OpenSpan Core Training

OpenSpan Studio Basics:

OpenSpan, Inc.
4501 North Point Parkway
Suite 140
Alpharetta, Georgia 30022
www.openspan.com

INTRODUCTION

This course is intended for new OpenSpan Studio users. The course provides instruction on the main functional areas of the OpenSpan Studio and OpenSpan Runtime applications. OpenSpan Studio runs within the Microsoft Visual Studio 2008 Isolated Shell design environment or as a plug-in to the Visual Studio 2008 standard or greater application. This course focuses on using OpenSpan Studio to create application integration and automation projects. For information on the Visual Studio framework, see the Microsoft MSDN website.

Prerequisites

The course requires the following:

- OpenSpan Studio 4.5
- Sample CRM application (installation file provided in OpenSpan Studio installation Extras folder)
- Training project files available for download from the OpenSpan Community website Learning and Certification page (download and then extract Zip file to a folder location of your choice)
  Note: If you are using the OpenSpan Studio plug-in, change the extensions of the training solution files from .ossln to .sln.
- Internet Explorer versions 6, 7 or 8
- Access to the OpenSpan training website: (http://training.openspan.com/index.html)

Conventions

This document uses the following typographical rules and conventions:

Convention	Meaning
Black bold characters	**Names of program elements that require emphasis, such as command buttons, menus, and dialog boxes, are shown in black bold text.**
Blue Bold Characters	**Text that you are supposed to type, such as in a text box, appear in blue boldface characters.**
Remember	**Definitions of terms and important concepts that bear remembering.**
	**Next to the Tip icon, you can find best practices and shortcuts to use OpenSpan Studio more effectively.**

CHAPTER 1: INTRODUCTION TO OPENSPAN STUDIO

The OpenSpan Platform fully enables rapid integration and automation between desktop applications, without programming, and at a fraction of the cost and time of any other solution.

OpenSpan Studio creates an interface for every application on the desktop, so complete desktop control and inter-operability can be designed with applications from virtually any platform or origin. The OpenSpan Studio designer focuses entirely on the desktop interface, and creates solutions by interrogating and optimizing application functions and interactions.

OpenSpan solutions do not require additional infrastructure or costly modifications. There are no special programmers to hire and no new programs for the user to master. Our integration technology not only interfaces with most platforms seamlessly, it has also proven itself to be one of the most reliable methods of application integration in the industry.

OpenSpan enables you to:

- Automate time-intensive manual tasks with ease
- Integrate applications previously not possible without programming
- Automatically detect your applications' user interface controls and build automations and integrations utilizing those controls
- Expose controls to the OpenSpan Studio designer for rapid drag and drop automations
- Smoothly integrate data between disparate applications based on your own configurable rules

OpenSpan Studio is an advanced, yet highly intuitive visual design environment that enables IT, implementation specialists and even domain experts to easily integrate and automate one or more desktop applications without either writing code or the need for additional hardware or middle-tier server investment.

When a desktop application is running, it is actually the operating system that manages the user interface (GUI) — Text boxes, buttons, menus, toolboxes, icons, images, links, mouse clicks, events, etc. The operating system and application communicate with each other at all times, constantly passing data and events back and forth. OpenSpan technology intercepts and deciphers the communications between the operating system and desktop application.

Utilize your existing GUIs with no dip in productivity and no training costs
Users can continue to use the applications they are comfortable with, but are no longer forced to waste time re-entering data, cutting and pasting, and executing the same tasks over and over again. Of course, if you'd like to build a complete new GUI – go right ahead… OpenSpan makes it possible.

Configuring Your View of the OpenSpan Studio Windows

The OpenSpan Studio designer contains three main categories of windows; Design, Tool, and Debug.

Design

The project item Design pane appears in OpenSpan Studio when you add or select a project item in the Solution Explorer. It is not dockable and cannot be arranged like the Tool and Debug windows, and it occupies all space not utilized by docked Tool and Debug windows. The Design pane is formatted based on the project item type selected (that is, the Design pane appears differently and contains different controls/functions for an Automation project item versus a Web Adapter project item).

Remember - An Adapter is a project item that allows users to integrate with an application built upon a specific platform (e.g., Windows, Web) by interrogating the application to produce metadata that describes the application. Adapters have accessible properties, methods and events.

Tool

There are five different types of Tool windows. Each allows you to perform specific tasks.

1. **Properties** – Displays component, adapter, automation, or project design properties. Note that these are not all of the properties associated with a selected project item. This window only displays the properties that have been selected as *Design* properties (visible on this window). Other item properties display from the Object Explorer - Configuration dialog.
2. **Solution Explorer** – Project Items such as automations, adapters, and folders appear here.
3. **Toolbox** – All controls and components for use in OpenSpan Studio solutions are located here. You can add controls to this toolbox from external .dll files and you can customize these windows to suit your design needs.
4. **Object Explorer** – All solution controls such as interrogated application targets, automations, and controls/components display in the Tree-view in the top part of this window. The lower part of the window displays properties, methods and events for any item selected in the Tree-View.
5. **Navigator** – The location(s) of a selected object is displayed here. For invalid connections due to broken links and missing controls, the names of all invalid connections display in Navigator.

Debug

There are five types of Debug windows.

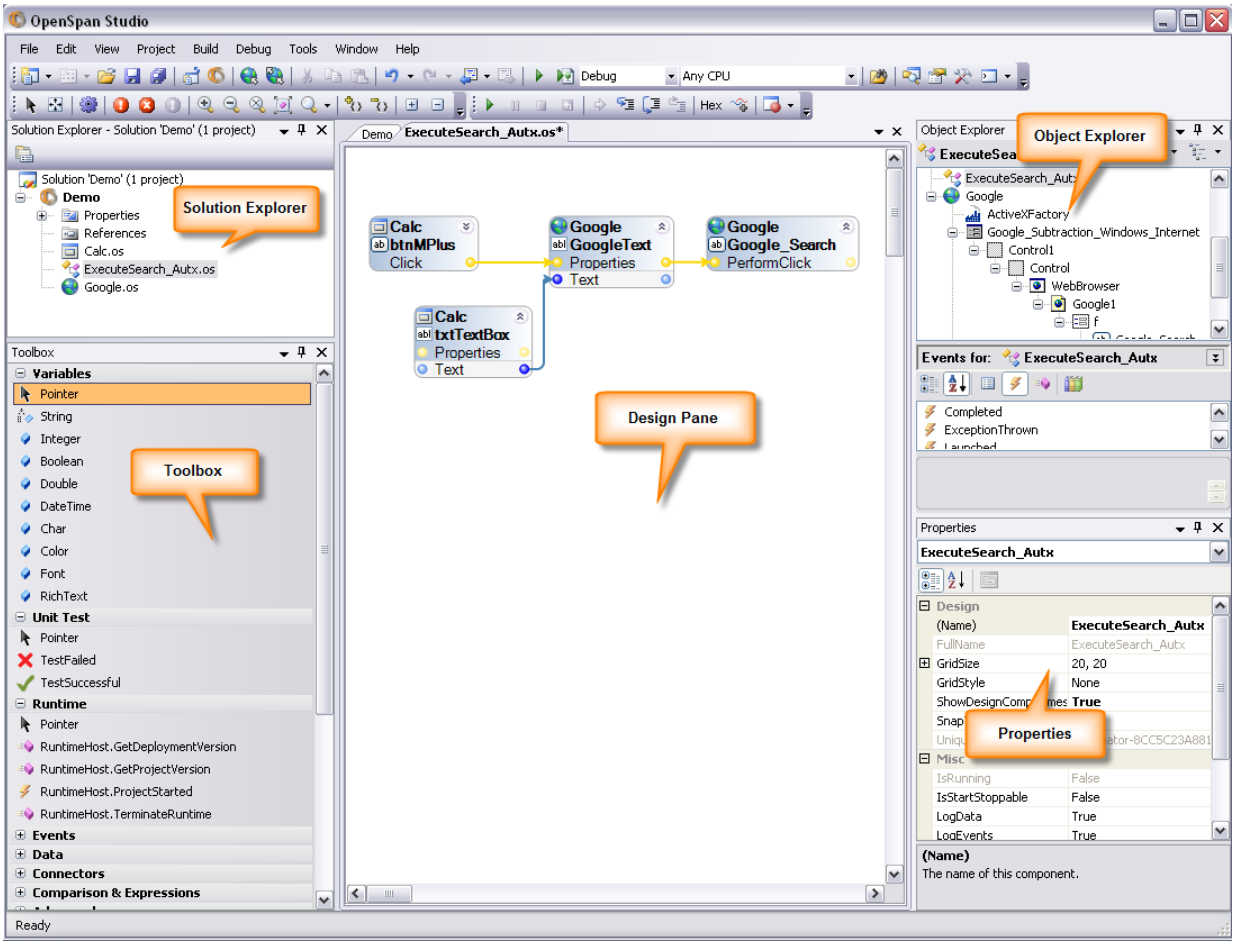
1. **Breakpoints** – Displays your current breakpoints in tree view or automation format. You can delete, enable, disable or focus on them as needed. Breakpoints are user-defined locations or conditions that pause project execution as needed for debugging purposes.
2. **Call Stack** – Allows you to monitor the currently selected thread and displays the stack trace for the selected component during project execution.
3. **Automation Locals** – Displays the local components, variables and parameters associated with a solution as it executes.
4. **Automation Watches** – Allows you to monitor the output values for your automation components and controls.
5. **Threads** – Allows you to examine the threads in the project solution as you debug it.

Note: You can customize your viewing and editing screen space by arranging the windows. When you have a window layout that you like, OpenSpan Studio automatically saves this layout between sessions so you do not have to reposition windows every time you open the program.

Window Arrangement

OpenSpan Studio's movable/dockable Tool and Debug windows provide almost unlimited flexibility in setting up your workspace. You can display multiple window types (as in the example below), stack them and access them through a tabbed interface, have them Auto Hide and display only when you need them, or enable them to float over the application – it's completely up to you.

The window arrangement below is a popular setup. In this example, all Tool window types are displayed, allowing you the greater flexibility and complete access to the available features



Window Categories

There are two categories of windows: **Dockable** and **Un-dockable**

1. Tool and Debug windows are movable and **dockable** windows that allow you to perform specific tasks. They can be reshaped and moved anywhere on the OpenSpan Studio user interface. You can turn off the docking feature for this category of window.
2. The Design Pane is an **un-dockable** window that is the work area for project items. You use the design pane to interrogate applications, create automations, and set configuration properties. The format of the design pane depends on the project item you have opened (from the Solution Explorer). The placement and size of the design pane accommodates the layout of the Tool and Debug windows.

Tool and Debug Window Settings

There are two basic settings for arranging Tool and Debug windows in OpenSpan Studio:

1. **Docked / Floating**—The window can be attached (docked) to any side of its parent window, or it can be detached and floating within the OpenSpan Studio application window.
2. **Close/Auto Hide**—These features enable you to make the OpenSpan Studio work area larger by moving Tool and Debug windows out of view.
   - A window hidden using the Close option (i.e., clicking the window title bar **X** command button) can only be viewed again by selecting it from the **View** menu.
   - A window hidden using the Auto Hide option is made available by moving your cursor over the tab containing the window's name. The window slides back into view, ready for use.

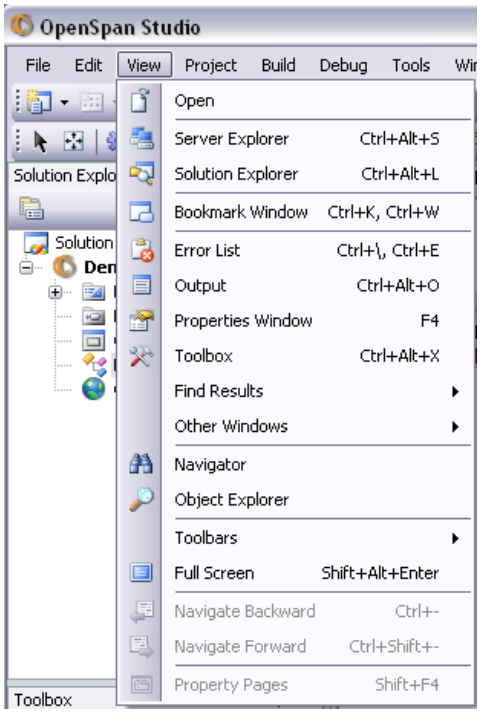
Refer to the following procedures in the OpenSpan Studio Help Contents for additional assistance with configuring the OpenSpan Studio design environment:

- Opening a Window Type
- Docking Tool and Debug Windows
- Hiding Tool and Debug Windows
- Separating Docked Windows
- Changing the Shape of Windows

Opening a Tool or Debug Window

To view a Tool window, perform the following steps:

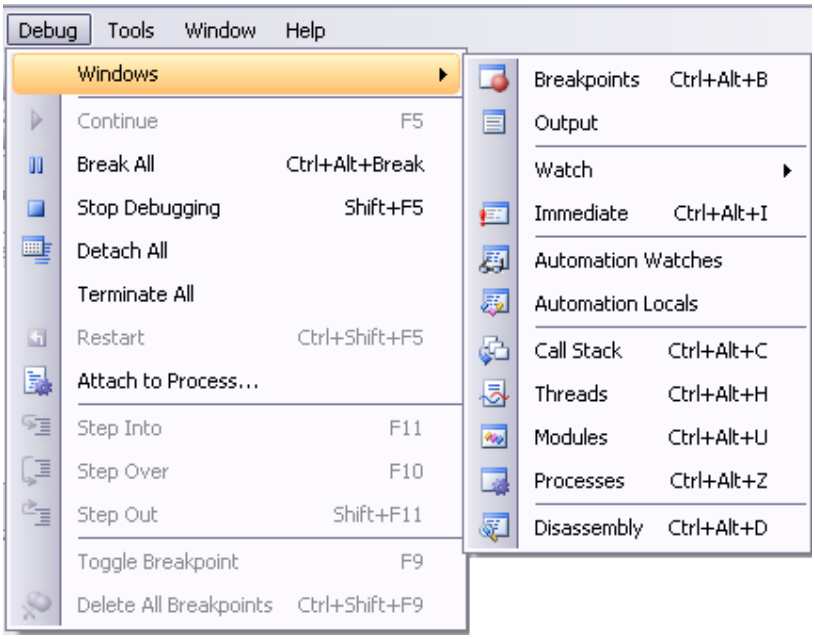
1. From the **View** menu, select the specific Tool window you want to open.



2. The window you select appears in the OpenSpan Studio window.

To view a Debug window, perform the following Steps:

1. From the Debug menu, Select Windows then the specific debug window you want to open.



2. The available debug windows will change depending is a project is running or not.

Docking Tool and Debug Windows

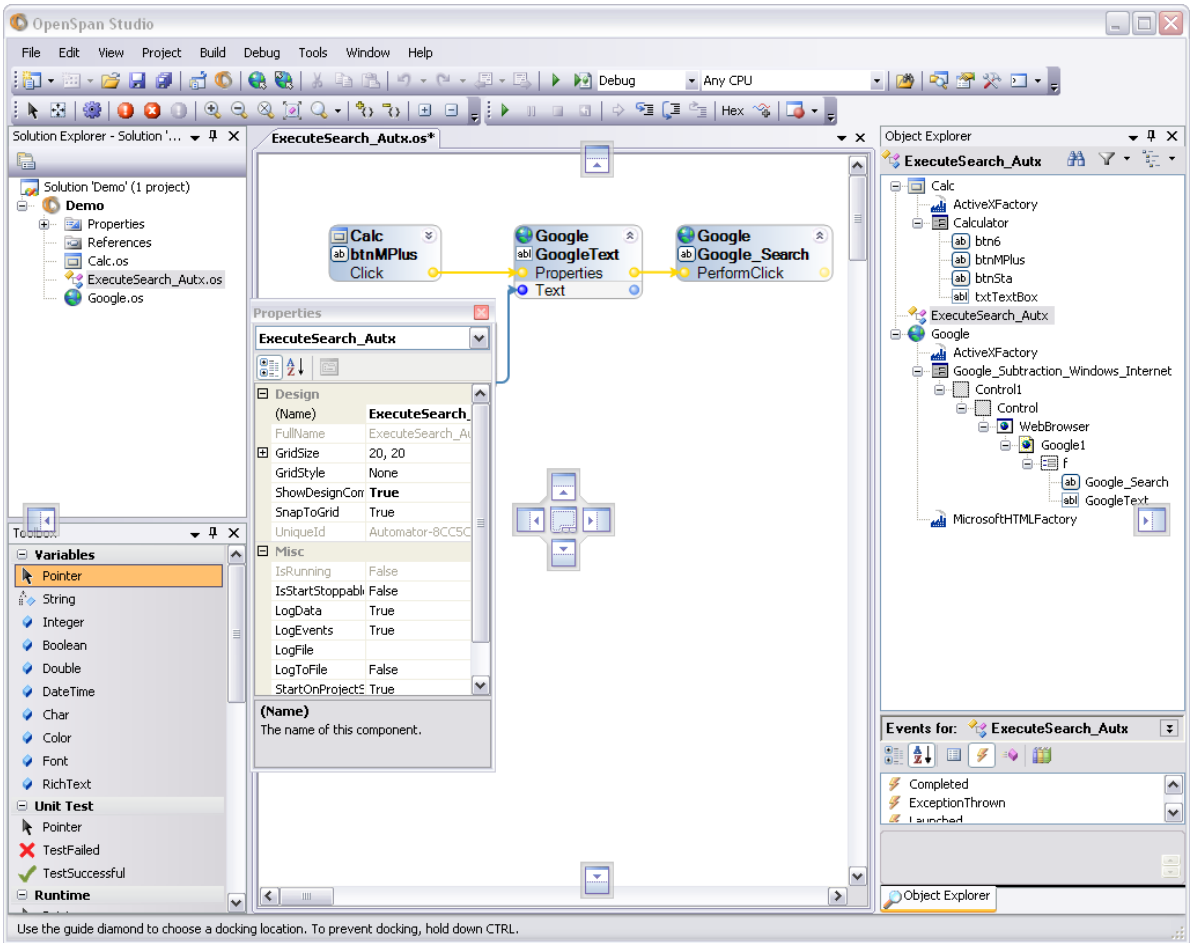
OpenSpan Studio provides the flexibility to position Tool or Debug windows in virtually any location on the screen.

All Tool and Debug windows are dockable and can be:

- Docked to the sides of the OpenSpan Studio window
- Docked to the sides of the design pane
- Tab-docked to other windows
- Collapsed or hidden at the side of the OpenSpan Studio window, appearing only when you move your cursor over the tab containing the window's name
- Floating above your application in separate windows

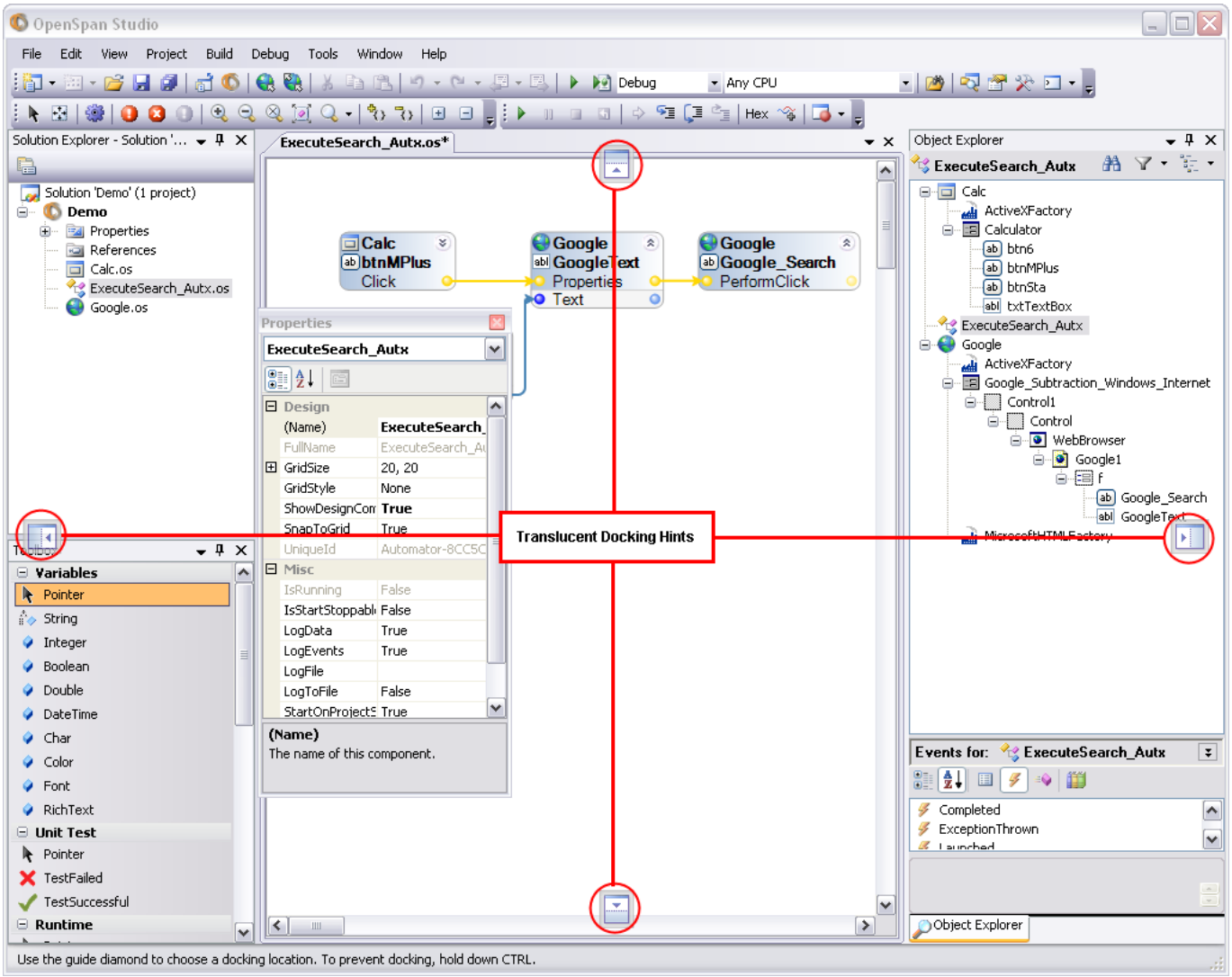
To dock or undock a window, click and hold the window title bar or tab and drag it to another location. When you move the window, a transparent representation of the window displays along with blue translucent docking hints to help guide you to the desired location.

