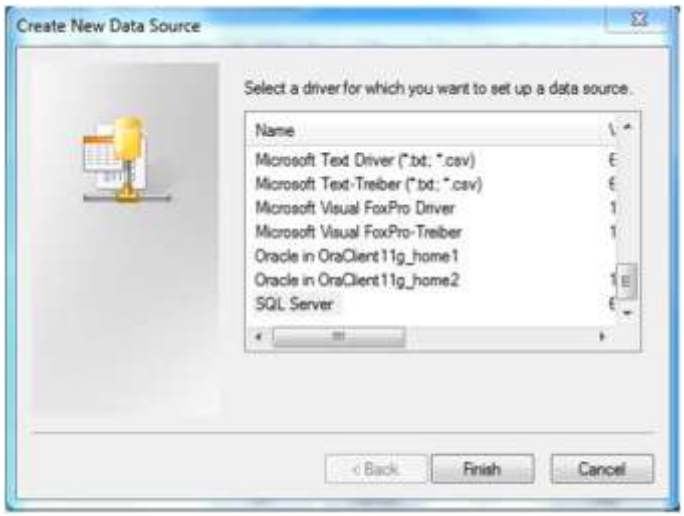
1. Open the command prompt and run “odbcad32

Note: On a 64bit windows OS run odbcad32 from the syswow64 directory.

2. In the ODBC Data Source Administrator window select the “Add” button.



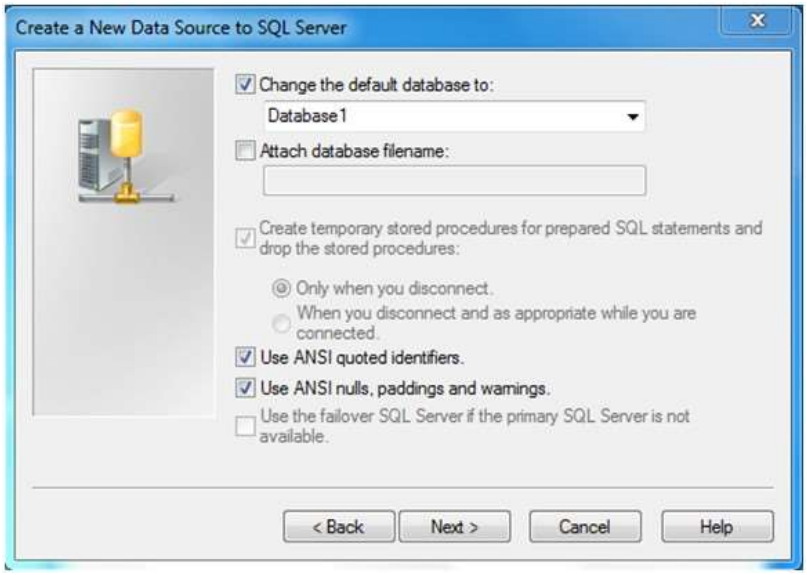
3. In the resulting Create New Data Source dialog scroll down to “SQL Server” for standard odbc connection or “Native SQLServer” for native odbc connection and select “Finish.”



4. In the Create New Data Source to SQL Server Dialog enter a reference name for the server, a description, and select the server you wish to connect to. Then select "Next."

5. Select "With SQL Server authentication using a login ID and password entered by the user". Enter in the Login ID and Password and select "Next."

6. Check the “Change the default database to” checkbox and select your database from the dropdown; select “Next.”



7. Select “Finish”
8. Select “Test Data Source…” and if the test is successful select “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 “[odbcrtr]” using the reference name you gave your server: remotedbname=[RefName],dns=[RefName]

    (e.g. remotedbname=[SVR1],dns=[SVR1])

Note about SQL Server TIMESTAMP

Due to strict Data Type Matching inWPF applications, SqlServer TIMESTAMPS (which are often used as Row IDs) are handled somewhat differently than they are in WIN32. Since TIMESTAMPs are binary data types, to bind a timestamp to a string you need to 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)
```

Note that the resulting string will be a string of bytes packed into a double byte Unicode string, so it will not be human readable, but it 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 the .NET explorer, as well as the assemblies you can access and libraries you can build with it.

.Net Explorer

The .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**.

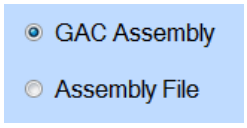
How to use the .Net Explorer

To use the .Net explorer:

1. In the **Tools** menu, select **.Net Explorer**. You will see the following:

Click "Next."