Note: You can also double-click a Tool or Debug window title bar to dock or undock the window.

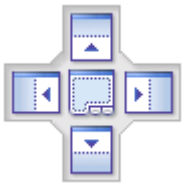


Translucent Docking Hints

When you drag a window from its docked or floating position, a transparent representation of the window displays along with blue translucent docking hints. These hints enable you to quickly dock the window to predetermined locations. Hints display at the center of each main window edge: top, bottom, left, and right.



When you drag a Tool or Debug window into the design pane or over another docked or tab-docked window, an additional docking hint displays.

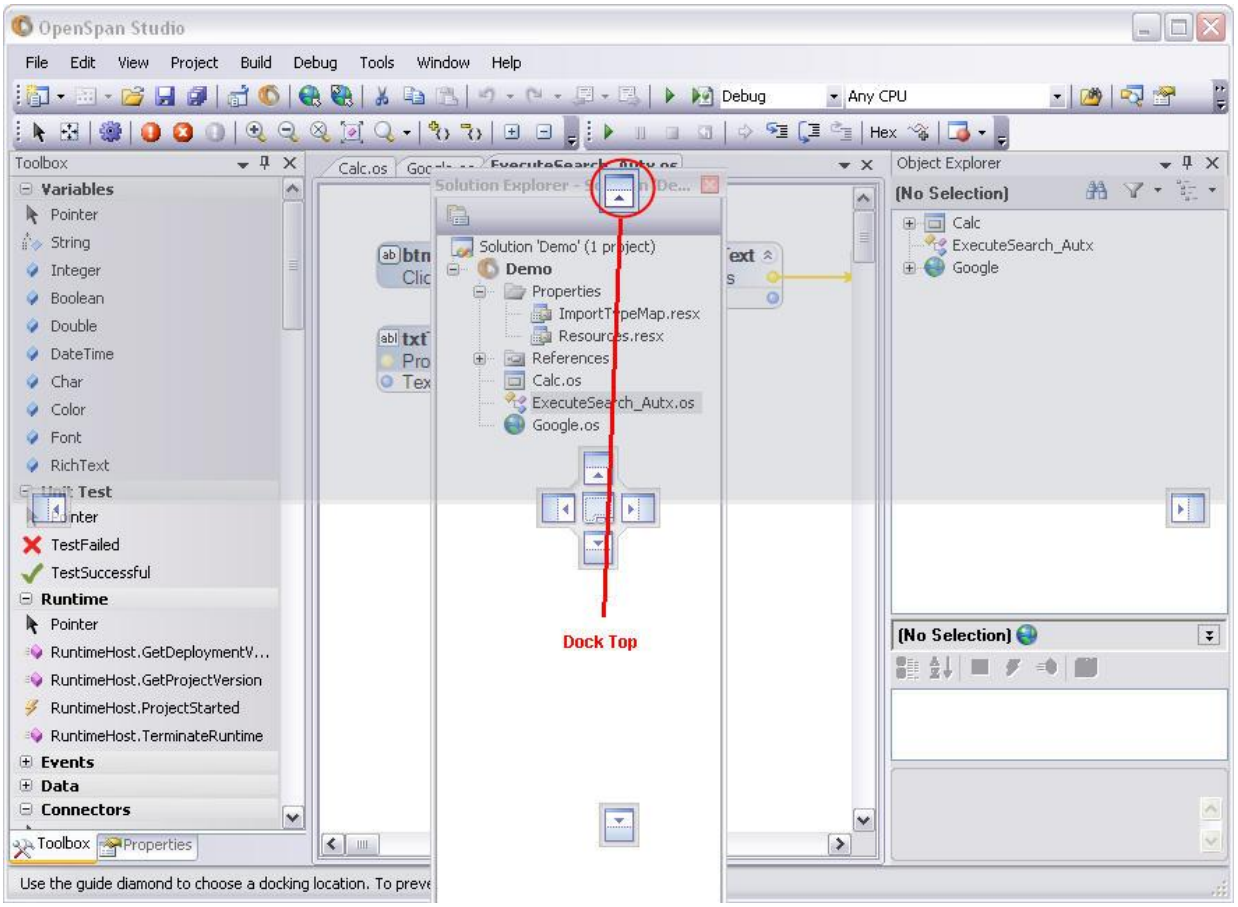


This four-sided docking hint enables you to dock the window within the edges of the design pane, to the edges of a docked Tool or Debug window, or tab-docked on top of another Tool or Debug window.

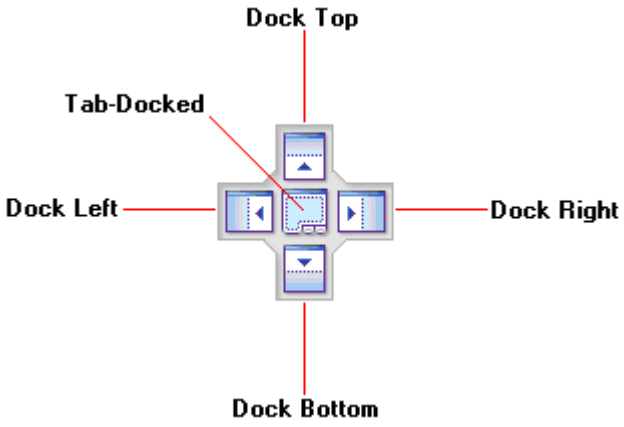
Positioning Tool and Debug Windows Using Translucent Docking Hints

When you drag a Tool or Debug window and position the mouse pointer over one of the translucent docking hints, a representation of the window displays in a predetermined location.

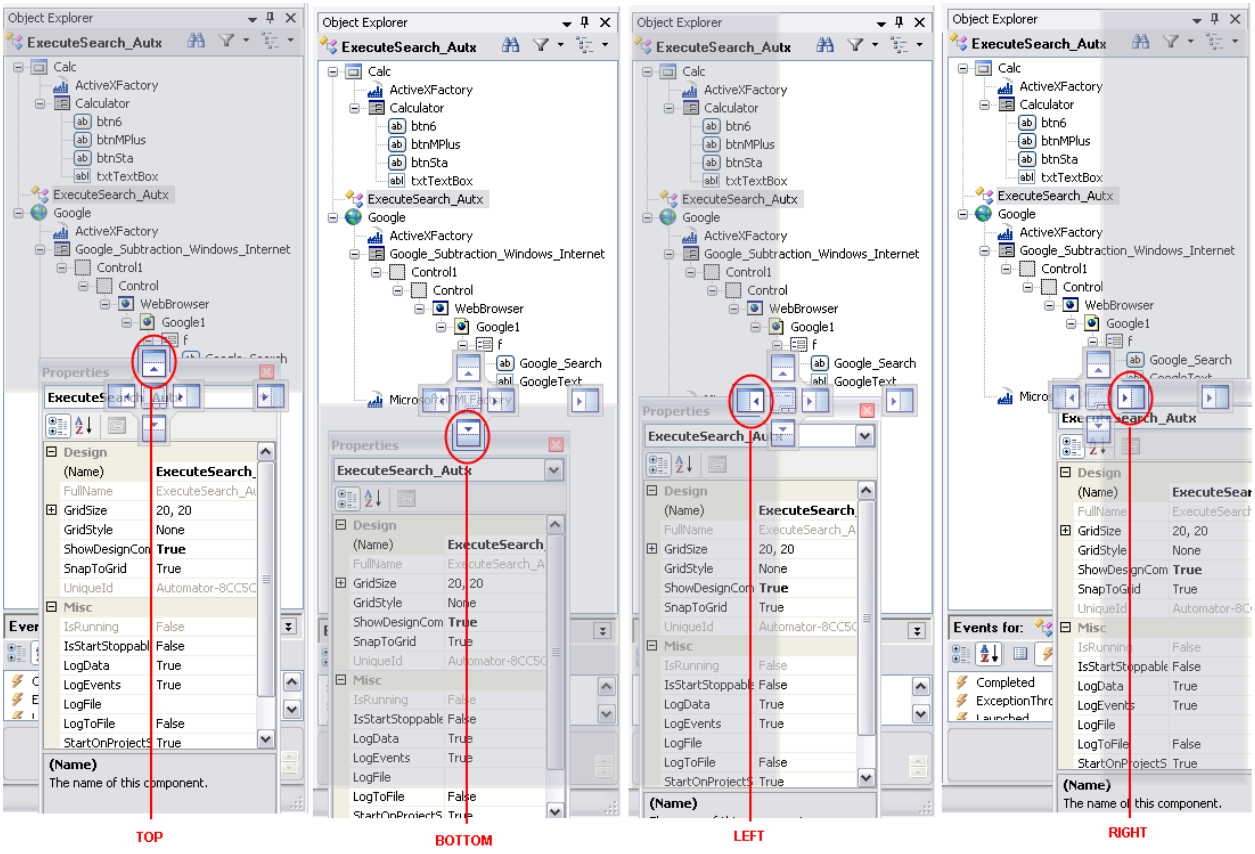
For example, if you position the mouse pointer over the hint at the top of the OpenSpan Studio window, the representation of the window displays horizontally across the top of the screen. If you release the mouse button, the window docks in that location and the screen adjusts accordingly.



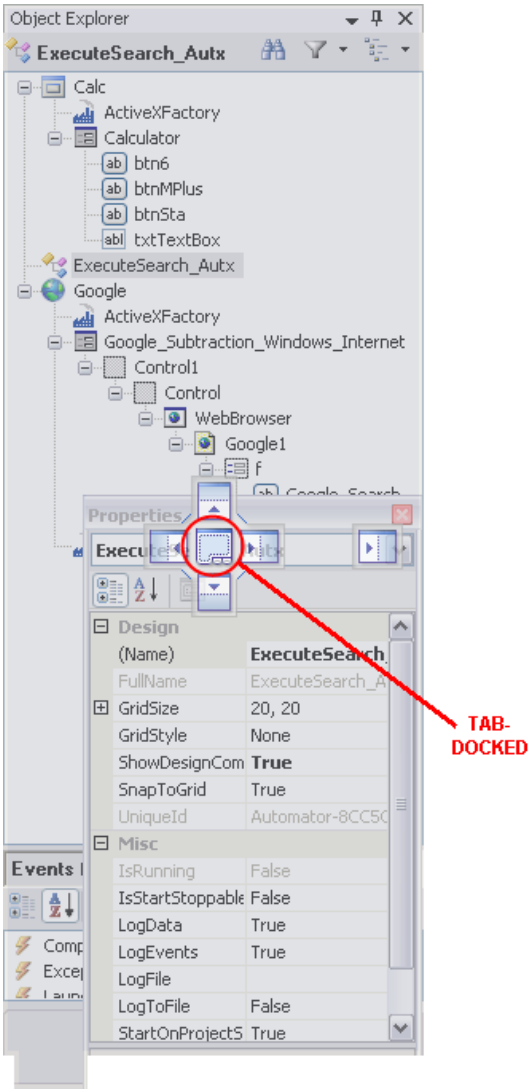
The four-sided translucent docking hint displays when you drag a Tool or Debug window over another window or the design pane. It enables you to easily position the window at the top, bottom, left or right, or stack it on top (tab docked) of another Tool or Debug window.



For example, if you drag a Tool or Debug window over a docked window, and position the mouse pointer over one of the hints, the window representation displays in the predetermined location indicated on the hint.



If you position the mouse pointer over the center docking hint, the representation of the dockable window displays on top of the docked window.



If you release the mouse button over the center docking hint, the Tool or Debug window docks on top and tabs display at the bottom representing all windows in the stack. You can access any of the “tab-docked” windows in the stack by clicking a tab.

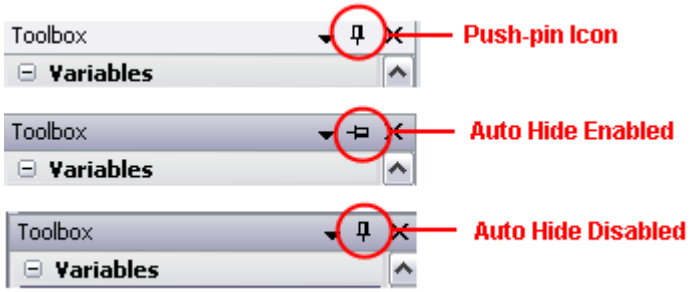
Auto Hiding Tool and Debug Windows

OpenSpan Studio Tool and Debug windows support a feature called Auto Hide. Auto Hide allows you to see more of your workspace by minimizing windows along the edges of the OpenSpan Studio window.

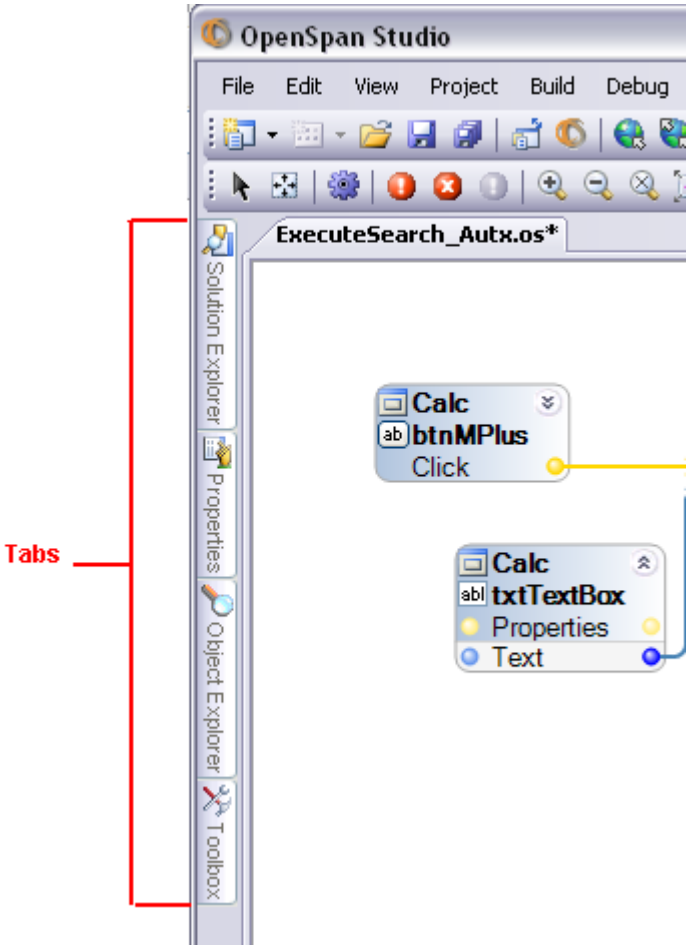
Note: Auto Hide is only available when the window is docked. If the window is in floating mode, you cannot Auto Hide it.

To enable the Auto Hide feature:

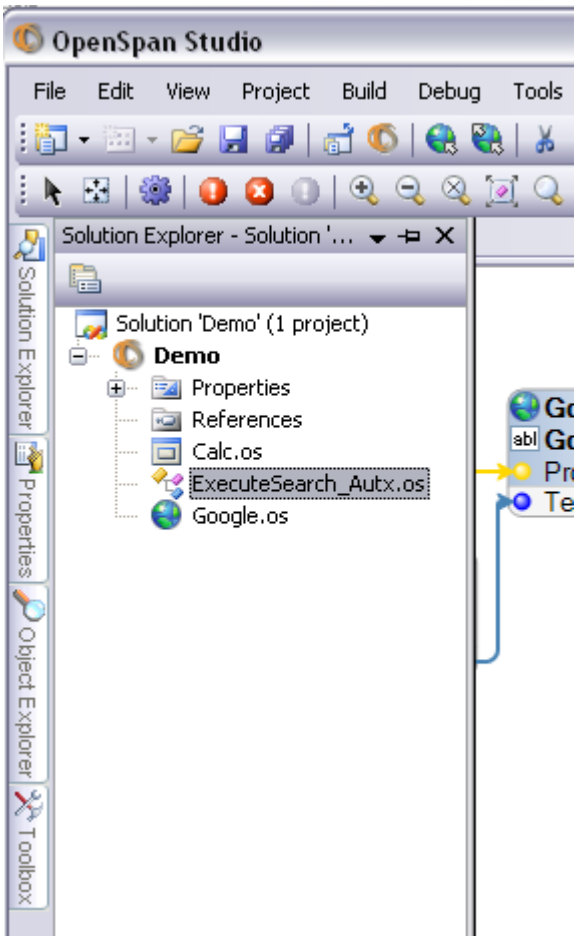
1. Click the **Push-pin** icon in the window title bar to enable and disable the Auto Hide feature.



When Auto Hide is enabled for a window, a tab with the window's name appears on an edge of the OpenSpan Studio window.


OpenSpan Studio
File
Edit
View
Project
Build
Debug
ExecuteSearch_Autx.os*
Solution Explorer
Properties
Object Explorer
Toolbox
Tabs
Calc
btnMPlus
Click
Calc
txtTextBox
Properties
Text

2. To restore a hidden window, simply click the tab and the window in Auto Hide mode slides into view.



3. To slide the window back out of view, click anywhere outside of the window.

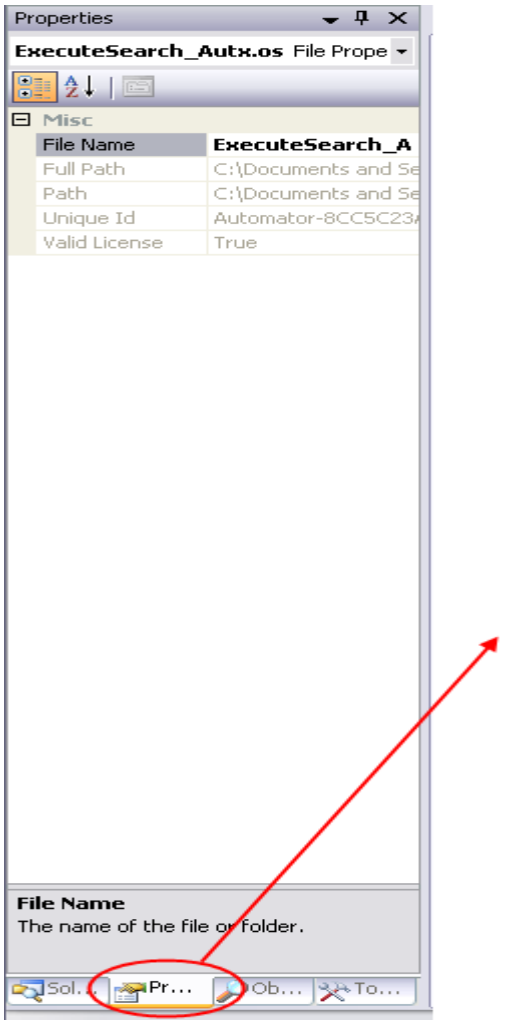
To disable Auto Hide:

1. Display the Auto Hide window.
2. Click the **Push-pin** icon.

Separating Docked Windows

To separate windows that are tab-docked, perform the following steps:

1. Click and hold a tab at the bottom of the tab-docked windows.

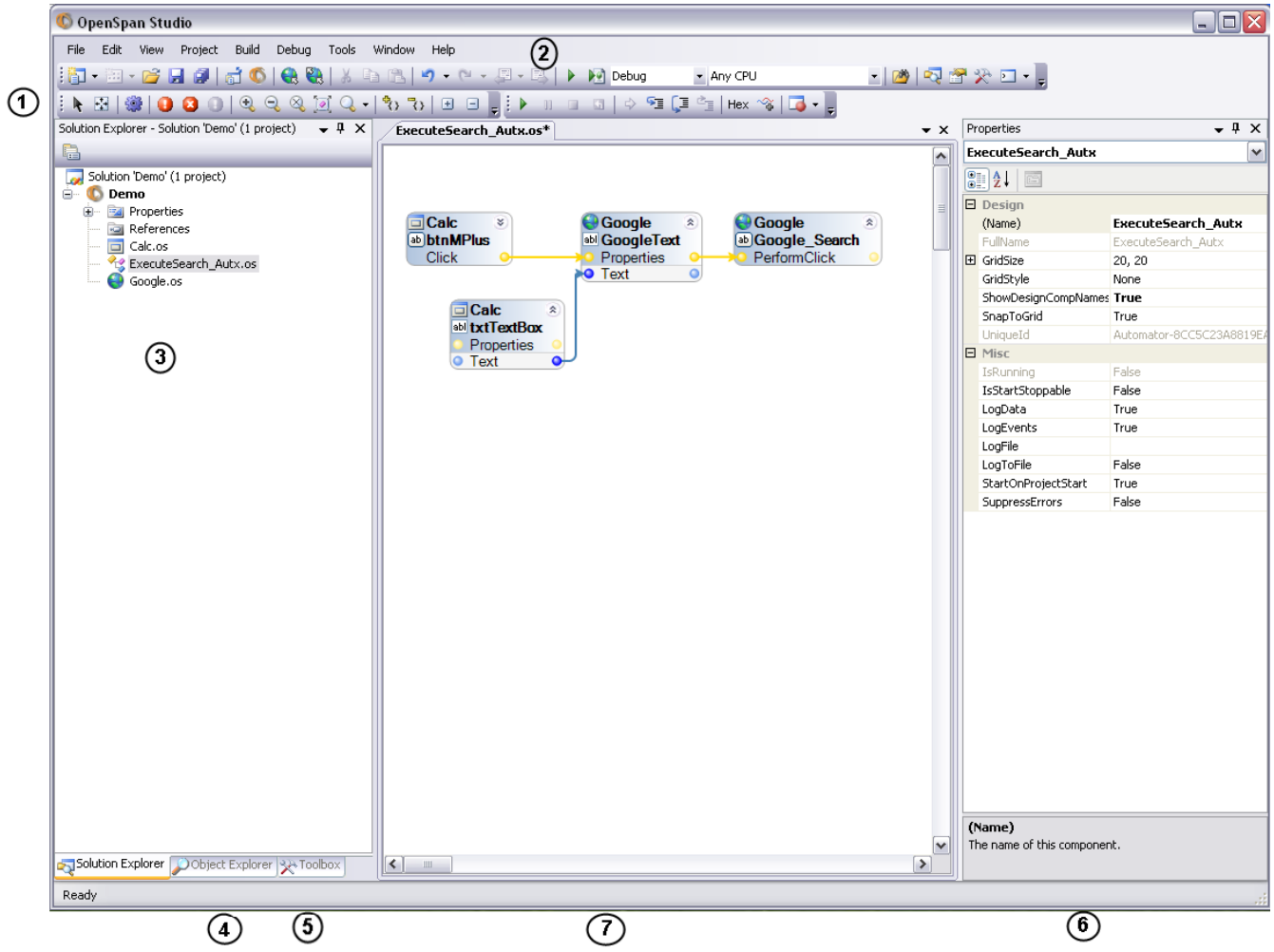


2. Drag the tab (window) to a new location and release.

Functional Areas and Interface Features

The OpenSpan Studio application provides an interface for creating new and composite application projects. Projects are deployable files that can be executed from either OpenSpan Studio or OpenSpan Integrator.

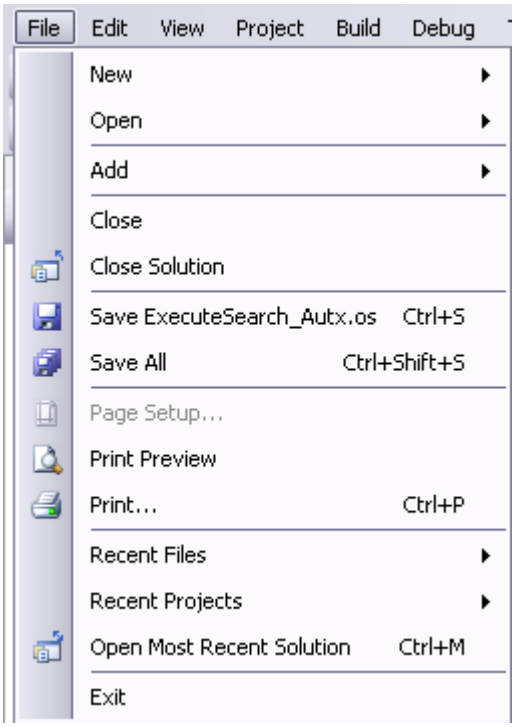
With OpenSpan Studio, you do not need to write code. All logic can be specified through a visual representation of the business logic you want to perform.



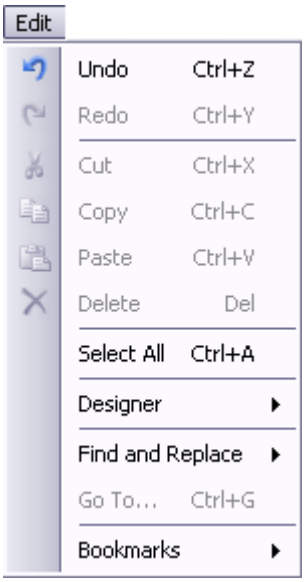
① Main Menu

The main menu contains access to the functions required to create, debug, and modify solutions, deploy projects, and configure OpenSpan Studio diagnostic messaging. You can also set options for how to display items in the designer and you can access product documentation. These Menu options are fully documented on the MSDN web site. (http://msdn.microsoft.com/en-us/library/ee482141.aspx)

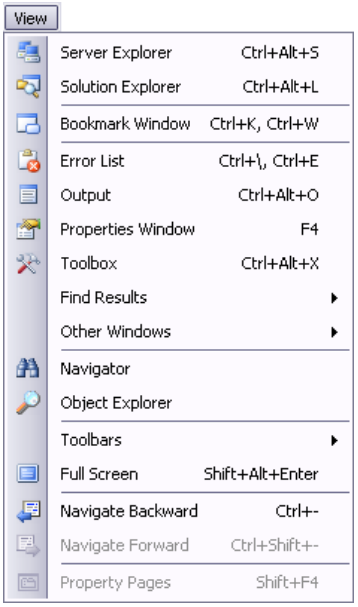
Menu	Options
File	The **File** Menu allows you to open and save solutions. With the **File** menu you can create new solutions or print work that you have already performed. You'll learn about integrations, automations and solutions in later chapters.



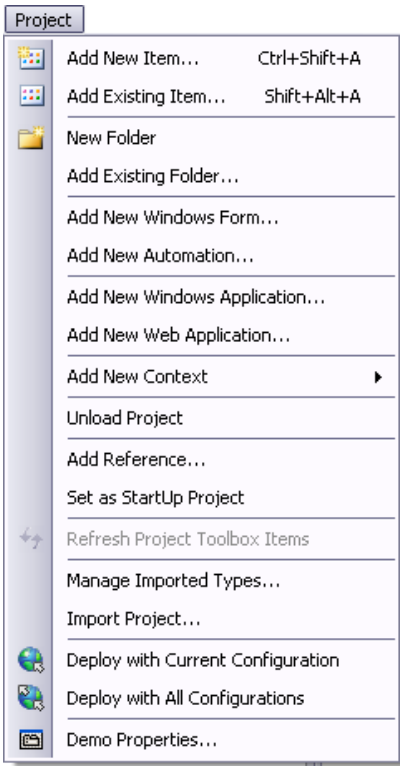
Menu	Options
Edit	The **Edit** menu allows you to Undo, Redo, cut, copy, paste and delete data in the same format found in many other Windows based applications. It also permits you to select all the elements on a screen.



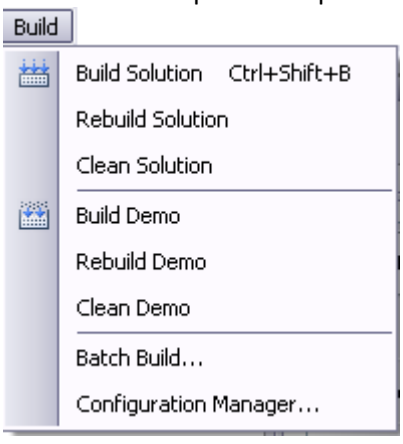
View	Use the **View** menu to open Tool and Debug windows.



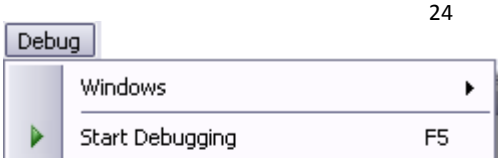
Menu	Options
Project	The Project menu allows you view and modify settings for the active top-level project. It is the location where you can add or remove project items to your project. You can also created deployment packages for run-time.

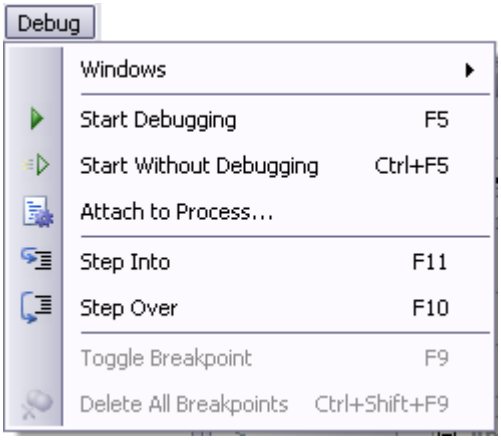
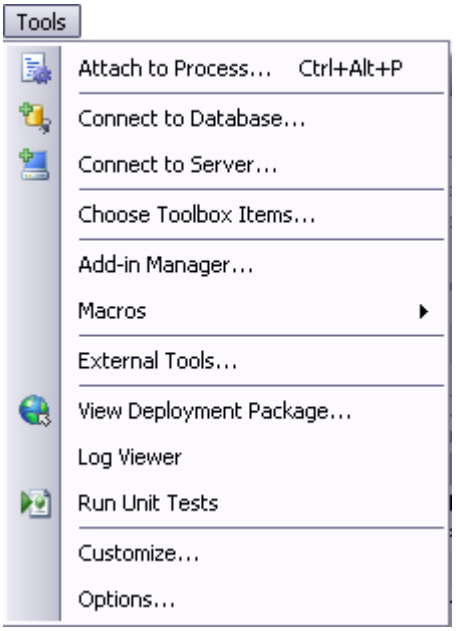


Build	The Build menu provides options for your design, such as building run-time images.

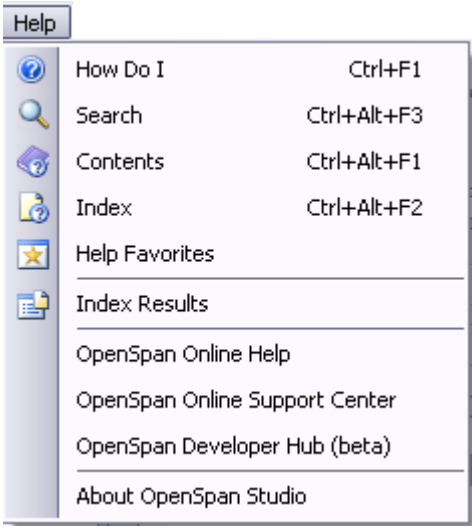


Debug	The **Debug** menu contains a wide variety of utilities that enable you to observe and correct solution errors. The **Start** and **Step Into/Over** commands will launch your automation and begin a debugging session. The **Attach To Process** command lets you break into a running automation and begin a debug session. You can use Breakpoints to tell the debugger where and when to pause the execution of a solution.



Menu	Options
	
Tools	The Options dialog allows access to the OpenSpan diagnostics, Deployment Security and Naming Rules through **Options**. The **Auto-Naming Rules** dialog lists what characters will be used to name adapter controls as well as what characters will be replaced when auto-naming. The **Diagnostics Configuration** dialog controls powerful diagnostic tools that can help you to improve your solution. Many of these topics are covered in later chapters.
	

Menu	Options
Help	Selecting **OpenSpan Online Help** will provide you the latest topics available from the web-based help system. **OpenSpan Online Support Center** takes you to the OpenSpan Support Center web portal where you can search the OpenSpan Knowledge Base, chat with support representatives, or submit a support request. The **MSDN on the Web** options take you directly to the Microsoft Developers Network. Select the **About OpenSpan Studio** menu item to see what version of OpenSpan Studio is installed.



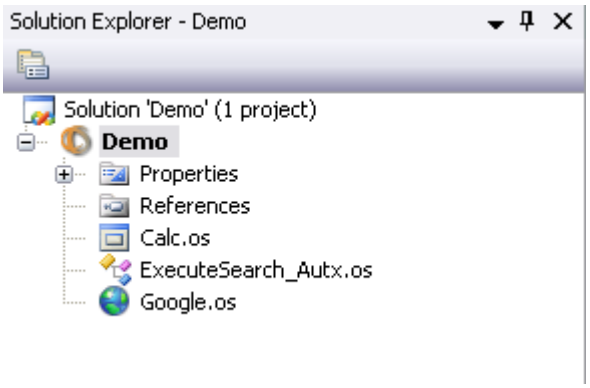
② Toolbar

The main toolbar contains shortcut icons to the commonly used functions within OpenSpan Studio.

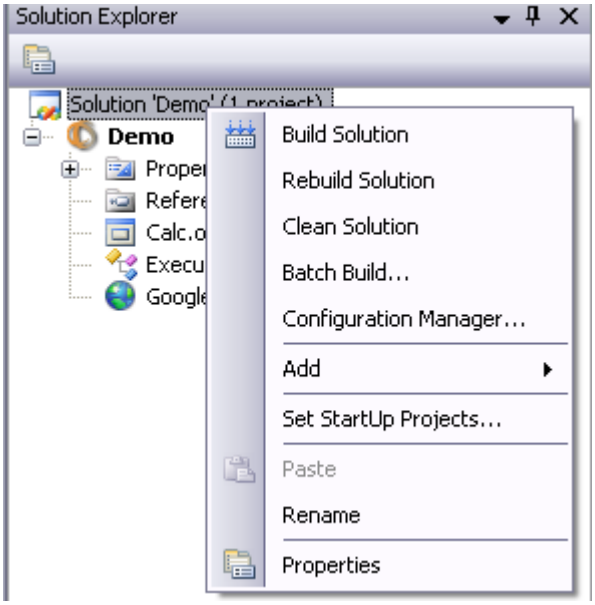
③ Solution Explorer

The **Solution Explorer** window lists the main objects of the solution, including projects, adapters, and forms. In the sample image shown below, the solution contains a single project containing an Automation, Web Application, and Windows Application. You use the **Solution Explorer** to open the project items within a project.

Note: OpenSpan support integration with web-based applications running in Internet Explorer versions 6, 7, and 8.



From the **Solution Explorer** window, **right-click** the solution to access the following context menu options:

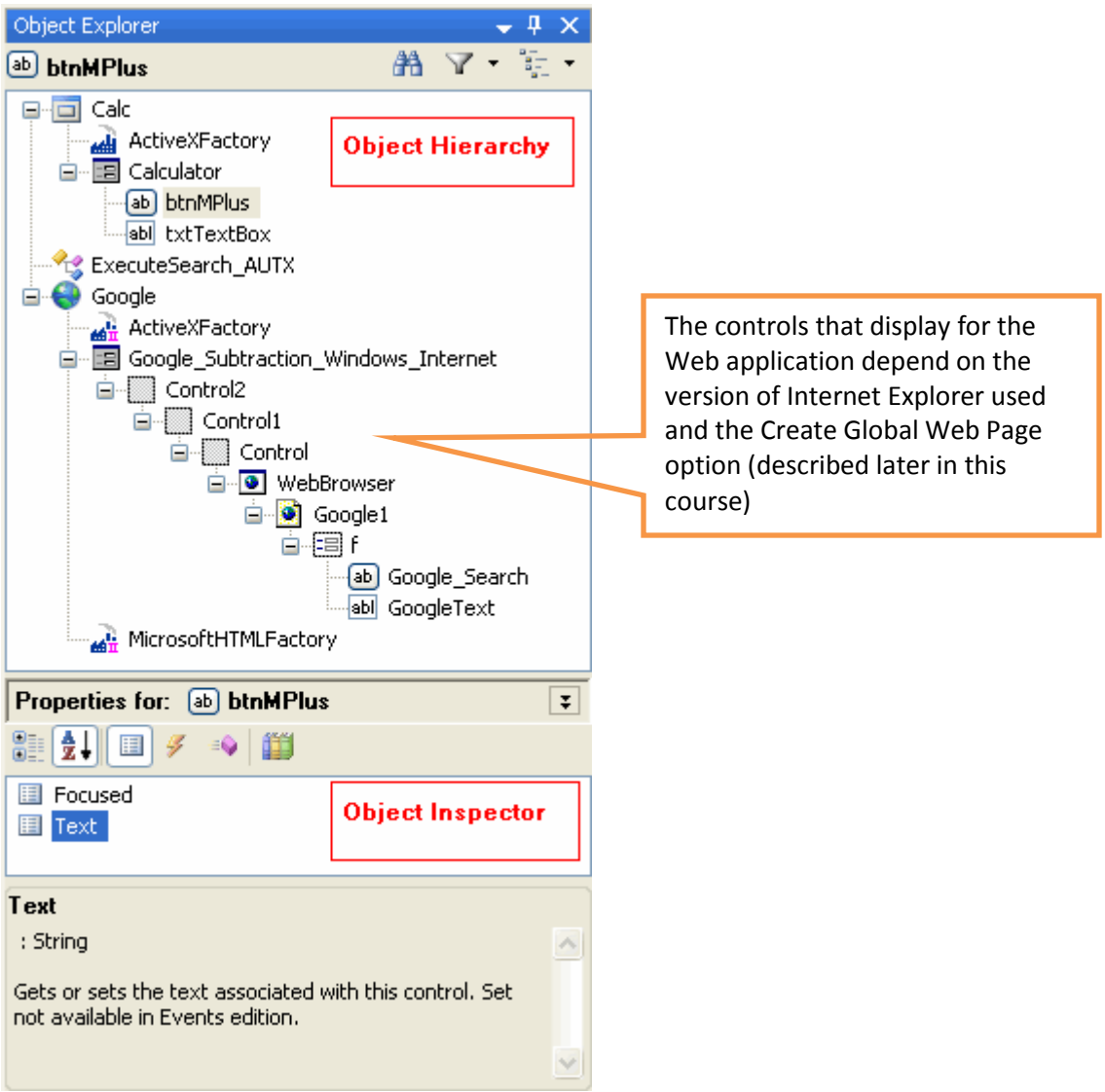


④ Object Explorer

The **Object Explorer** window lists all controls contained in a solution and provides access to the properties, methods and events required to design solutions. The Object Explorer has two panes: Object Hierarchy and Object Inspector.

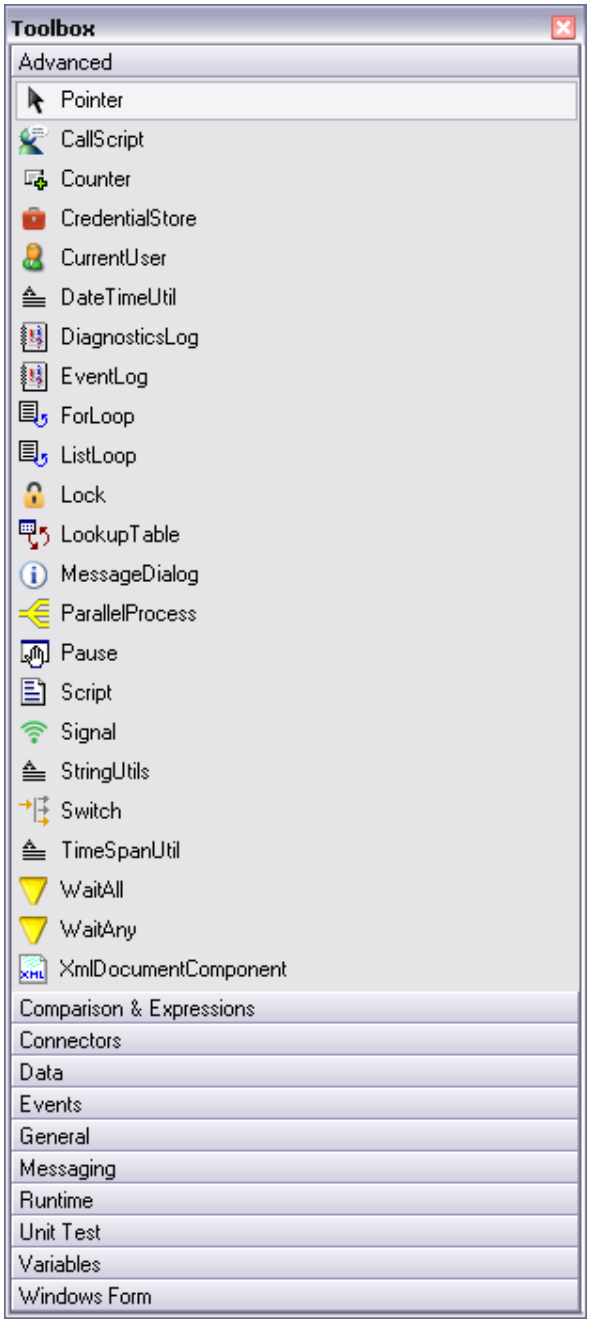
Project Items, such as adapters and automations display in the Object Hierarchy. Interrogated controls display beneath the corresponding adapter in a hierarch representing how the controls are parented in the target application. Beneath the Object Hierarchy, the Object Inspector pane displays properties, methods, and events for the control selected in the Object Hierarchy. You use the Object Explorer to locate a particular property, method or event for use in an OpenSpan automation. Automations are special design panes within OpenSpan Studio that enable you to layout data and event logic between controls and/or components. Automations will be covered in more detail later in this training.

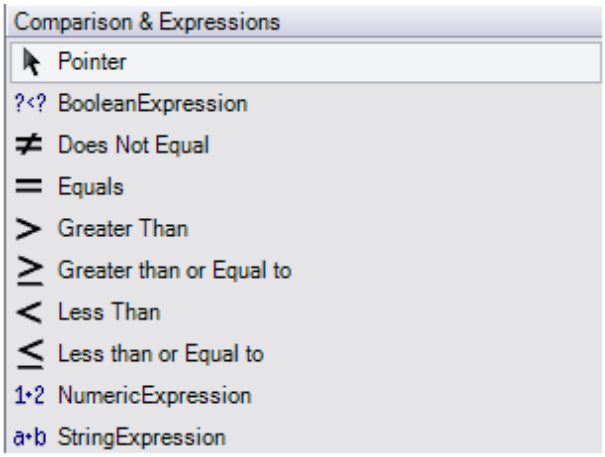
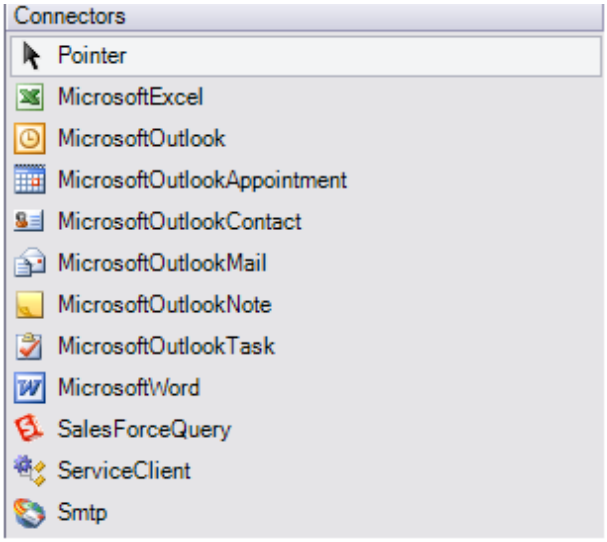
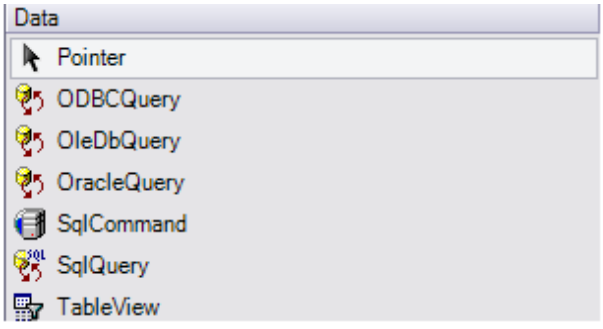
An example of the Object Explorer showing a Windows application (Calc.exe), Web application (www.google.com), and an automation (ExecuteSearch_Autx) is shown below.