2. The next screen allows you to select "GAC Assembly" or "Assembly File."

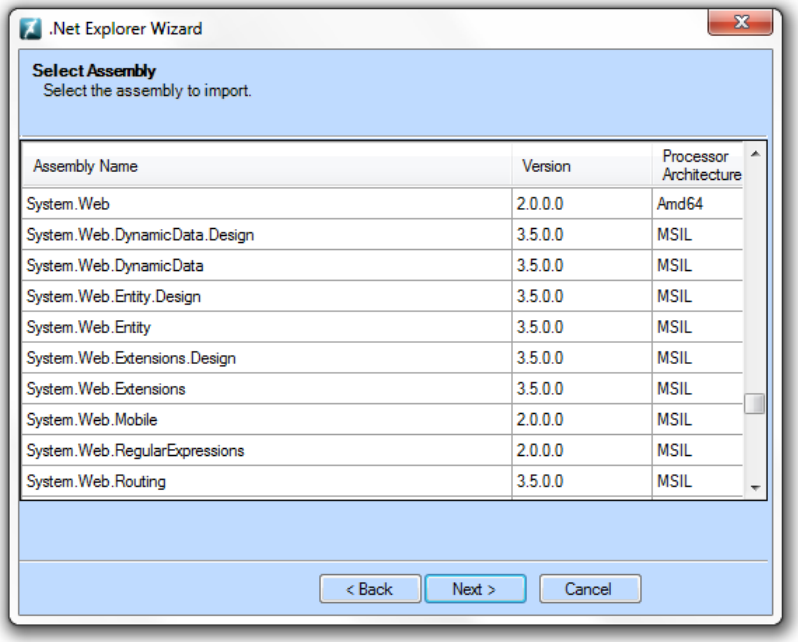


- GAC Assembly - Choose this option to import an assembly from the Global Assembly Cache (a collection of .Net assemblies that exist on your machine).

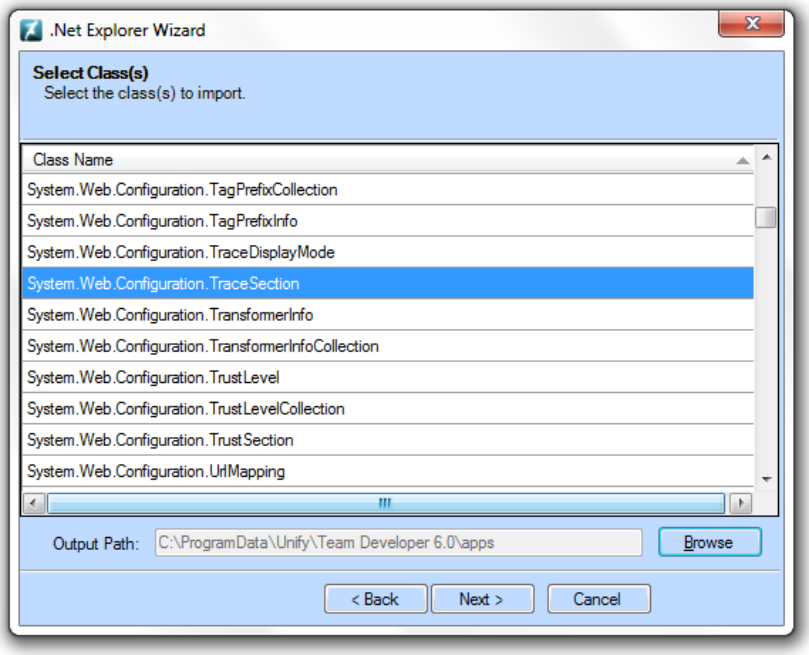
- Assembly File - Choose this option to import a custom assembly. Click "Next."

3. The next screen will be different depending on which option you selected in step 2. If you chose...
   - ...GAC Assembly, you will see something like the screenshot below.
   - ...Assembly File, you will see a data field and "Browse" button where you can indicate the location of the assembly file.

Note: Steps 4 and 5 assume that you chose GAC Assembly. See steps 6 and 7 for information that applies to either option.

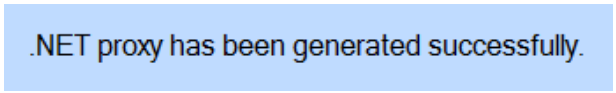


4. Find the assembly you would like to import, and select it. Then click "Next."
5. The next screen lists classes contained in the assembly you selected. Select the class(es) you would like to import. Use Shift+click, Ctrl+click, or click and drag to select multiple classes.



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

Then click "Next."



6. When you see the above message, click "Finish."

   The .Net explorer will close, and assuming Team Developer is still the most recently active application, you will return to your application outline. Note that **the generated APL is not automatically included in your outline**. Thus, you can repeat steps 1-6 to generate as many APLs as you would like.