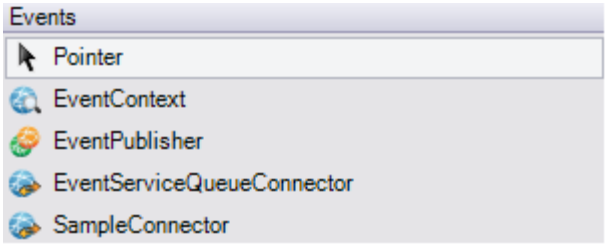
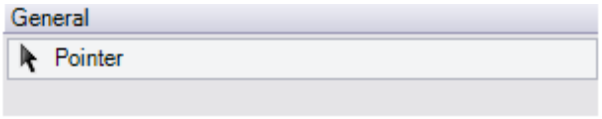
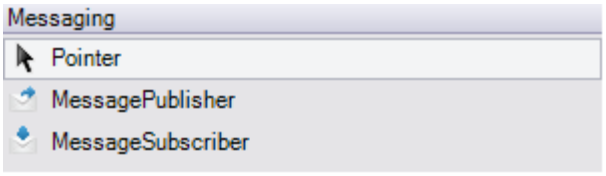
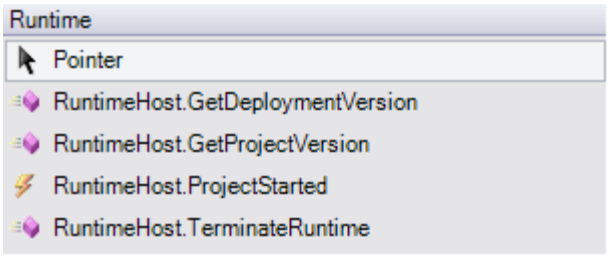
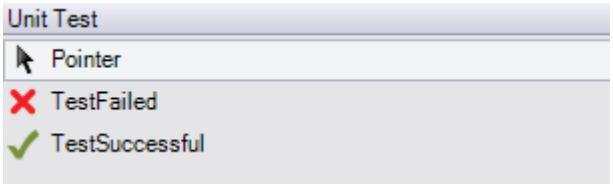


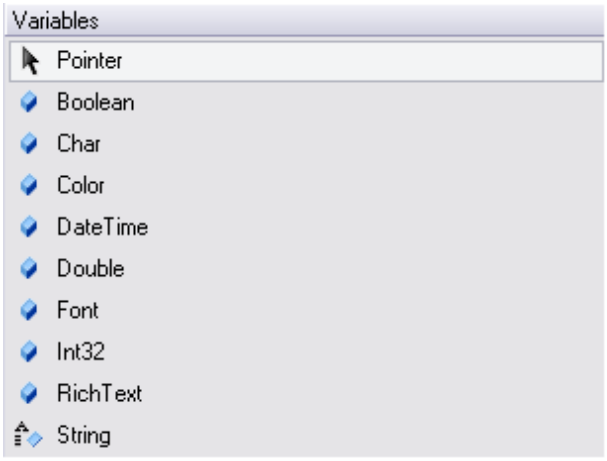
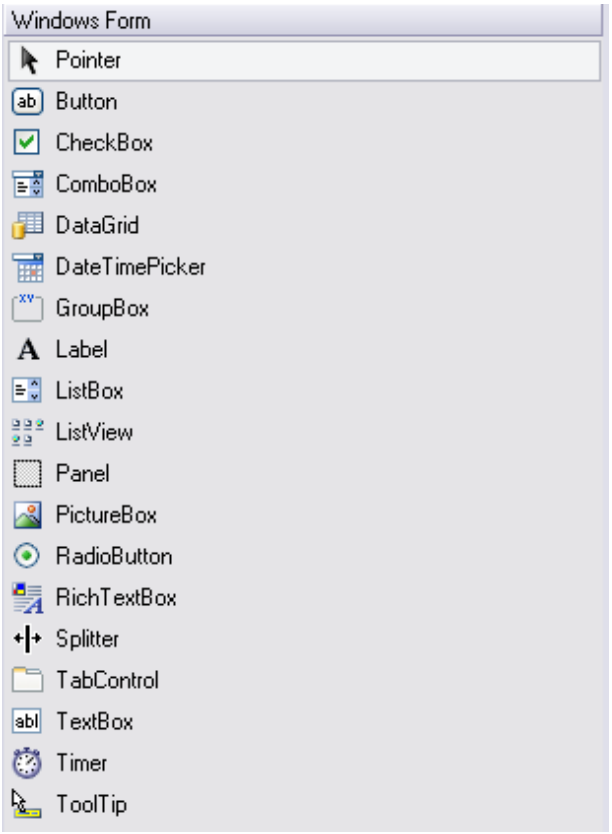
⑤ Toolbox

The **Toolbox** window contains the .Net components and the OpenSpan Studio components available for use in solutions. The standard .Net components are useful when creating WinForm and Application bars in solutions. The OpenSpan Studio components are designed for use in automations when working with application adapters. These components are located on the additional tabs.

Tab	**Available Components**
Advanced	

Tab	**Available Components**
Comparison Expressions	
Connectors	
Data	

Tab	**Available Components**
Events	
General	
Messaging	
Runtime	
Unit Test	

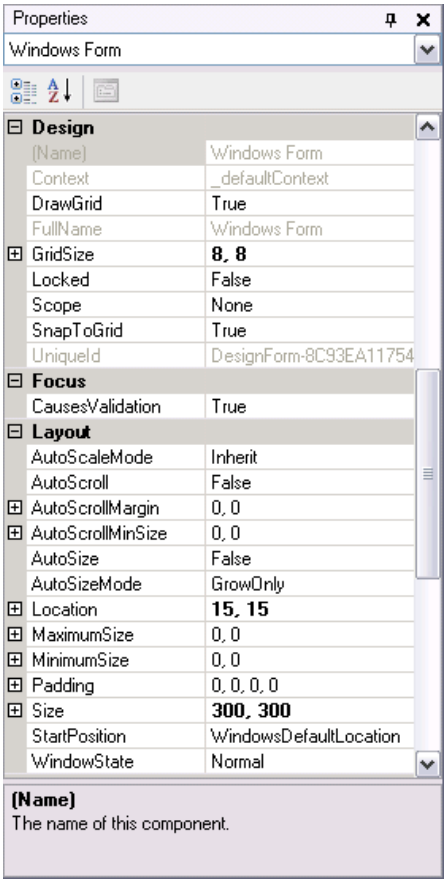
Tab	**Available Components**
Variables	 **Note:** The **Variables** tab components are only visible when an automation pane is open.
Windows Form	

⑥ Properties

The design properties which define how controls and components display on WinForms and Application bars are accessible from the **Properties** window. You can display the properties either alphabetically or categorized (as shown in the image below).

For information on these controls and their use, refer to the following Microsoft website:

http://msdn.microsoft.com/en-us/library/system.windows.forms.aspx



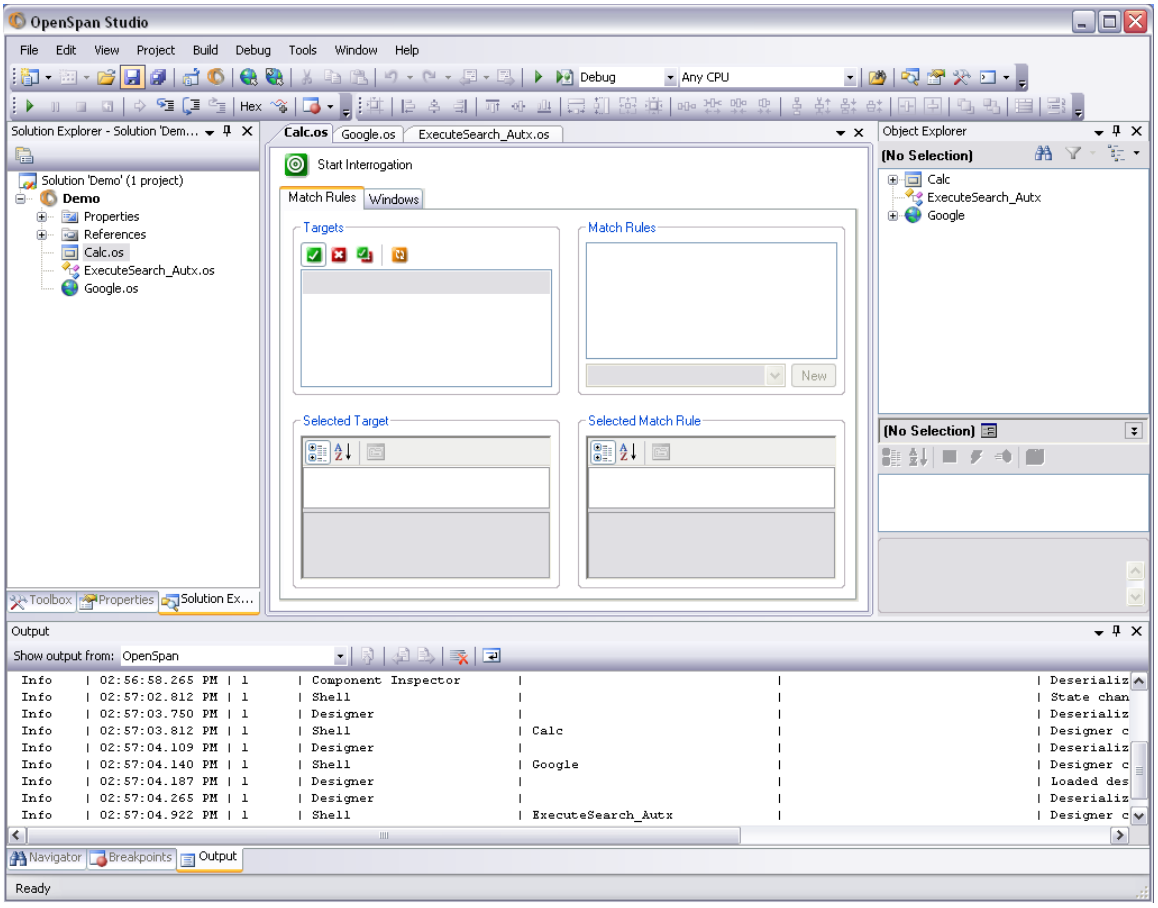
Note: All OpenSpan Studio solutions and project items have at least some properties that display in the Properties window. For example, the design name and component ID display in the **Properties** window, along with information about how project items behave at project startup.

⑦ Design Pane

For each project item, you use the design pane to graphically layout the components and logic for these items and specify how the items work together within the project. An example of the Application bar design pane is shown below.

These pages are stored as .os files (note the .os extension on the page tab). The format and functions available from the design pane depends on corresponding project item.

To open a design pane, from the **Solution Explorer**, right-click the project item and select **Open**. You may also open a design pane by double-clicking the item in the Solution Explorer or in the Object Explorer by right-clicking a project item and selecting **Open** from the context menu.



CHAPTER 2: GETTING STARTED WITH OPENSPAN STUDIO

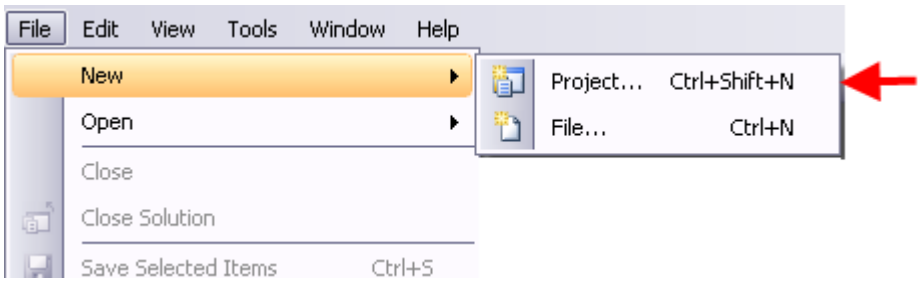
In this chapter, you will learn your first basic techniques for using OpenSpan Studio. OpenSpan solutions can be comprised of a single or multiple projects and one or more project items, such as Windows forms, adapters, and automations.

OpenSpan automations contain the logic for executing OpenSpan Studio projects. You will use automations to establish data and execution paths between and within integrated applications. Adapters enable OpenSpan solutions to interact with applications. A solution, therefore, is a combination of automations and adapters that work together to integrate your applications.

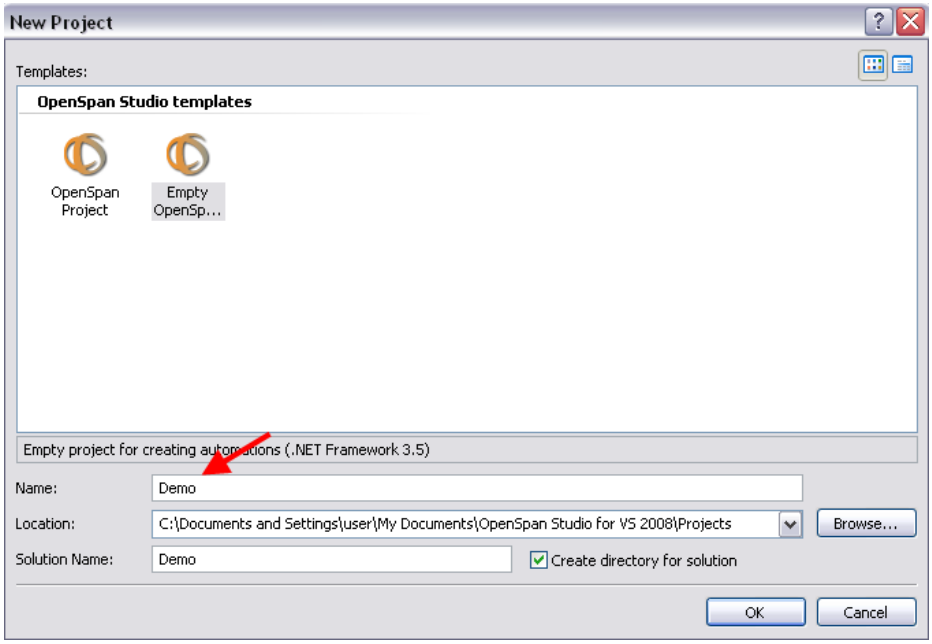
The following exercise provides an introduction to the OpenSpan Studio design interface. Upon completion, you will have designed a solution that shows the interaction between a Windows application – the Windows Calculator program, and a web site – Google search.

Exercise 1 - Building Your First Solution

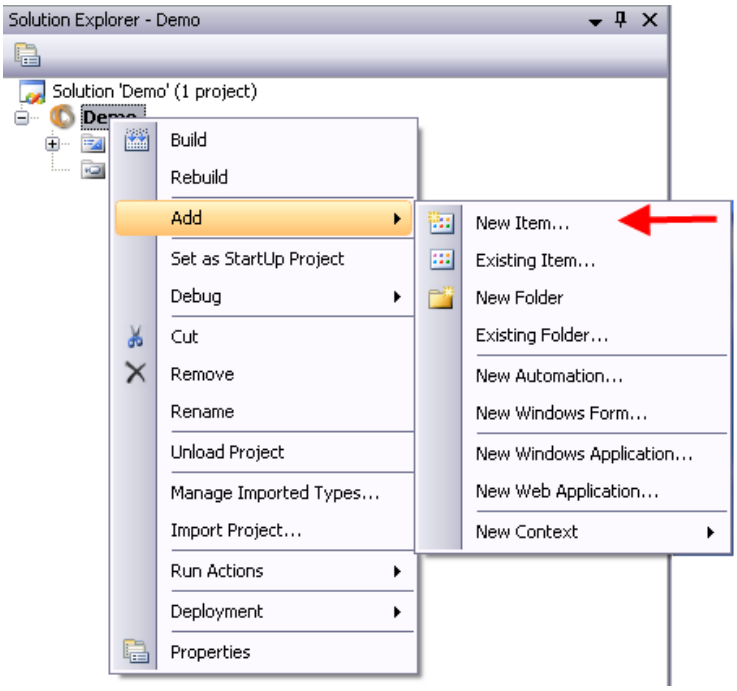
1. Start OpenSpan Studio.
2. Select **File | New |Project**.



3. Select **Empty OpenSpan Project** and type **Demo** in the **Name** field, enter a folder location or accept the default, and then select **OK**.

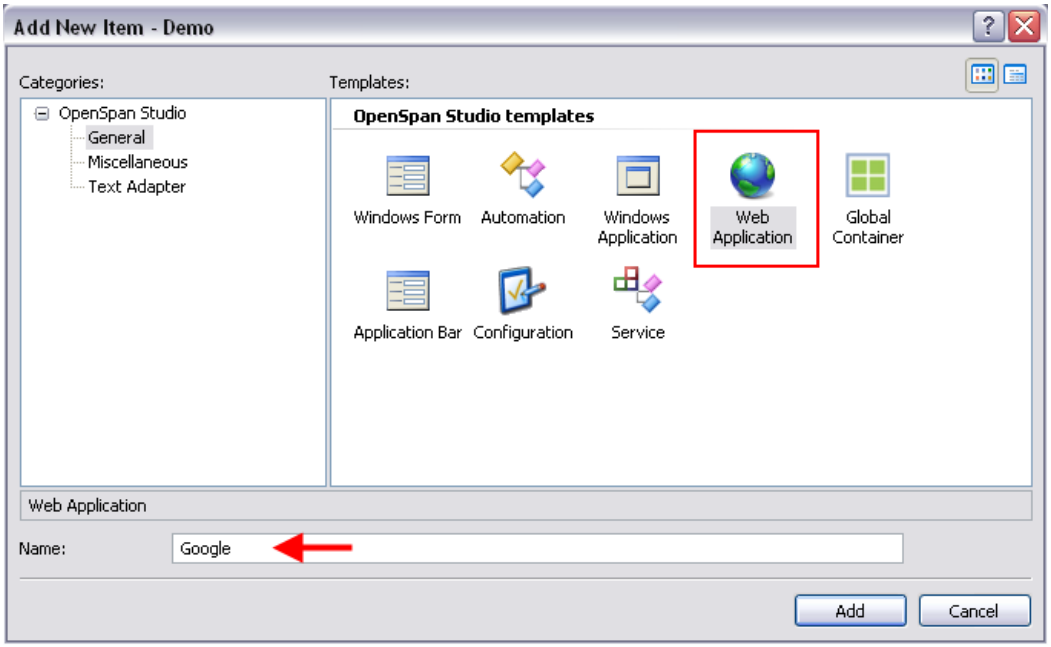


In the **Solution Explorer** window, **right-click** the **Demo** project and select **Add | New Item** from the context menu.

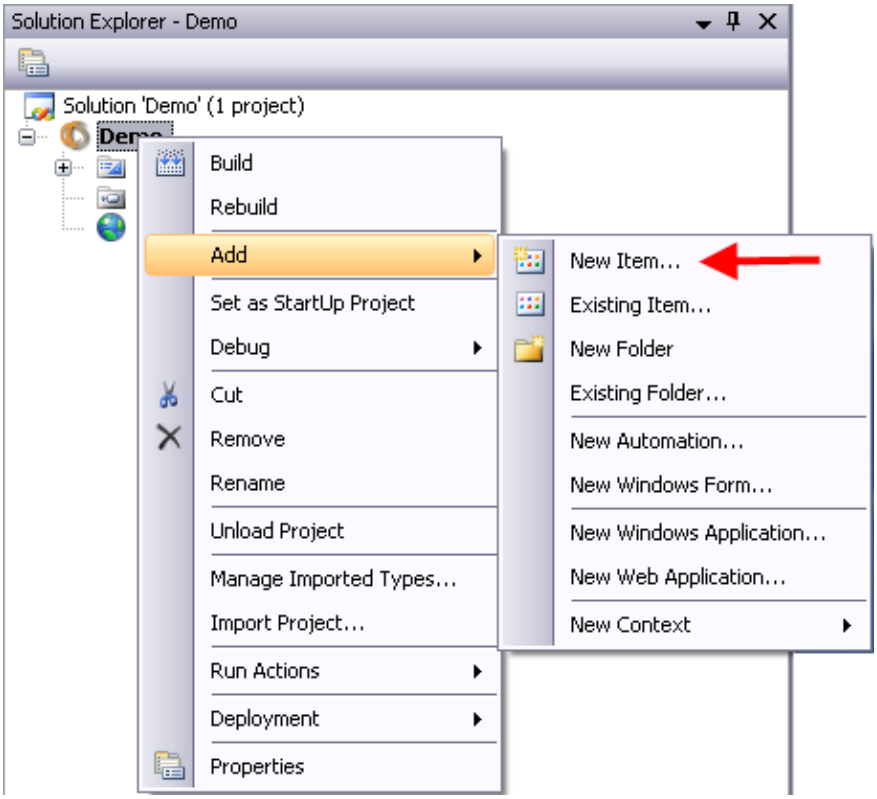


4. The **Add New Item** dialog opens. This dialog displays all the available Design Components that you can add to a solution.

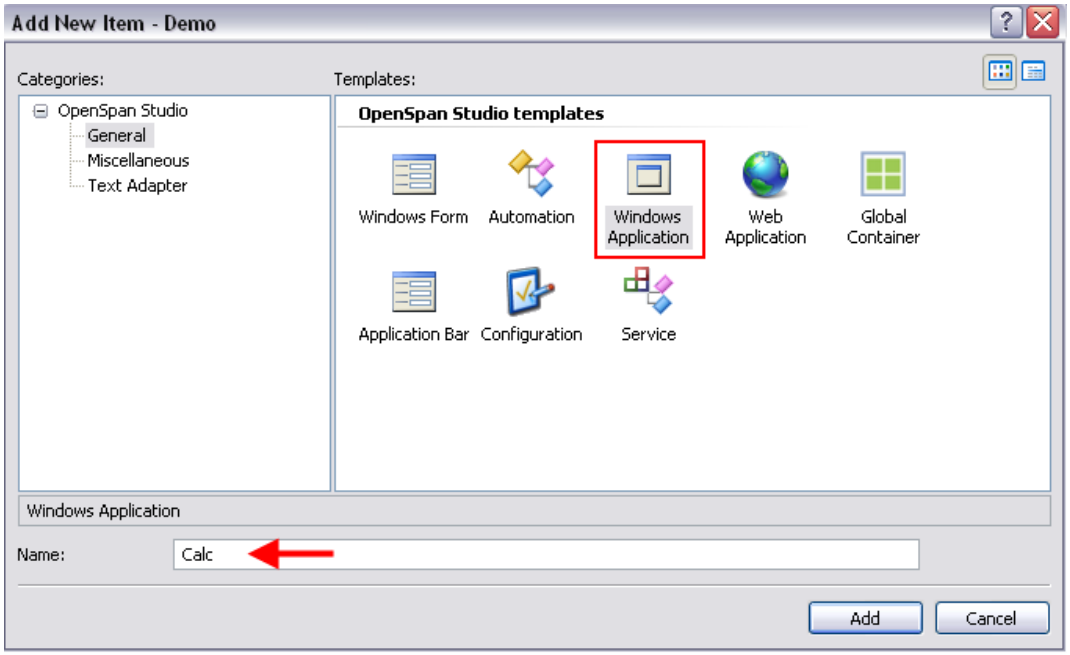
5. Ensure the **General** category is selected, and then click the **Web Application** template.
6. Type **Google** in the **Name** field, and then select **Add**.



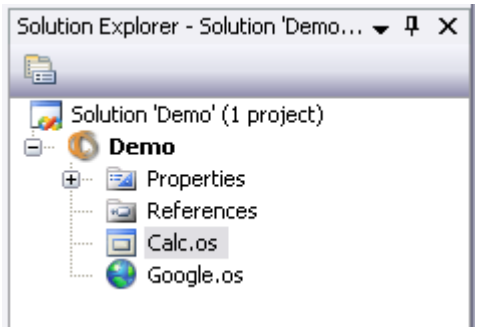
7. Add another new project item by **right-clicking** the **Demo** project and selecting **Add | New Item** from the context menu.



8. Ensure the **General** category is selected, and then click the **Windows Application** template.
9. Type **Calc** in the **Name** field, and then select **Add**.



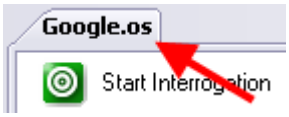
Your **Solution Explorer** window should now look like this:



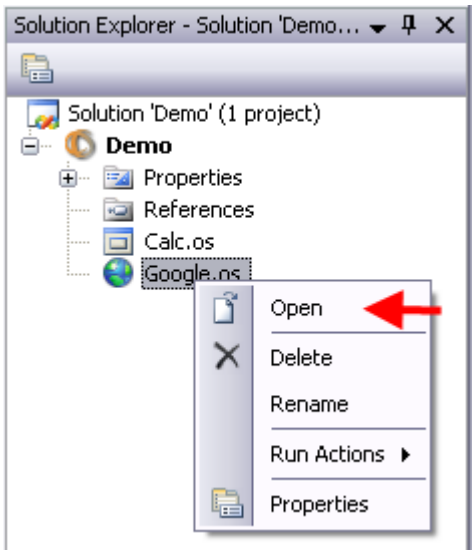
10. Select **File | Save All**.

Setting Properties for the Web Application

1. In the OpenSpan Studio design pane, click the **Google.os** tab to bring the Google design pane to the front.

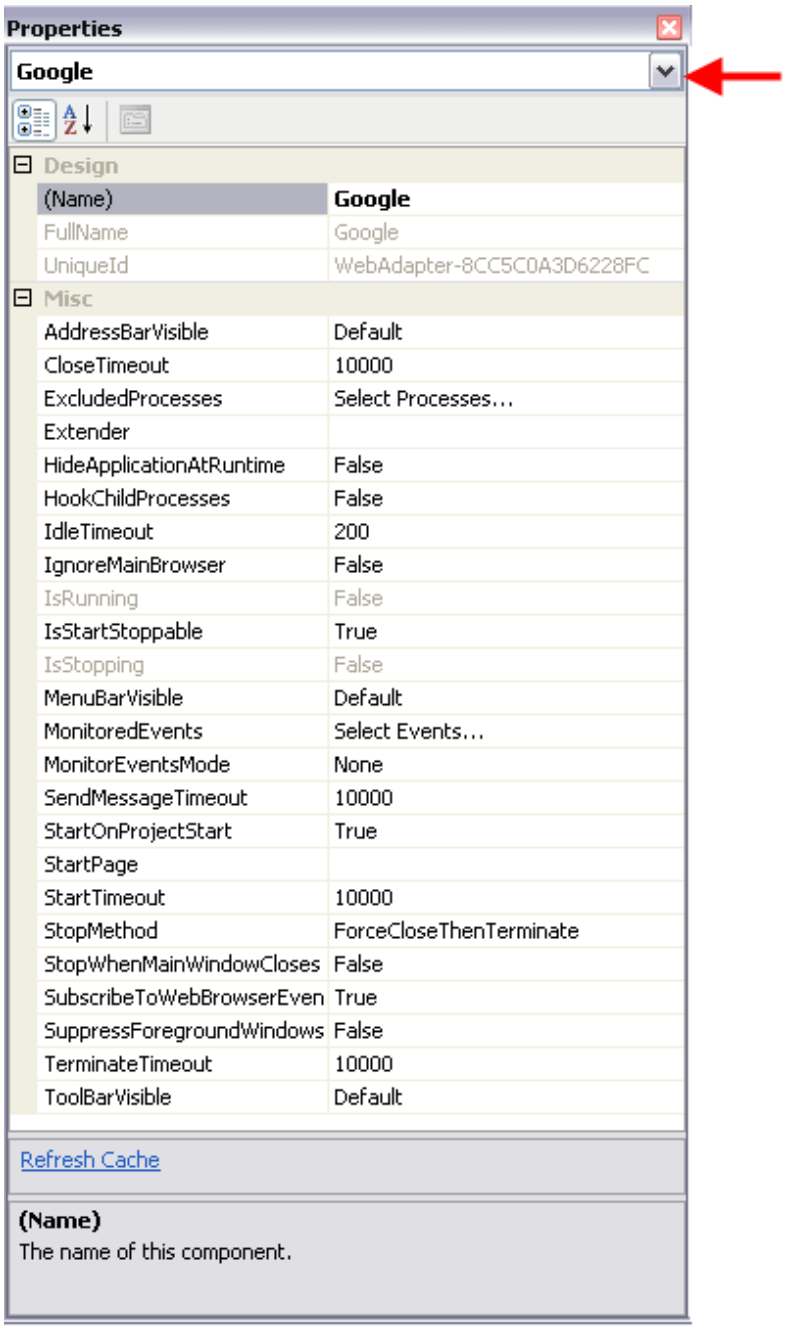


Note: If the design pane is not open (i.e., you don't see the **Google.os** tab), then in the **Solution** Explorer window, **right-click** the **Google** web application and select **Open** from the context menu.

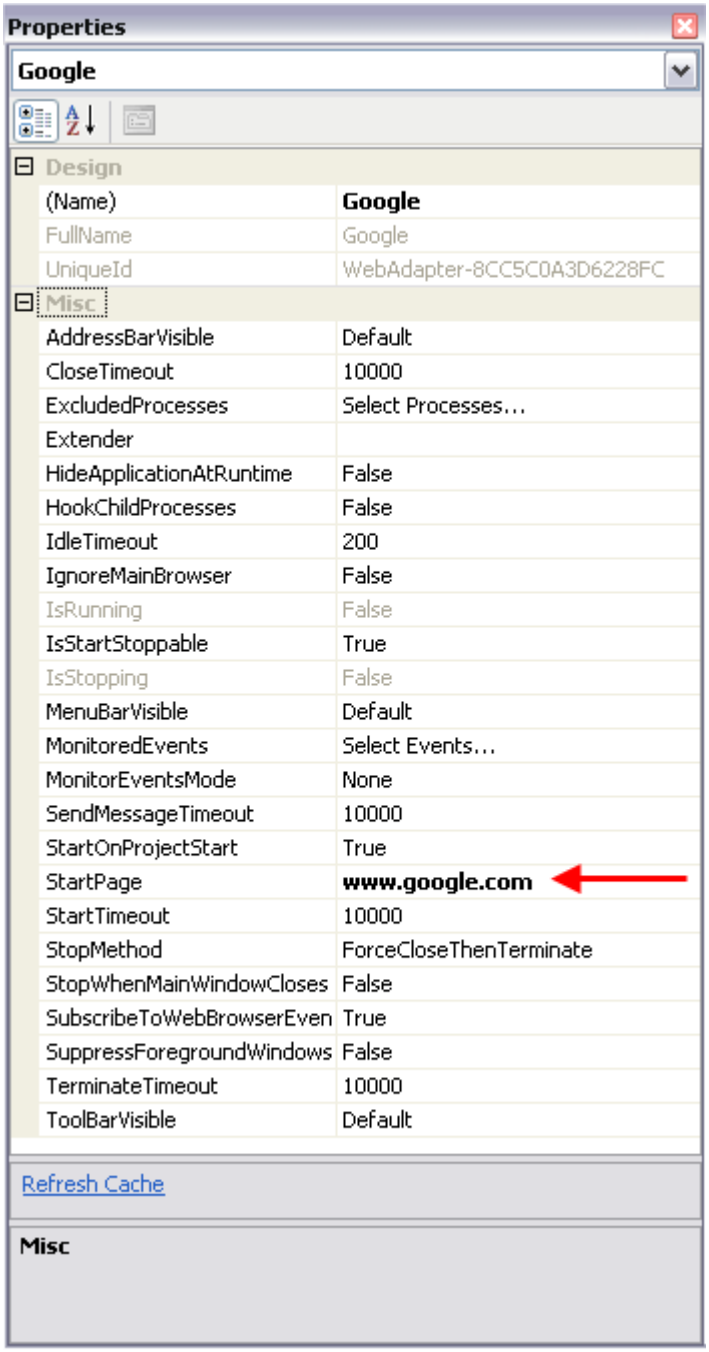


2. Select **View | Properties Window**

3. In the **Properties** window, **Google** should already be selected in the drop-down list at the very top of the window.



4. With the **Google** web application still selected in the **Properties** window, enter **www.google.com** in the **StartPage** field.

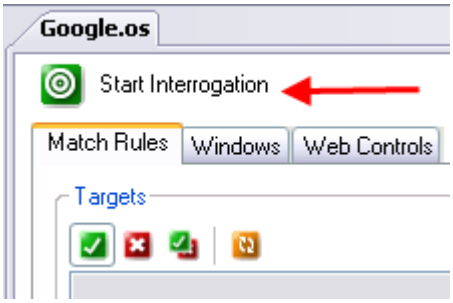


Note: By specifying the **StartPage** property, you are ensuring that the solution begins on the www.google.com web page each time. If a start page is not specified, then the solution will use the default start page set in Internet Explorer.

5. Select **File | Save Google.os**.

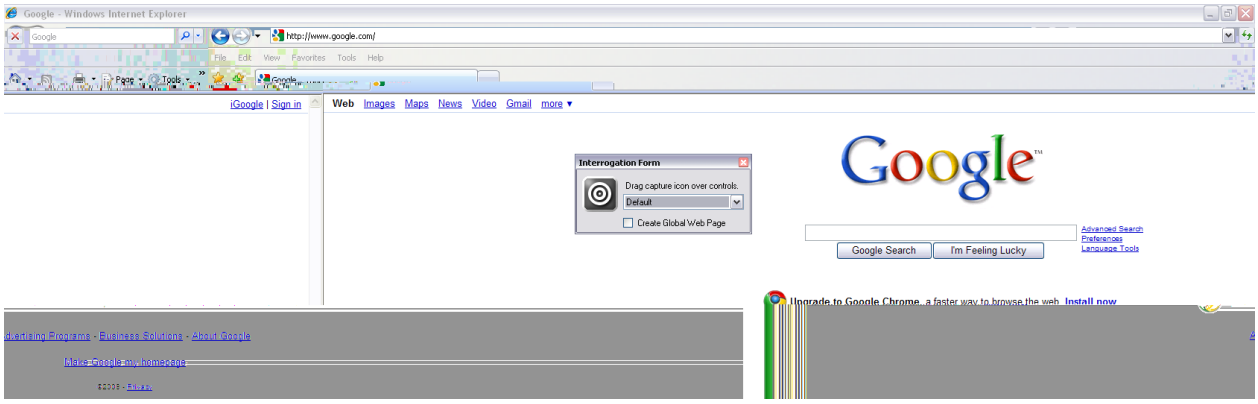
Interrogating the Web Application

1. Click the **Start Interrogation** button at the **top** of the **Google.os** design pane to launch the **Interrogation Form** dialog and the Google web page.



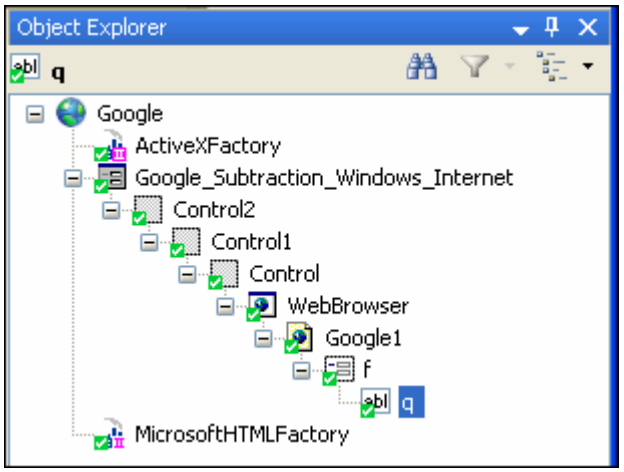
Note: The Interrogate function allows you to choose the controls/components on the web page that you want to use in the solution.

The **Interrogation Form** dialog should open on top of the Google web page.



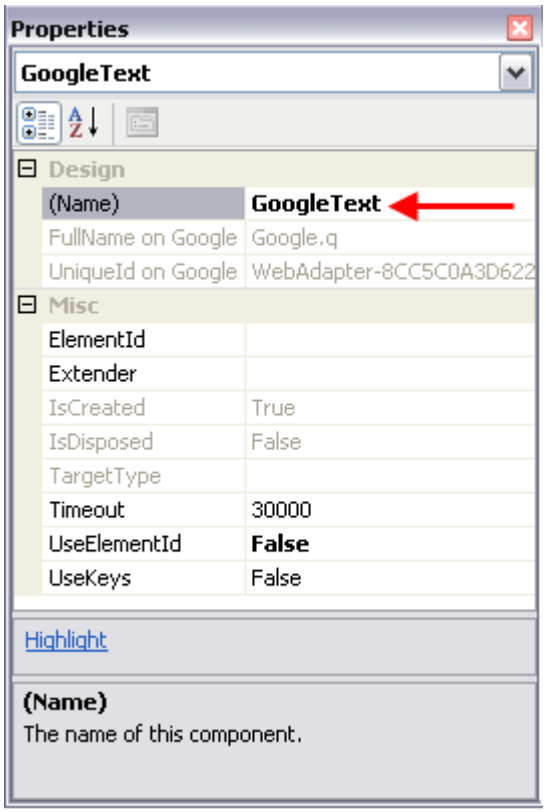
2. Click the bulls-eye shaped icon in the **Interrogation Form** dialog, hold the mouse button down, and drag the icon over the **Search** text box control on the Google window. When the **Search** text box is highlighted, release the mouse button.

   OpenSpan Studio uses Match Rules to define the control selected on the Google window and adds it as an item to the **Object Explorer** in the solution – named **q.**

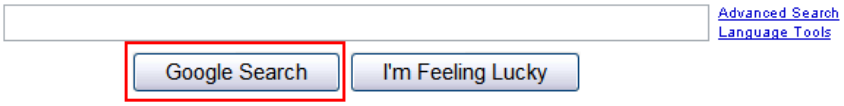


   Note: The controls which parent the Web Browser differ depending on the version of Internet Explorer used. Your Object Explorer may look different than the one shown above.

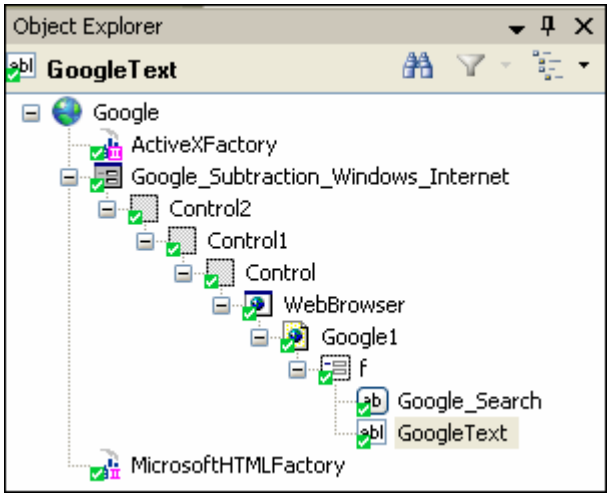
3. In the **Properties** window, rename the **q** project item to **GoogleText** by entering the new name in the **(Name)** field.



4. Continue the interrogation by dragging the bulls-eye icon over the **Google Search** button on the Google web page.



   Release the mouse button when the command button is highlighted. OpenSpan Studio uses Match Rules to define the area on the Google window and adds it as an item to the solution – named **Google_Search**.



5. Stop the interrogation by closing the **Google** web page, clicking the window title bar **x** command button on the **Interrogation Form** dialog, or clicking the **Stop Interrogation** button on the **Google.os** design pane.

   Note: The **Interrogation Form** dialog closes automatically when you close the web page.

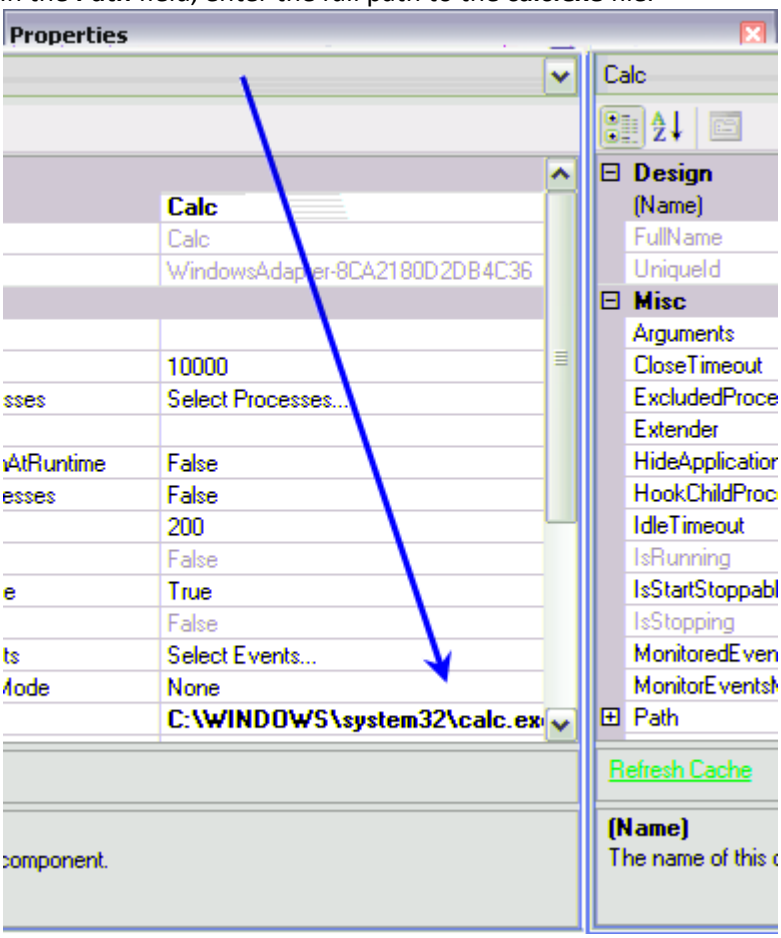
6. Select **File | Save Google.os**.

Setting properties for the Windows Application

1. In the OpenSpan Studio design pane, click the **Calc.os** tab to bring the **Calc** design pane to the front.

Note: If the design pane is not open (i.e., you don't see the **Calc.os** tab), then in the **Solution Explorer** window, **right-click** the **Calc** Windows application and select **Open** from the context menu.

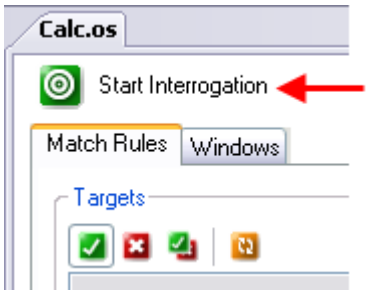
2. Select **View | Properties Window** to open the **Properties** window.
3. In the **Properties** drop-down list, select **Calc** to display its properties.
4. In the **Path** field, enter the full path to the **calc.exe** file.



5. Select **File | Save Calc.os**.

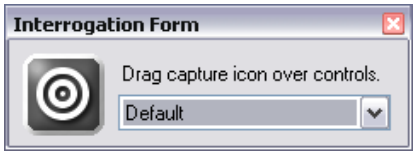
Interrogating the Windows Calculator Application

1. **Click** the **Start Interrogation** button at the top of the **Calc.os** design pane to launch the Interrogate function and the Windows Calculator.



Note: The Interrogate function allows you to choose the controls/components on the Calculator that you want to use in the solution.