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, the .NET Explorer will display the following error message when you try to import the DLL: "Could not load the selected assembly because its CPU architecture is not supported."

---

7. To include the generated APL, right click on the Libraries section of your outline, select "Add Next Level," and then select "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" checkbox. Then click "Open."

About AXLs, APLs, DLLs

Keep the following in mind as you work with the .Net explorer and AXL, APL, and DLL file types.

Definitions

For the purposes of this chapter, AXL, DLL, and APL files can be defined as follows:

AXL - An AXL file is an 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

Include the APL

When you import a .Net assembly and create an APL via the .Net explorer, an AXL file of the same name is also generated. As mentioned above (*How to use the .Net Explorer*, step 7), 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, and the APL is only needed if your finished product will be a Win32 application. Both files are included in the outline for two reasons:

1. With both files included, you can compile your application to .NET or Win32 without changing your file inclusions.
2. 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

Include the 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 will look in the same directory for the DLL and automatically include 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 the .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.

If a user wants to resize a native .NET array, they need to do it 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 you should choose 32-bit as target CPU type for debugging purposes.

.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 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

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

The following steps describes on how to use and debug .NET Class Libraries in Team Developer .NET applications.

Step 1: Build the Class Library using Visual Studio. Make sure to generate a PDB file for the library.

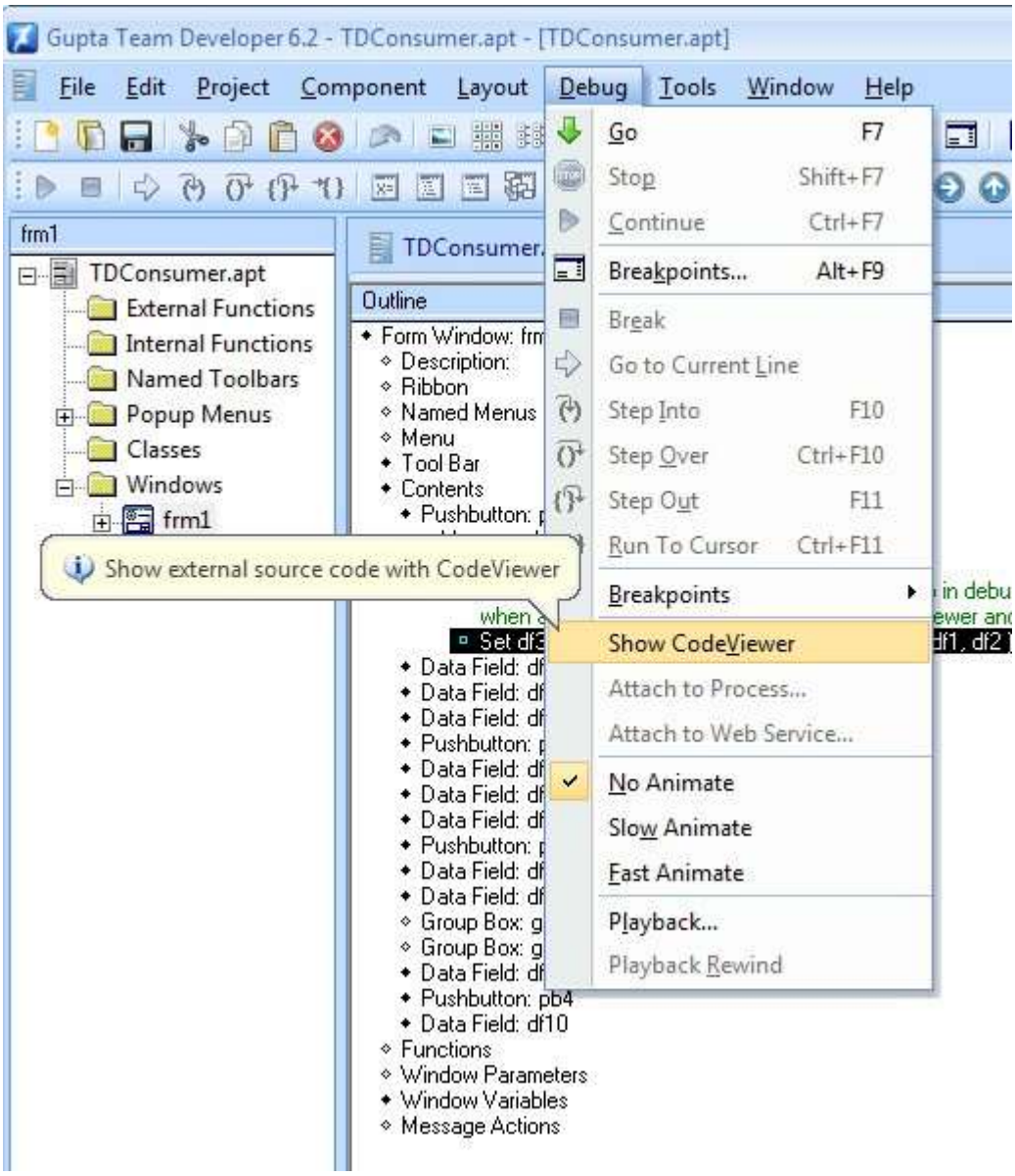
Step 2: Use .NET Explorer and import the Class Library that is generated in Step1 to create corresponding APL and AXL files.

Step 3: Open Team Developer application that is using the library and set the breakpoint at the function call. Make sure build settings for the application is set to either .NET WPF Desktop or .NET WPF Browser.

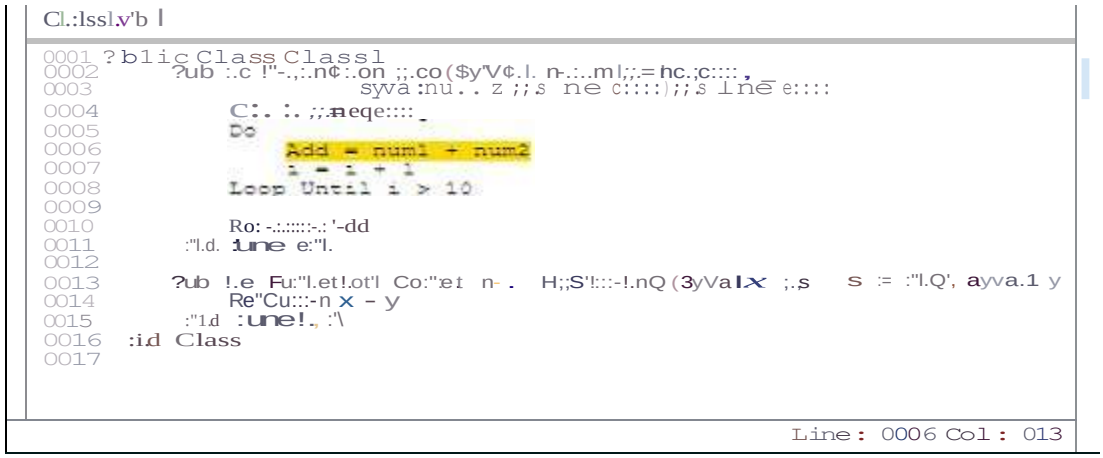
Step 4: Press F7 from TD IDE and run the application in Debug mode. When the breakpoint is hit, click on Step-In and observe that SQLWindows opens the corresponding library source file in a separate code-viewer.

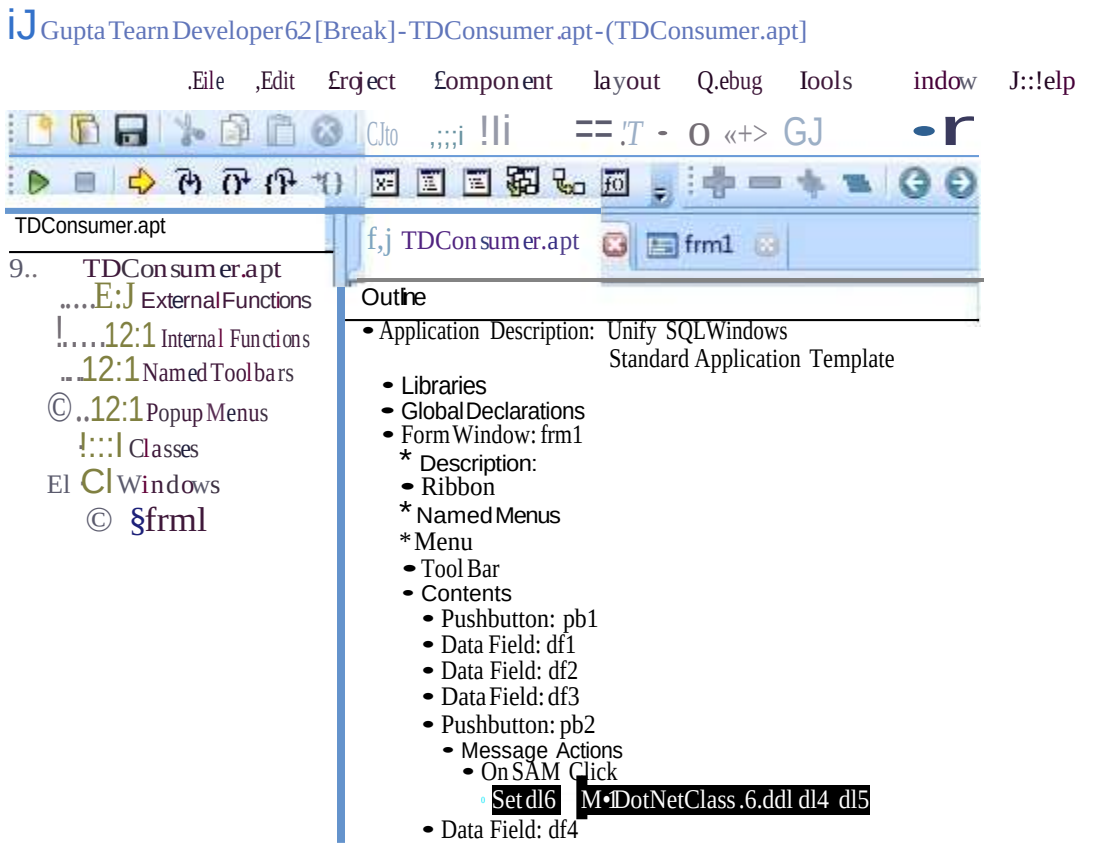
Step 5: Keep hitting Step-In and observe the focus in the separate code viewer moves as it executes the code functions.

***Note** : Alternatively, the user can open a separate code viewer by selecting “Show CodeViewer” menu item under popup menu “Debug” in the IDE as shown below.



The following picture illustrates when SQLWindows steps into other sources.


Cl.:lssl.v'b I
0001 ?blic Class Classl
0002 ?ub :.c !"-.,:.n¢:.on ;;.co($y'V¢.l. n-.:..ml;;.= hc.;c:::: ,
0003 syva:nu.. z ;;s ne c::::);;s Ine e::::
0004 C:. :. ;;.aeqe:::: _
0005 Do
0006 Add = num1 + num2
0007 i = i + 1
0008 Loop Until i > 10
0009
0010 Ro: -.:.:::::-.: '-dd
0011 :"l.d. tune e:"l.
0012
0013 ?ub !.e Fu:"l.et!.ot'l Co:"et n- . H;;S'!:::-!.nQ (3yVa!x ;.s s := :"l.Q', ayva.1 y