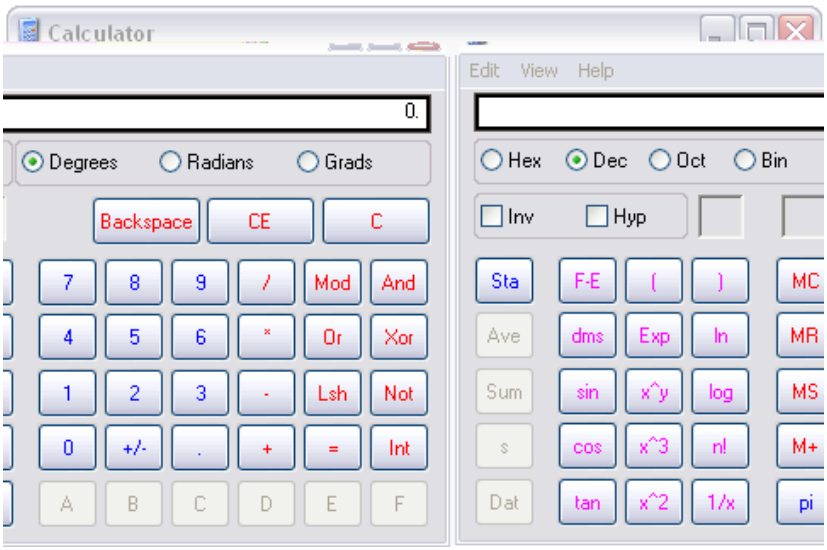
The **Interrogation Form** dialog opens on top of the Calculator. Ensure that the Calculator is set to **scientific** view by selecting **View | Scientific** in the Calculator program.



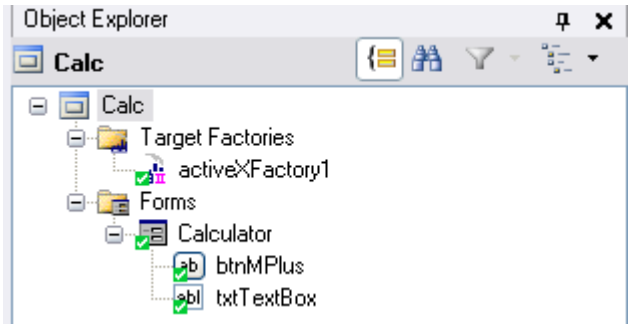
2. Click and hold-down the **bulls-eye** shaped icon in the **Interrogation Form** dialog and drag the icon over the **M+** button on the Calculator. When the **M+** button is highlighted, release the mouse button.

OpenSpan Studio defines the Calculator **M+** button and adds it as an object to the project – named **btnMPlus**.

3. Interrogate the **Results** box by clicking and holding-down the **bulls-eye** shaped icon on the **Interrogation Form** dialog and dragging the icon over the Calculator **Results** box. When the **Results** box is highlighted, release the mouse button.



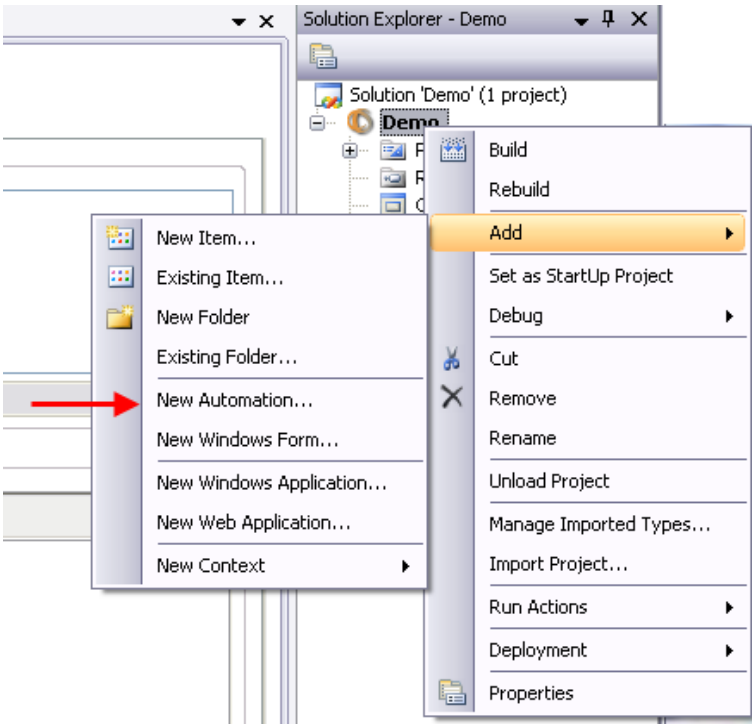
OpenSpan Studio defines the **Results** box and adds it as an object to the project – named **txtTextBox**.



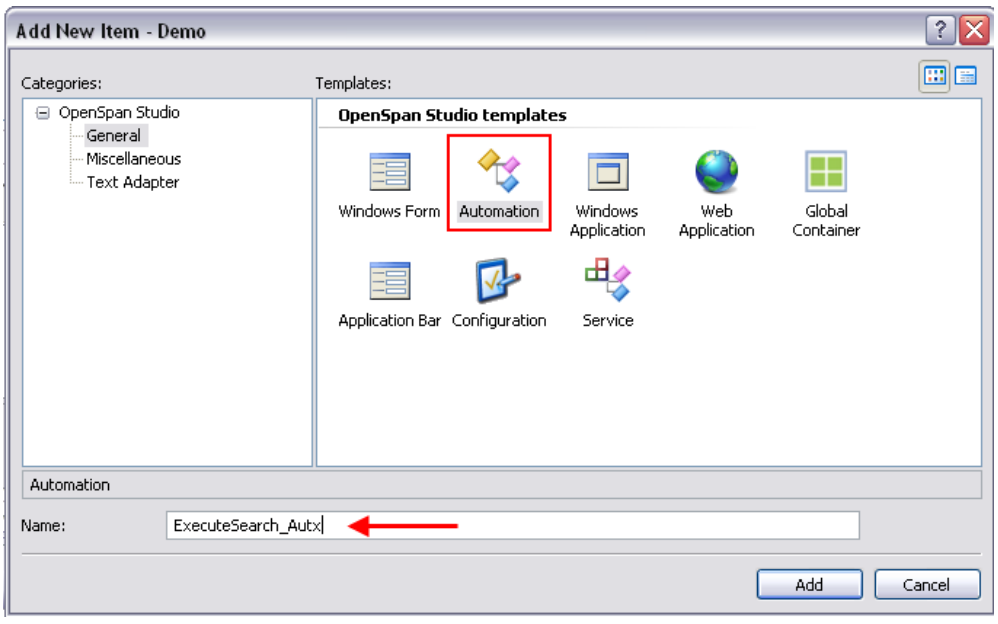
4. Stop the interrogation by closing the **Calculator,** clicking window title bar **x** command button on the **Interrogation Form** dialog, or clicking the **Stop Interrogation** button on the OpenSpan Studio design pane.
5. Select **File | Save Calc.os** to save the changes.

Adding an Automation to a Solution

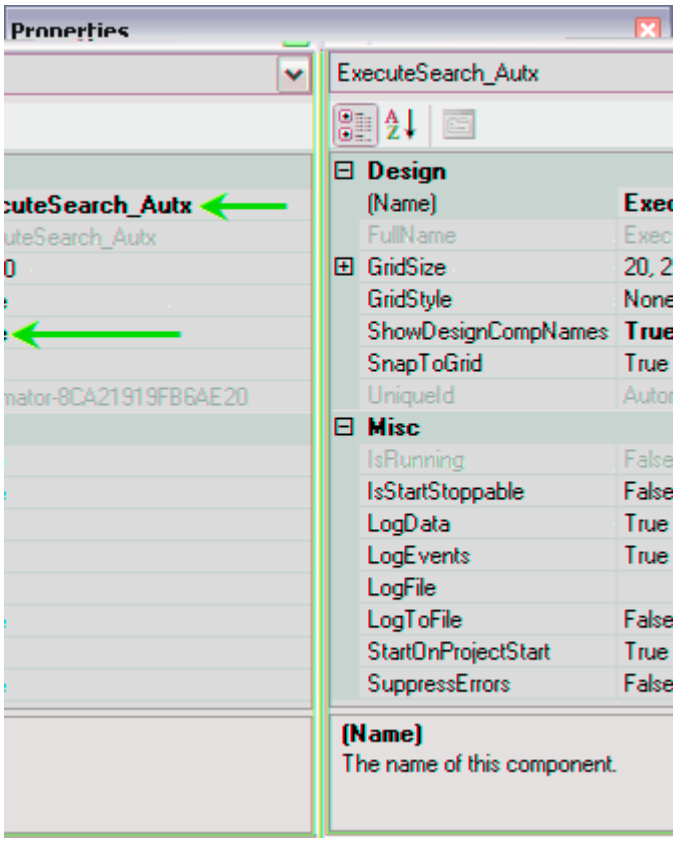
1. In the **Solution Explorer** window, **right-click** the **Demo** project and select **Add | New Automation** from the context menu.



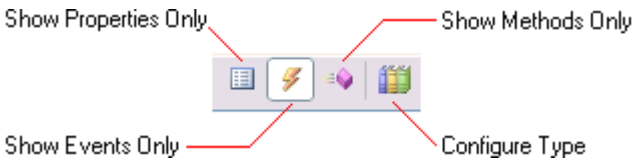
2. The **Add New Item** dialog opens. This dialog displays all the available Design Components that you can add to a solution. The Automation Icon is highlighted allowing you to type in the name of the automation. Rename **Automation1** to **ExecuteSearch_Autx** by entering the new name in the **(Name)** field, then select the **Add** button.



3. Select **View | Properties** to open the **Properties** window.
4. Change the **ShowDesignCompNames** property to True. Changing this property to True causes the Project Item name, such as **Calc** or **Google**, to display at the top of automation design blocks.

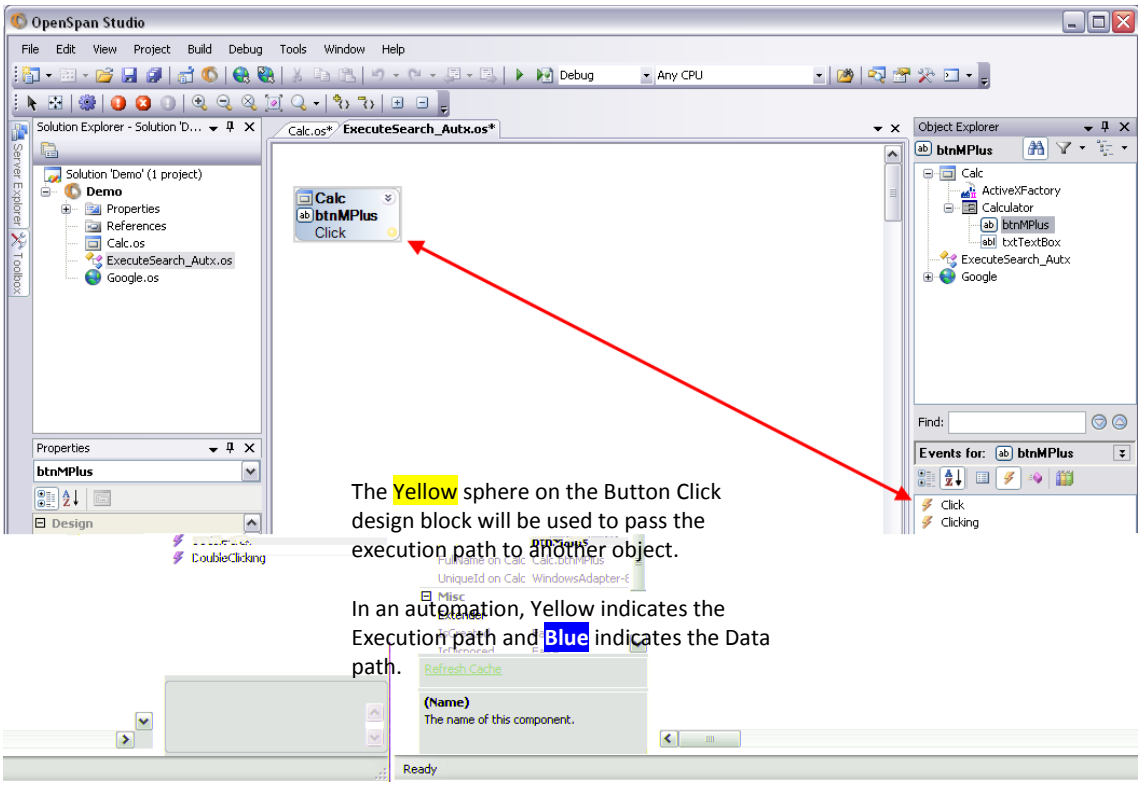


5. In the **Object Explorer** window, select the Calculator Button object – **btnMPlus** (The **M+** button on the Calculator).
6. In the lower pane of the **Object Explorer** window, display the Events for the **btnMPlus** button by clicking on the **Show Events Only** (Lightning Bolt) icon.



Note - An Event is a notification by a program or operating system that "something has happened." An event may be fired (or raised) in response to the occurrence of a pre-defined action (e.g., a window getting focus, a user clicking a button, a timer indicating a specific interval of time has passed, or a program starting up or shutting down).

7. Select the **Click** event and drag and drop this item to the upper left corner of the **ExecuteSearch_Autx** automation design pane.



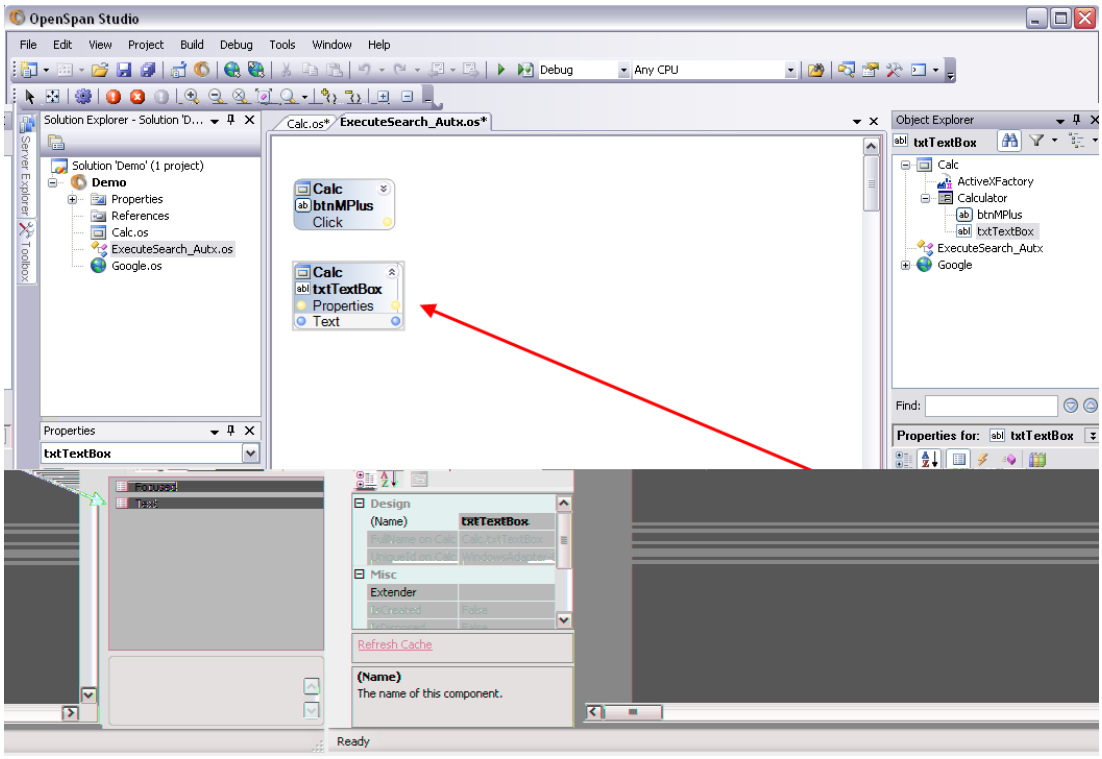
The **M+** button's Click event displays as a design block object on the automation design pane.

8. In the **Object Explorer** window, select the Calculator text box object - **txtTextBox** (The **Results** box on the Calculator).

9. In the lower pane of the **Object Explorer** window, display the Properties for the **txtTextBox** object by clicking on the **Show Properties Only** button  .

   Note – A Property is a Common Language Runtime (CLR) feature that allows the value of a single member variable to be modified using getter and setter methods defined in a class or structure.

10. In the **Properties for:** list, select the **Text** property and drag and drop it on the **ExecuteSearch_Autx** automation design pane - below the **btnMPlus** design block.



11. Repeat this process to bring the **PerformClick** Method  for the **Google_Search** button and the **Text** property  for **GoogleText** to the automation design pane.

Note – A Method is a function defined within a class. Methods (along with events) define the behavior of an object.

12. Select **File | Save ExecuteSearch_Autx.os**.

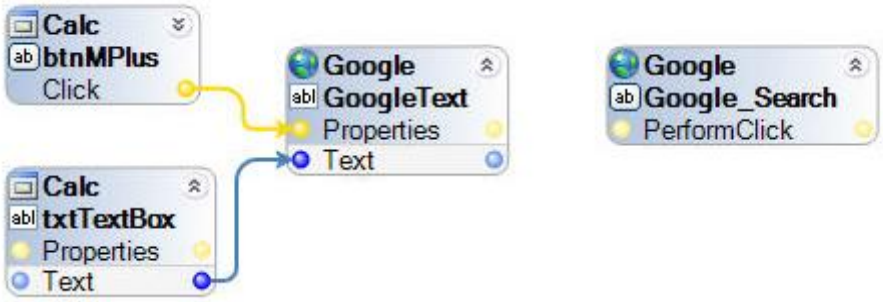
Connecting the Data and Execution Paths

At this point, all properties, methods and events for the controls in the solution are exposed for use. In the next step, you will connect the data and execution paths to complete the logic flow for the solution.

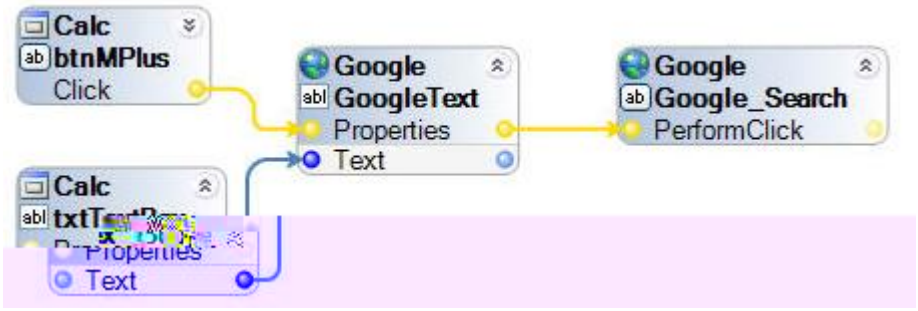
1. To connect the data path, draw a line (blue) from the blue data port on the right side (origination point) of the Calculator text box (**txtTxtBox**) object to the left blue data port for the Text property of the **GoogleText** object.

Note: Blue lines represent the data path, and **Yellow** lines represent the execution path. To draw lines between design blocks, click a colored dot on one object and drag to the target object and release.

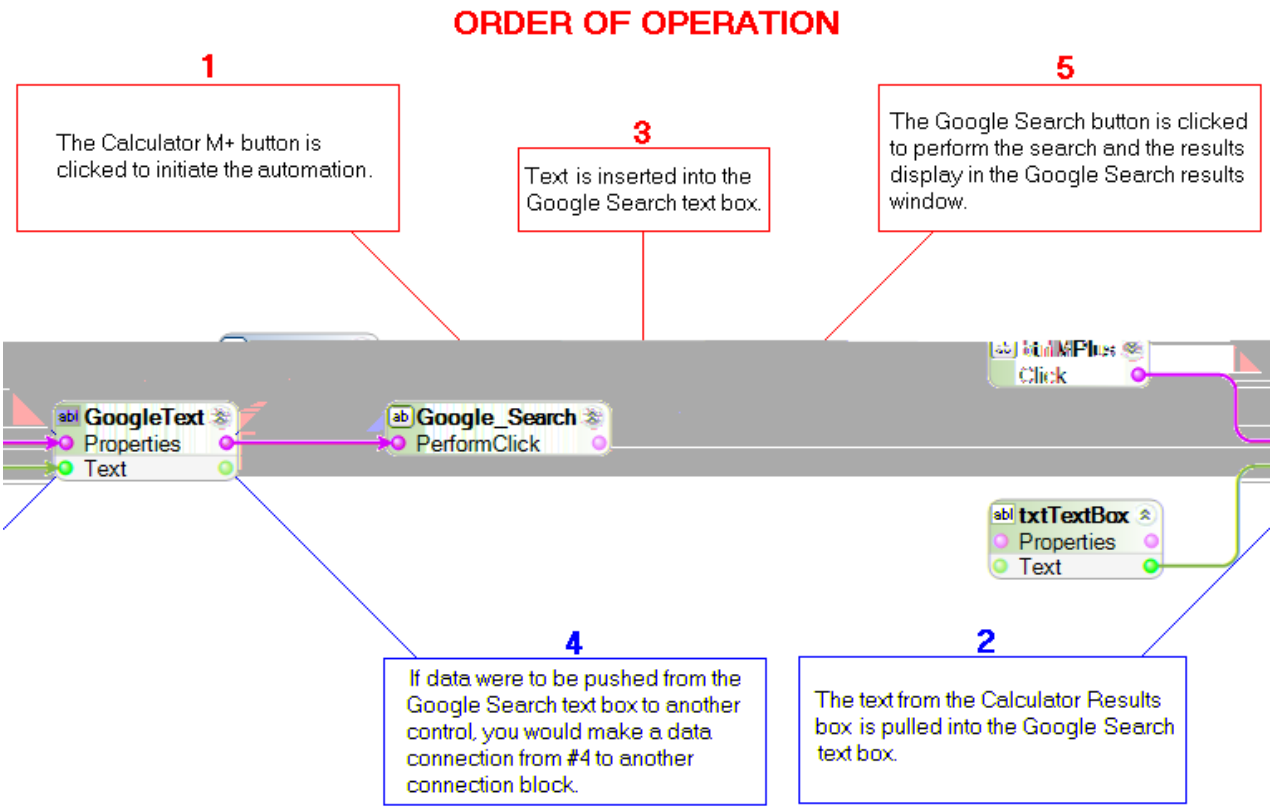
2. To connect the execution paths, draw a line (yellow) from the yellow execution port on the right side of the Calculator Button (**btnMPlus**) design block to the yellow execution port on the left side of the **GoogleText** design block. This triggers the Google text box to pull data.



3. Complete the execution path by connecting the Text Properties from the **GoogleText** design block to the PerformClick method of the **Google_Search** button. This will click the Google Search button and initiate the search.