0014 Re"Cu:::-n x - y
0015 :"1d :une!., :'\
0016 :id Class
0017
Line: 0006 Col: 013


iJ Gupta Tearn Developer62 [Break] - TDConsumer.apt - (TDConsumer.apt]
.Eile ,Edit £roject £omponent layout Q.ebug Iools indow J::!elp
TDConsumer.apt
9.. TDConsumer.apt
....E:J External Functions
!.....12:1 Internal Functions
..12:1 Named Toolbars
©..12:1 Popup Menus
!:::I Classes
El Cl Windows
© §frml
f,j TDConsumer.apt
frm1
Outline
• Application Description: Unify SQLWindows
Standard Application Template
• Libraries
• Global Declarations
• Form Window: frm1
* Description:
• Ribbon
* Named Menus
* Menu
• Tool Bar
• Contents
• Pushbutton: pb1
• Data Field: df1
• Data Field: df2
• Data Field: df3
• Pushbutton: pb2
• Message Actions
• On SAM Click
Set dl6 M•1DotNetClass.6.ddl dl4 dl5
• Data Field: df4

.NET Web Services

SQLWindows allows debugging of .NET Webservices 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 Webservices with Team Developer.

The following steps describes how to debug Web Services hosted on a local machine.

Step1: Create a Web Service dll and corresponding asmx file using SQLWindow's .NET WebServices build setting (DataTypesTest.apt). **Step 2:** Host the

generated Web Service onto IIS **Step 3:** Open Team Developer as

Administrator.

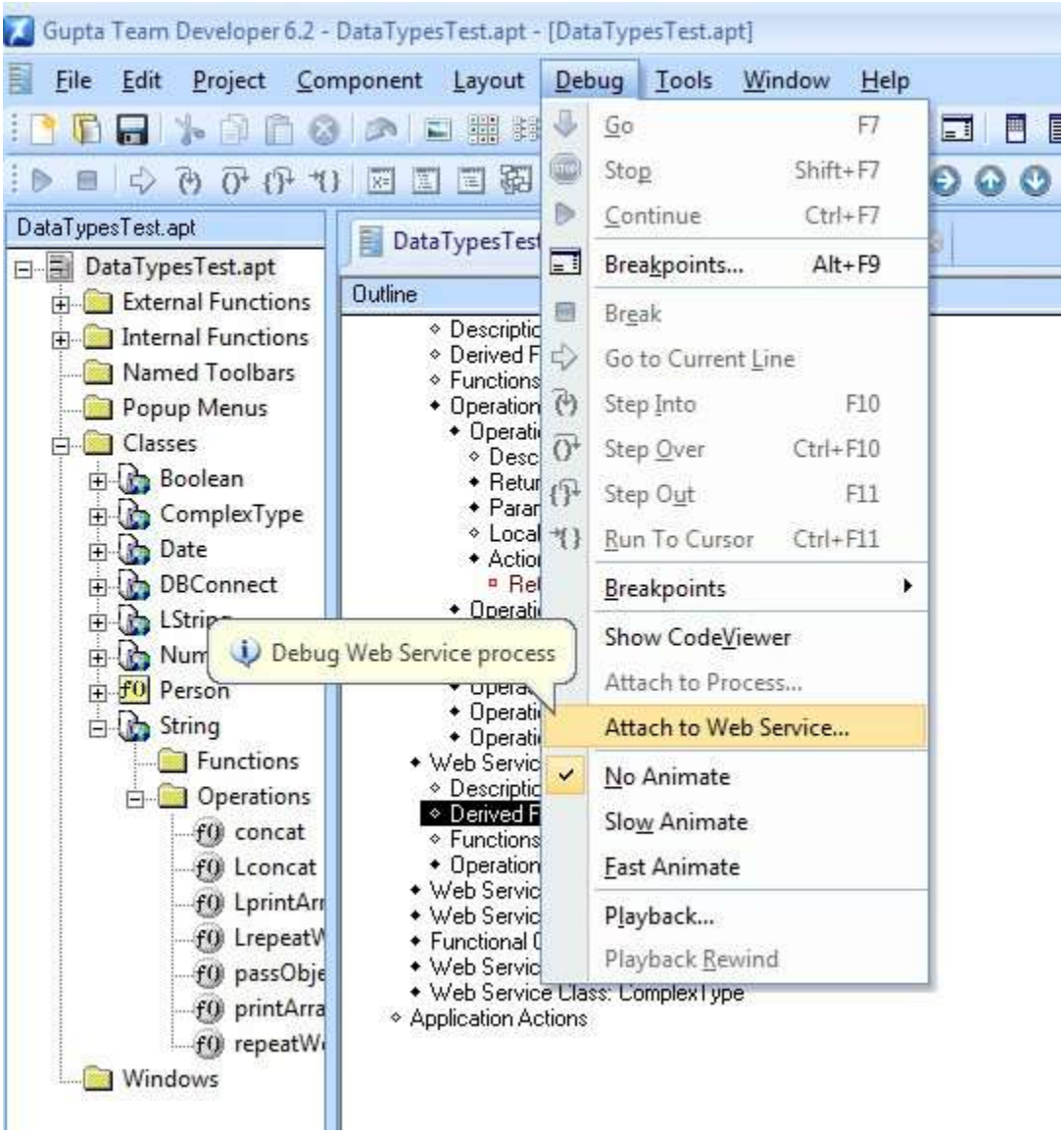
Step 4: Open the web service source, DataTypesTest.apt, from the IDE

Step 5: Select popup menu "Debug. Select "Attach to web service" menu item. This command attaches the current code to the web service process (w3wp).