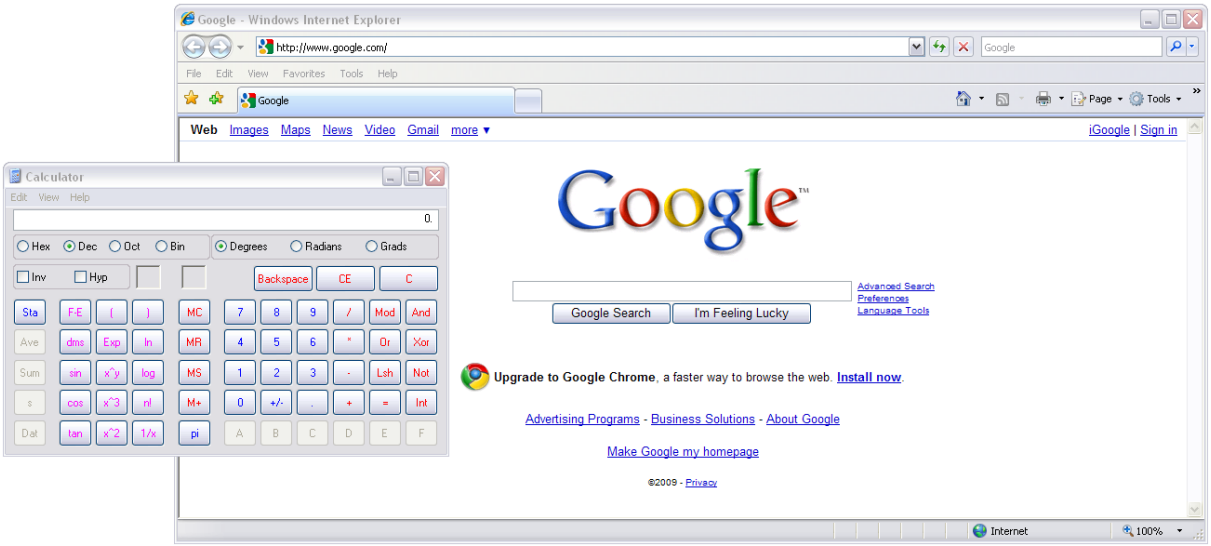
The example below illustrates the automation's order of operation:



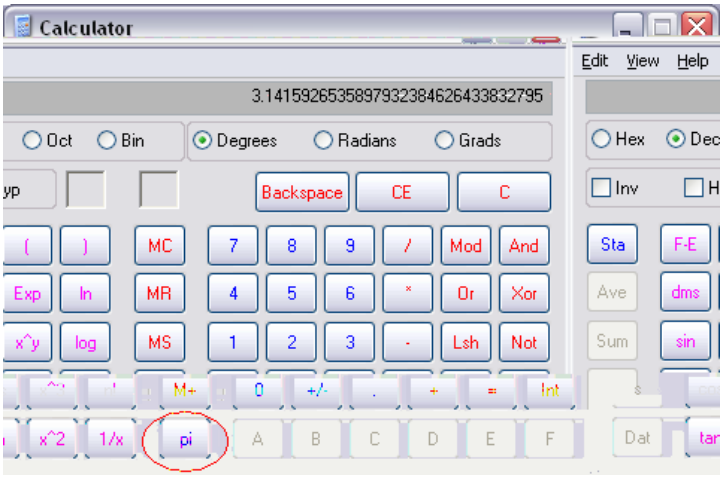
4. Select **File | Save ExecutionSearch_Autx.os**.

Running the Solution

1. Click the Start Debugging button  to start the project.
2. After both applications open confirm that you can use the applications as usual.

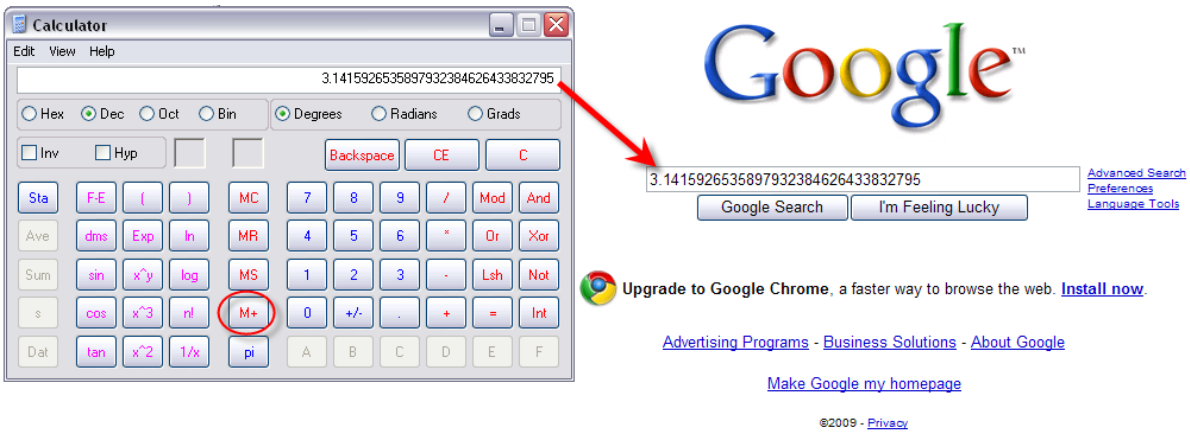


3. In the Calculator, press the **pi** button. The value **3.1415…** should display in the Calculator **Results** text box.

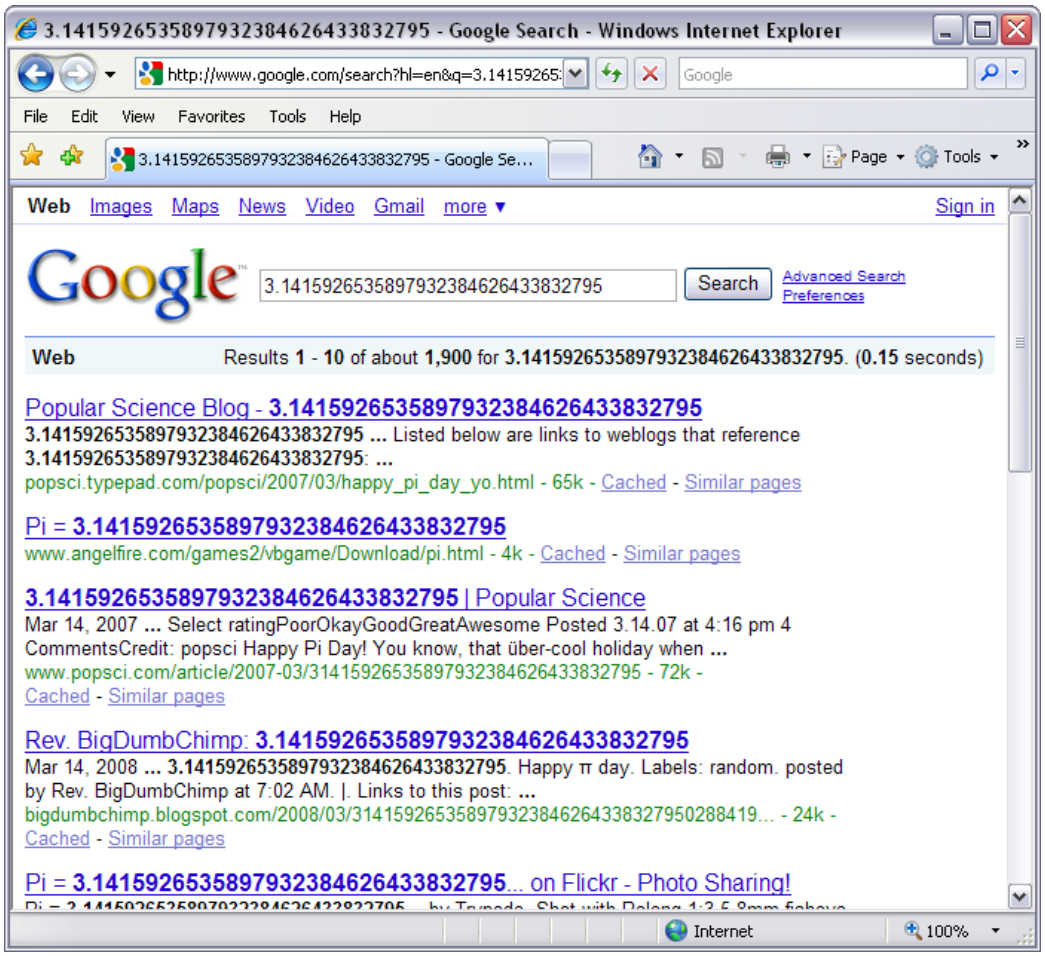


4. Press the **M+** button to pass the data to the Google textbox and initiate a search on the number.

The value is pushed from the Calculator Results text box to the Google text box on the Google web page:



Once the data has been passed to the Google text box, the Google Search button is automatically clicked, and the results of the search display in a Google search results window.



Note: If you attempt this process again by pressing the **M+** on the Calculator, the solution will fail. This is because we have specifically identified the main Google application window for receipt of the data and you are no longer on the page.

Exercise Summary

This exercise introduced you to some of the basics of the OpenSpan Studio design interface:

- Creating a new solution and project
- Adding project items
- Interrogating a web page
- Interrogating a Windows application
- Setting properties in the Properties window
- Renaming controls for design identification
- Saving and running solutions

On Your Own…

Now that you've completed your first OpenSpan solution (with the help of the training guide and class instructor) it's time to try and create a solution on your own.

You're going to start by recreating the same Calc/Google solution from the previous exercise, but this time you'll add functionality to make it a more complete and fail-safe solution.

Goal: Try to recreate the Calc/Google solution from memory. Only refer to the instructions or ask questions of the instructor if you get completely stuck.

1. Create a new blank solution and name it **Demo2**.
2. Interrogate both the **Calculator** Windows application and the **Google** Web page.
3. Using the **M+** button Click event to initiate the automation, recreate the Calc/Google automation from the previous exercise.
4. Once you've successfully recreated the original exercise, demonstrate the automation for the instructor before moving on to the next step.
5. From the **ExecuteSearch_Autx.os** design pane, add a method that will automatically check the web browser to see which URL it's on before attempting to pass data from the Calculator to the Google web page.

   Hint: If the web browser is not on the predefined start page (www.google.com) you'll need to navigate back to the correct URL address before you can proceed with the automation.

   Note: Adding this functionality will allow you to run the automation multiple times.
6. Demonstrate the additional functionality to the instructor.

CHAPTER 3: WORKING WITH THE WINDOWS ADAPTER

Integrating Windows Applications

OpenSpan Studio contains a special adapter for integrating Windows applications. This adapter is added as an item to a project. The adapter exposes the functionality of the selected application and enables the OpenSpan Studio Interrogator to isolate and uniquely identify the controls within the Windows application with which you want to interact.

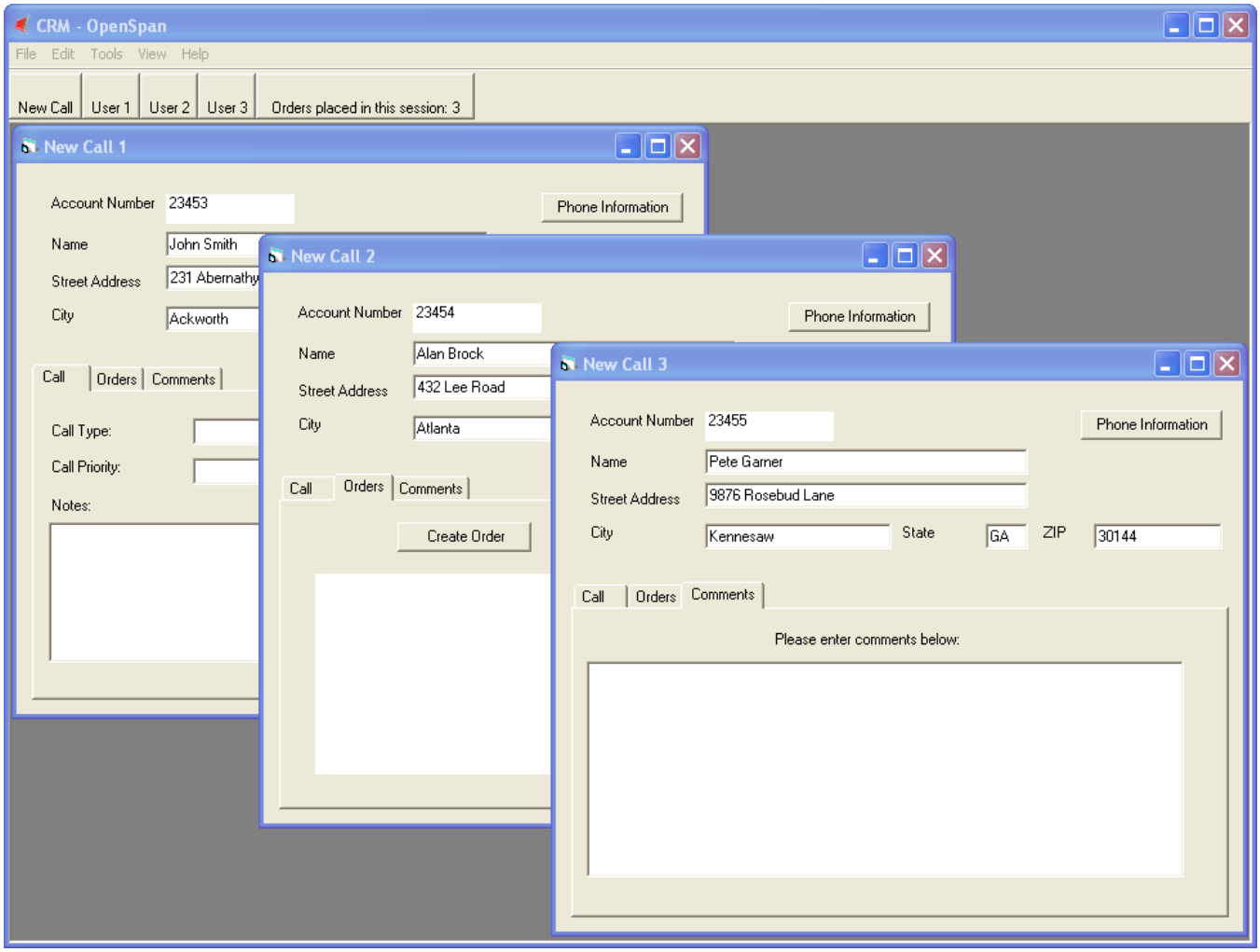
There are 6 steps to using a Windows application in an OpenSpan Studio solution:

1. Creating a Solution and Project.
2. Adding a Windows adapter to the project.
3. Defining the Windows adapter properties to identify the application with which you wish to work.
4. Interrogating the Windows application to identify the application controls that are going to be used in the project.
5. Adding any required data and automations to create the work process flow desired.
6. Saving, deploying and running the solution.

An issue that arises with Windows applications is the identification of controls on Multiple-document Interface (MDI) child windows. OpenSpan Studio has custom properties, methods and events for dealing with the special circumstances surrounding automating Windows applications with MDI child windows. More information about working with MDI Child properties and windows is presented later in this chapter.

The following exercises demonstrate the key concepts you need to know in order to integrate a Windows application into an OpenSpan Studio solution. These exercises make use of a simple Windows application, CRM.exe, and the Application Bar that you will create in the next exercise.

Example of the CRM Application:



The CRM application contains five toolbar buttons. The **New Call** and **User** buttons launch a window containing fields for capturing customer information (account number, name, and address). If the **New Call** button is selected, a window with blank fields is launched and the Call tab is on top. The **Orders** tab simply changes the colors on a text box, and the **Comments** tab contains a text box of the same name. Note that the account number is automatically generated, unique, and cannot be edited. The goal of this exercise is to demonstrate OpenSpan Studio's bidirectional integration with Windows applications.

Using an Application Bar that you will design, the following functionality will be enabled by the project you will create in this exercise:

- Open CRM application and automatically launch the **New Call** window
- Populate the textboxes on the Application Bar with account data from the CRM window

Installing the CRM Application

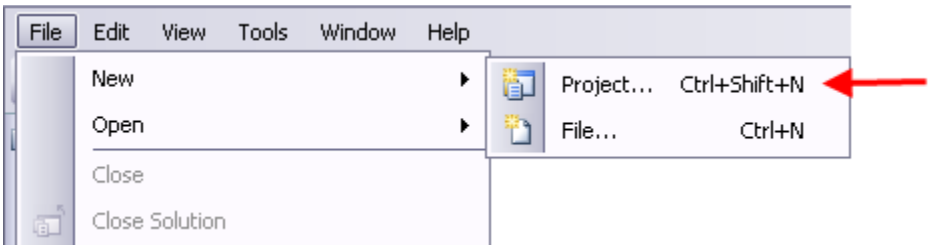
Before beginning, you need to install the CRM application.

1. Navigate to C:\Program Files\OpenSpan\OpenSpan Studio for Visual Studio 2008\Extras\
2. **Open (double-click)** the file **CRM Setup.msi.**
3. Allow the application to install in the default folder, which should be **C:\Program Files\OpenSpan\CRM Setup**.

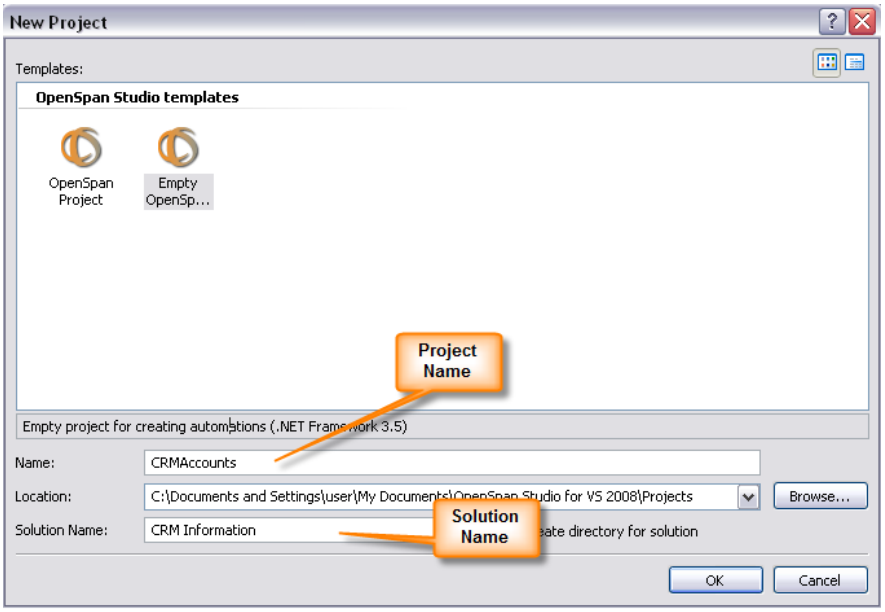
Exercise 1 - Building an Application Bar

This exercise provides an introduction to using the OpenSpan Studio design interface.

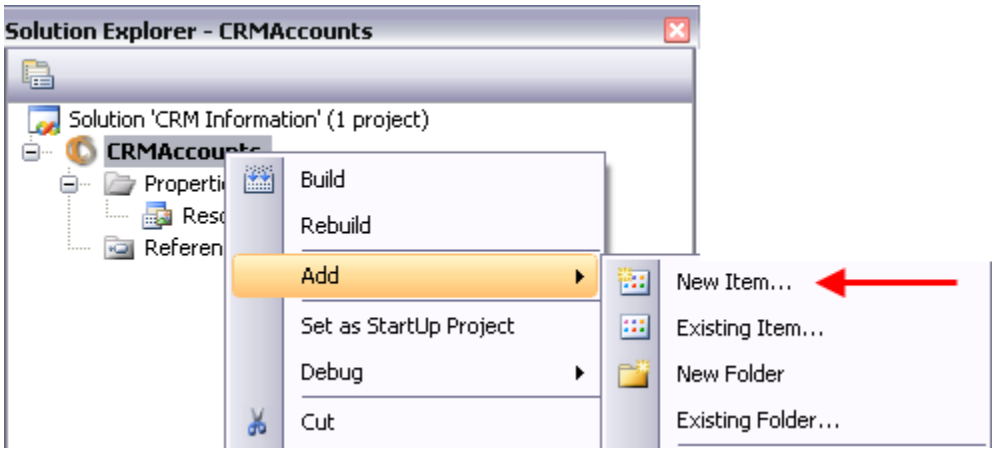
1. Start OpenSpan Studio.
2. Select **File | New |Project**.



3. Select Empty OpenSpan Project and type **CRMAccounts** in the Name: Field and CRM Information in the Solution Name field. Enter a folder location or accept the default, and then select **Ok**.



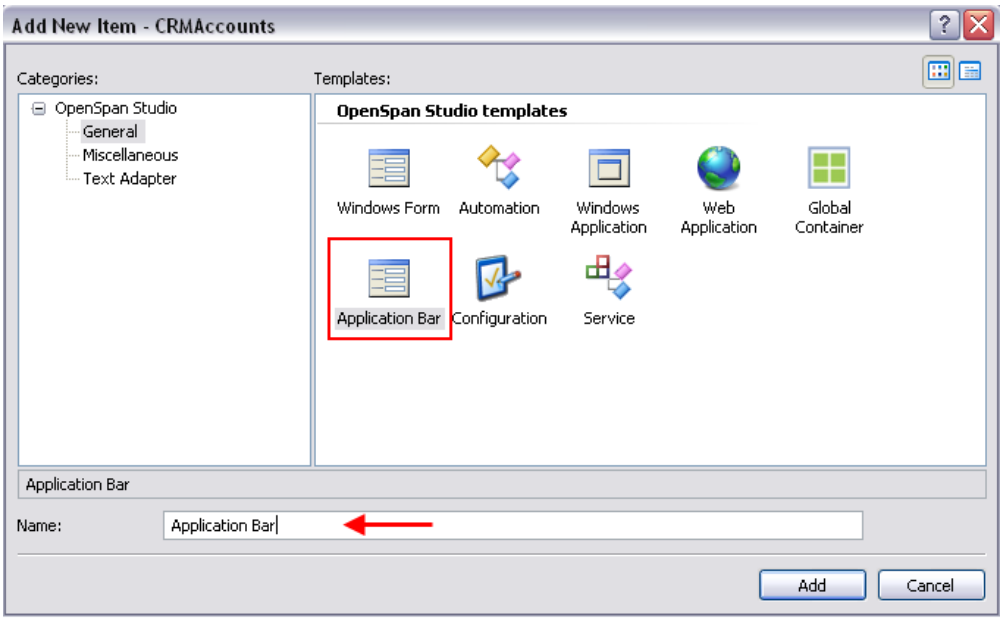
4. In the **Solution Explorer** window, **right-click** the **CRMAccounts** project and select **Add | New Item** from the context menu.