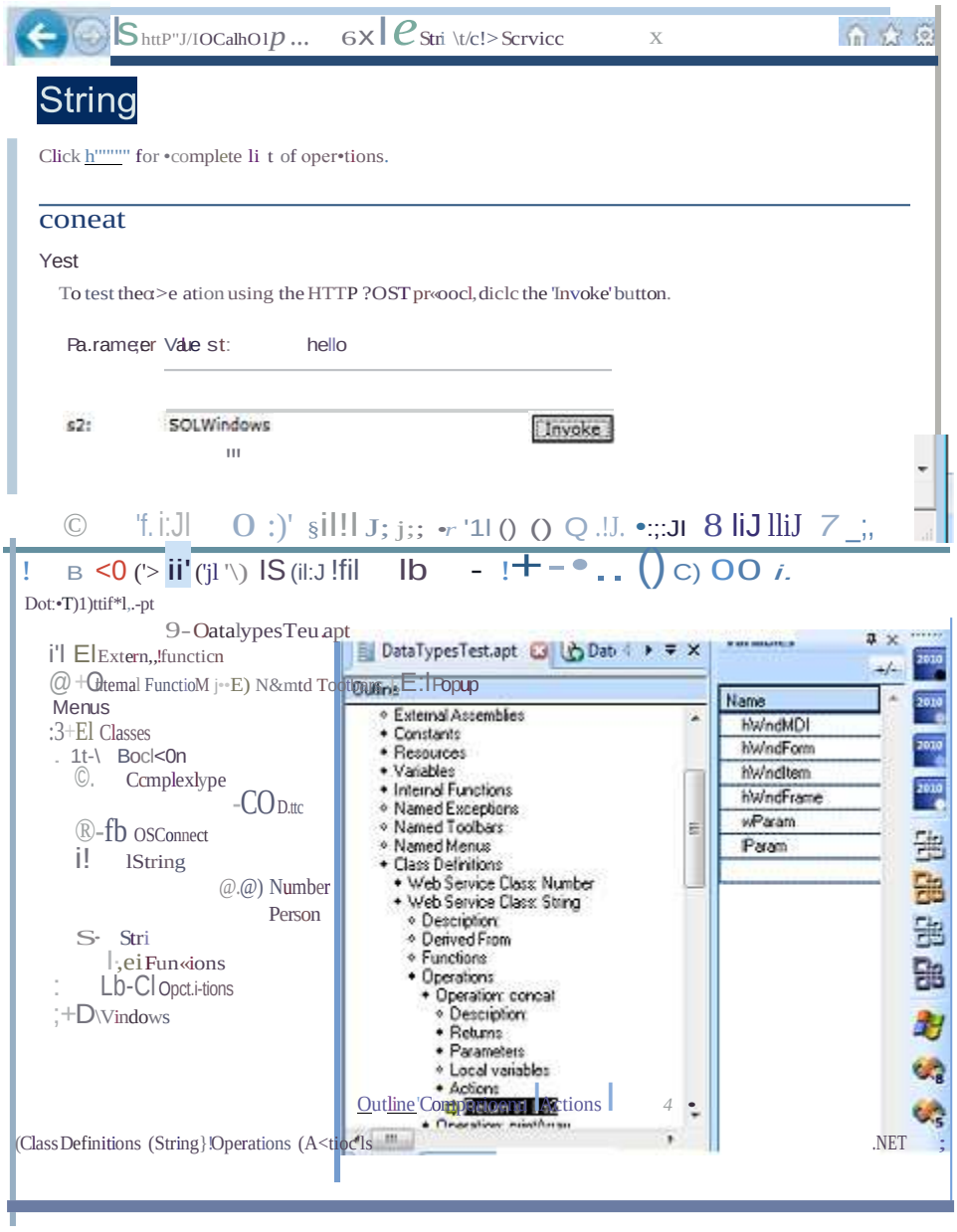
Step 6: Access the web service from Internet Explorer using corresponding asmx file and observe that Team Developer steps into the current code.

***Note :** Currently, the new debugger can attach/debug only in 32-bit since Team Developer is a 32-bit application. SQLWindows allows customers to build .NET applications or libraries as 64-bit but you should choose 32-bit as target CPU type for debugging purposes.

Please refer to the following illustration.



The following is a Web Service debug illustration.


String
Click h""""" for •complete li t of oper•tions.
coneat
Yest
To test theα>e ation using the HTTP ?OST pr«oocl, diclc the 'Invoke' button.
Pa.rame,er Vaue st:
hello
s2:
SOLWindows
Invoke
Menus
Classes
Person
Outline
Name
hWndMDI
hWndForm
hWndItem
hWndFrame
wParam
lParam
External Assemblies
Constants
Resources
Variables
Internal Functions
Named Exceptions
Named Toolbars
Named Menus
Class Definitions
Web Service Class: Number
Web Service Class: String
Description
Derived From
Functions
Operations
Operation: concat
Description
Returns
Parameters
Local variables
Actions
.NET

.NET SAL Libraries

.NET SAL Libraries are equivalent to Dynamic Libraries (.APD) in WIN32. In order to debug a SAL library, users need a consumer application which includes .NET SAL Libraries.

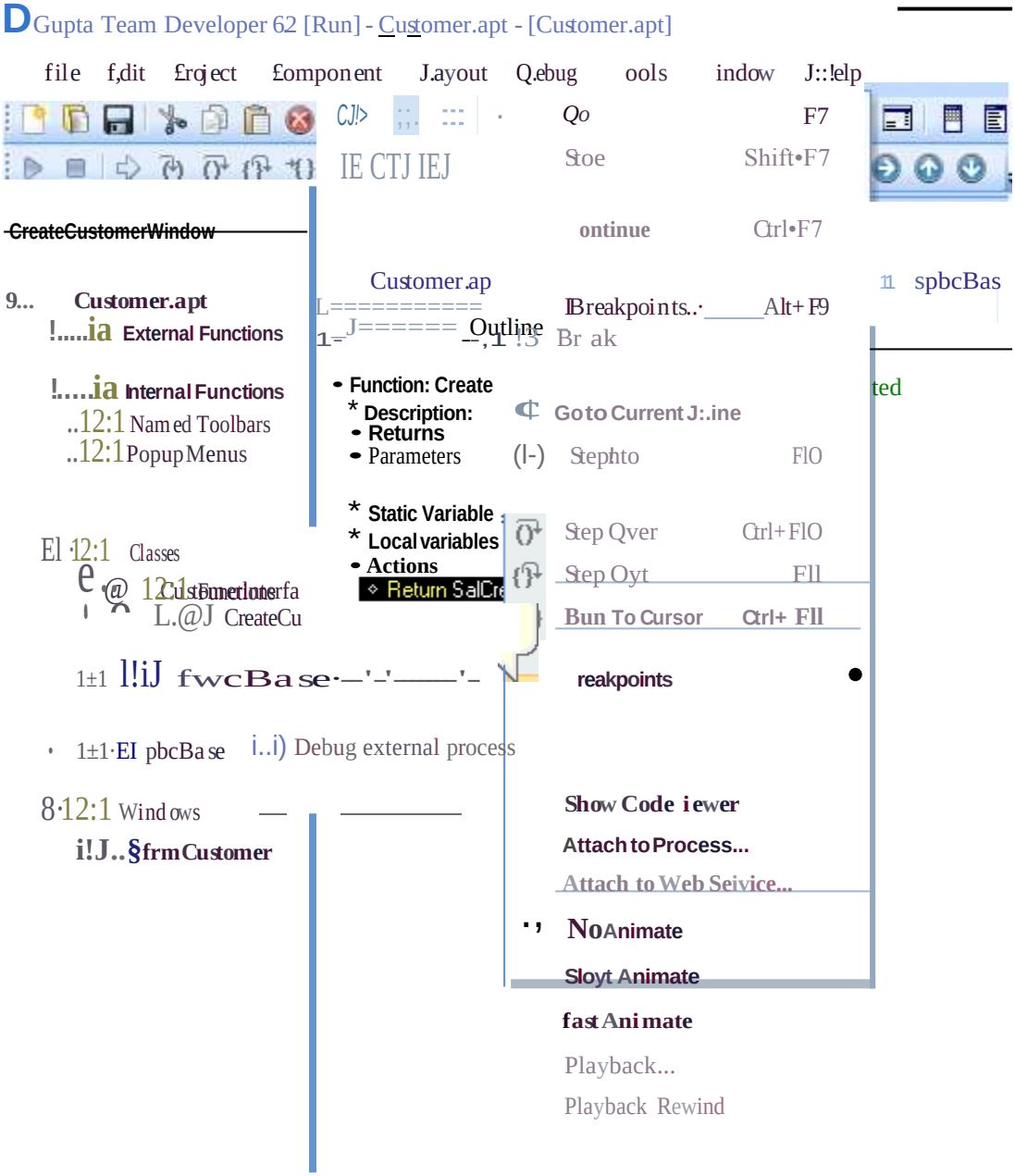
Step 1: Run the consumer application (WPF EXE).

Step 2: Open .NET SAL Library source in Team Developer IDE as administrator.

Step 3: From Team Developer , select popup menu “Debug”. Select menu item “Attach to Process”. Select the consumer application EXE.

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

The following screenshot illustrates the menu item ”Attach to the process”.


Gupta Team Developer 62 [Run] - Customer.apt - [Customer.apt]
file f,dit £rqject £omponent J.ayout Q.ebug ools indow J::!elp
F7
Shift•F7
ontinue Ctrl•F7
CreateCustomerWindow
Customer.ap
Outline
Breakpoints.. Alt+F9
Br ak
spbcBas
9... Customer.apt
External Functions
Internal Functions
Named Toolbars
Popup Menus
Classes
Function: Create
Description:
Returns
Parameters
Static Variable
Local variables
Actions
Return SalCre
Go to Current J:.ine
Step hto FlO
Step Qver Ctrl+FlO
Step Oyt Fll
Bun To Cursor Ctrl+ Fll
reakpoints
Debug external process
Windows
frmCustomer
Show Code iewer
Attach to Process...
Attach to Web Seivice...
No Animate
Sloyt Animate
fast Animate
Playback...
Playback Rewind