The **Add New Item** dialog Opens. This dialog displays all the available design components that you can add to a Solution.

5. Ensure the **General** category is selected, and then click the **Application Bar** type. Type **Application Bar** in the **Name** field, and then select **Add**.

The Application Bar.os design pane displays in the OpenSpan Studio window.

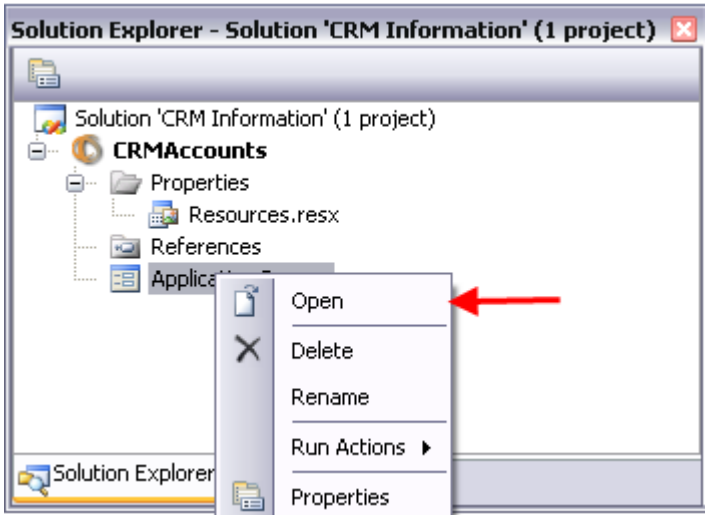


6. Select **File | Save Application Bar.os**.

Adding Controls to the Application Bar

This section of the exercise shows you how to add controls to the application bar, rename them, and set their properties using the **Toolbox.** The **Toolbox** contains all available .Net controls and the OpenSpan Studio controls.

1. In the **Solution Explorer** window, **right-click** the **Application Bar** item and select **Open** from the context menu, if the application bar is not already open.

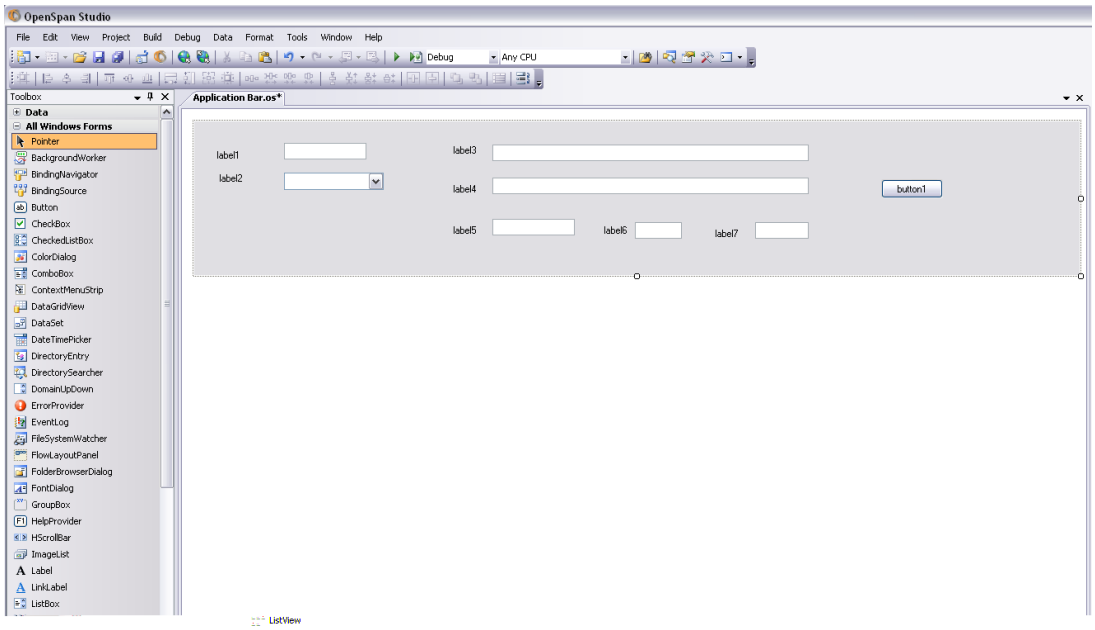


   The **Application Bar.os** design pane displays in the OpenSpan Studio window.

2. Select **View | Toolbox.**
3. In the **Toolbox** window, expand the **All Windows Forms** section, and then drag and drop the following controls on to the **Application Bar** design pane.

Quantity	Control Type
7	Label
6	TextBox
1	ComboBox
1	Button

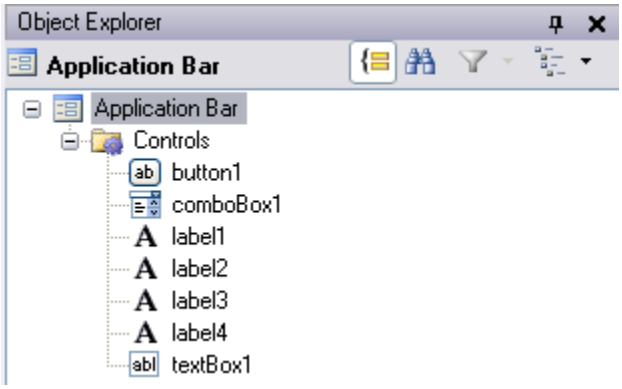
4. Position the controls on the **Application Bar** design pane as follows:



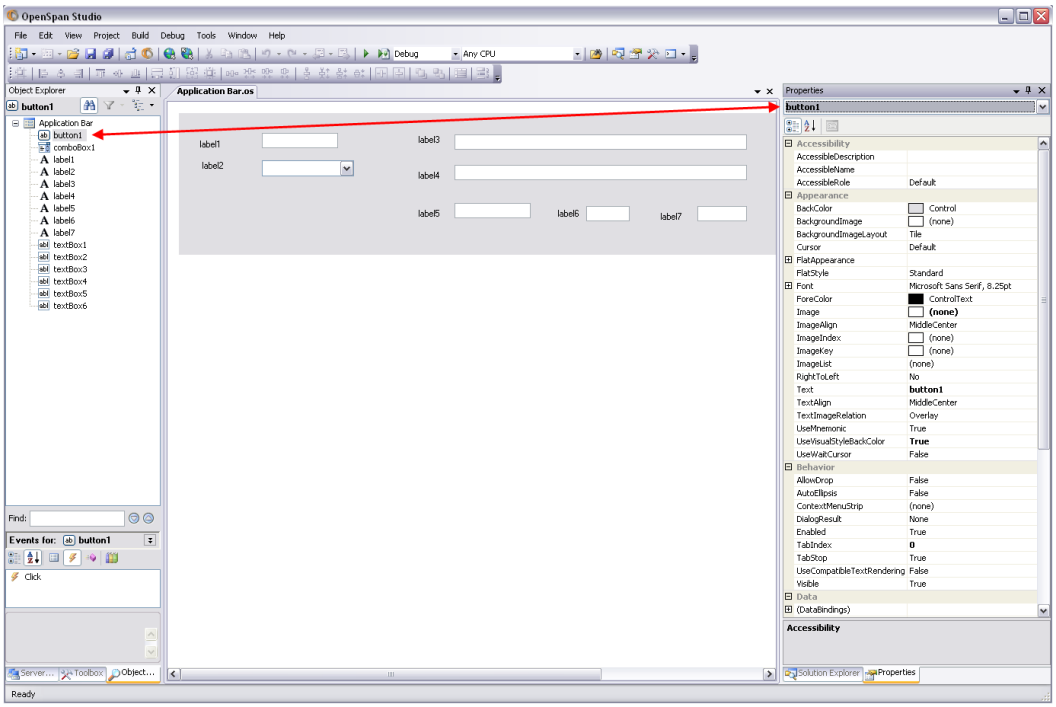
5. Select **File | Save Application Bar.os**.

Working with Control Properties

From the Object Explorer, you can see each of the controls you added to the Application Bar. To make it easier to work with the controls, you should rename each of the controls to something more descriptive of their function. In addition, you can also set various properties for each control that will change its appearance or behavior.



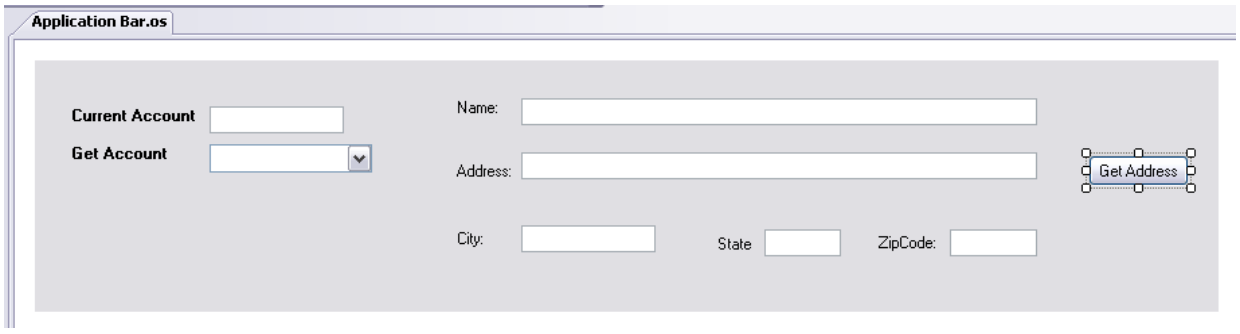
1. In the **Object Explorer** window, select the **button1** control.
2. Select **View | Properties Window**. The **button1** control will be automatically selected in the **Properties** window because it is already selected in the **Object Explorer**.



3. In the **Properties** window, set each control's name and properties as defined in the table below:

Control/Object	New Name	Additional Properties
Label1	**lblCurrentAcct**	Text – "Current Account" Font- Bold
Label2	**lblGetAcct**	Text – "Get Account" Font- Bold
Label3	**lblName**	Text – "Name:"
Label4	**lblAddress**	Text – "Address:"
Label5	**lblCity**	Text – "City:"
Label6	**lblState**	Text – "State:"
Label7	**lblZipCode**	Text – "ZipCode:"
TextBox1	**txtCurrentAcct**	Text – "--------"
TextBox2	**txtName**	Text – "" (blank)
TextBox3	**txtAddress**	Text – "" (blank)
TextBox4	**txtCity**	Text – "" (blank)
TextBox5	**txtState**	Text – "" (blank)
TextBox6	**txtZipCode**	Text – "" (blank)
ComboBox1	**cmbGetAcct**	Text – "" (blank) MaxDropDownItems – 100 Sorted - True
Button1	**btnGetAddr**	Text – "Get Address"

The completed application bar should look similar to this:



4. Select **File | Save Application Bar.os.**

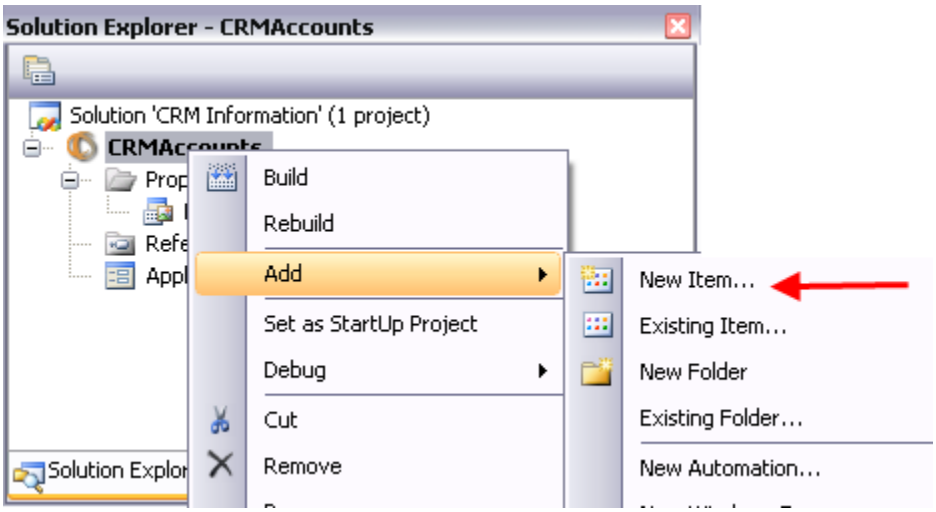
Exercise Summary

This exercise introduced you to some of the basics of the OpenSpan Studio design interface:

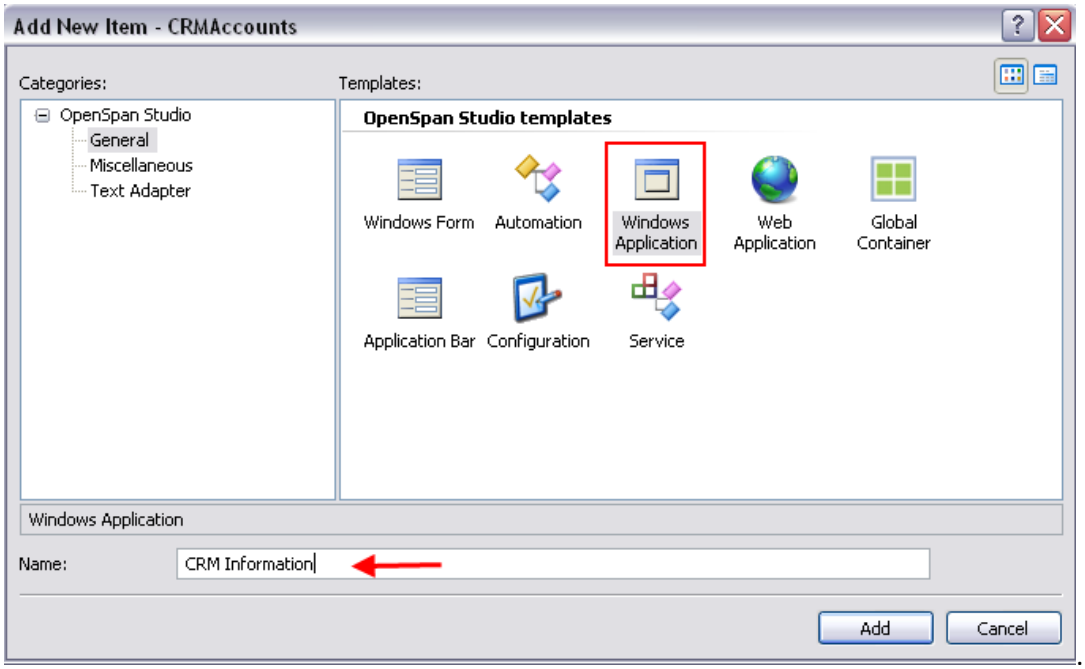
- Building a Solution using an Application Bar
- Adding an Application Bar design pane
- Adding Controls to an Application Bar
- Working with Control Properties

Exercise 2 - Adding a Windows Application to a Project

1. In the **Solution Explorer** window, **right-click** the **CRMAccounts** project and select **Add | New Item** from the context menu.

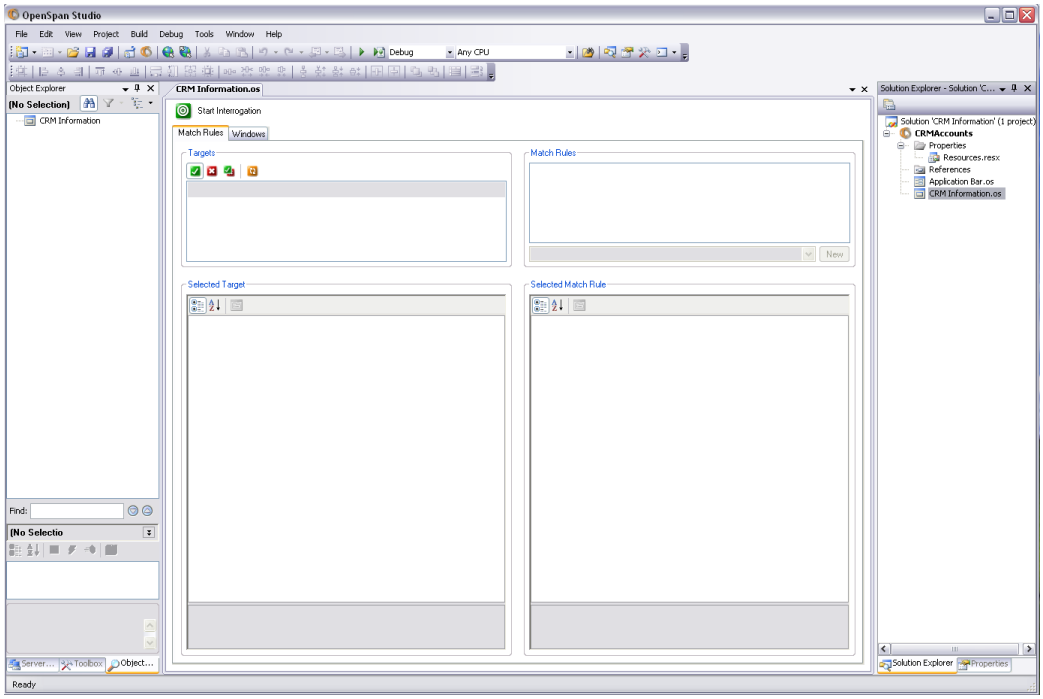


The **Add New Item** dialog displays:



2. Select the **General** category and the **Windows Application** template.
3. Type **CRM Information** in the **Name** field, and then select **Add**

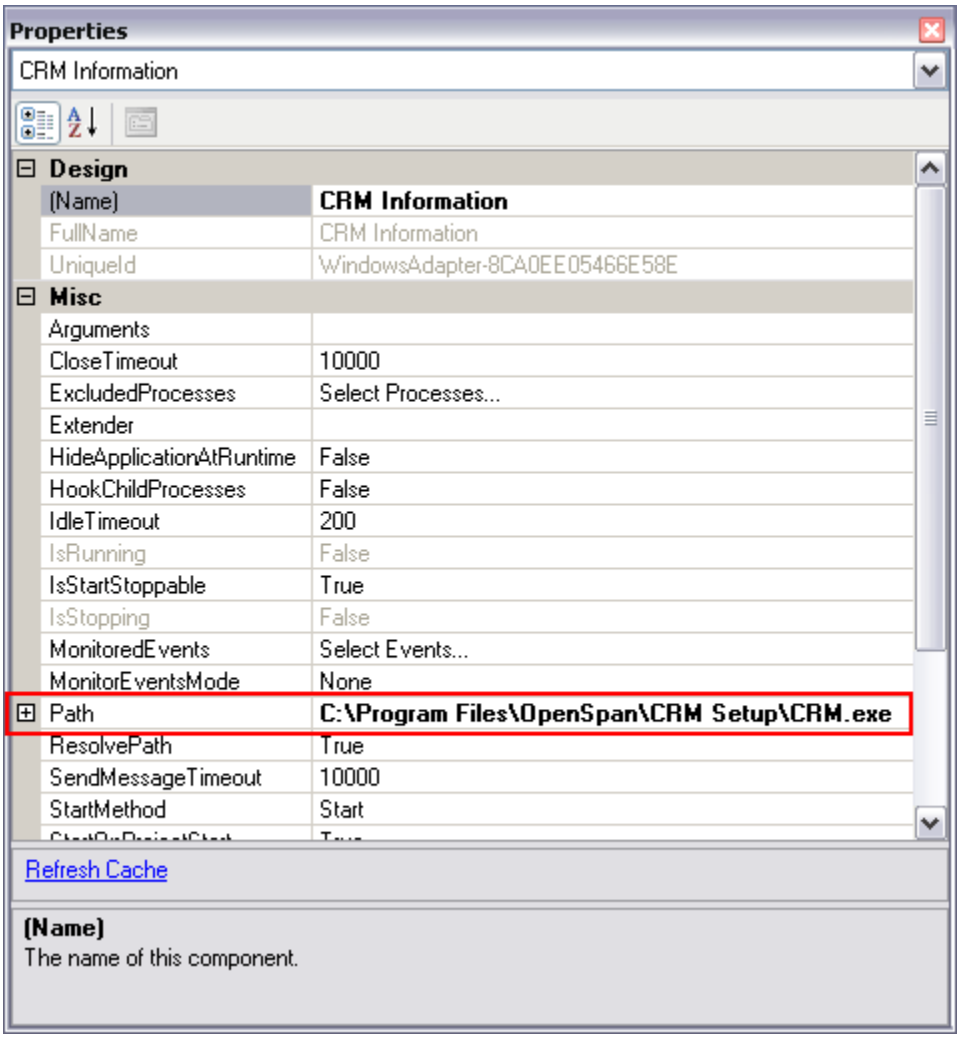
The main OpenSpan Studio window should look like this:



4. In the **Properties** window for the CRM Information Windows application, enter the path to the location where you installed the CRM Information application earlier in this chapter.

   If you used the suggested default folder, then the full path, including the name of the executable file name, should be **C:\Program Files\OpenSpan\CRM Setup\CRM.exe.**

   Note: To browse to the application location, click the ... (browse) button and select the application's path and executable file (CRM.exe).

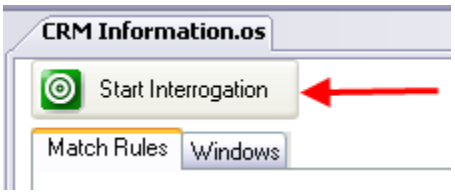


   By setting the **Path** property, the Windows adapter knows where to find the application.

   Note: If the application is moved from this location, the solution will fail.

Interrogating the Windows application

1. Click the Start Interrogation button at the top of **CRM Information.os** design pane to launch the **Interrogation Form** dialog and the CRM Information application.

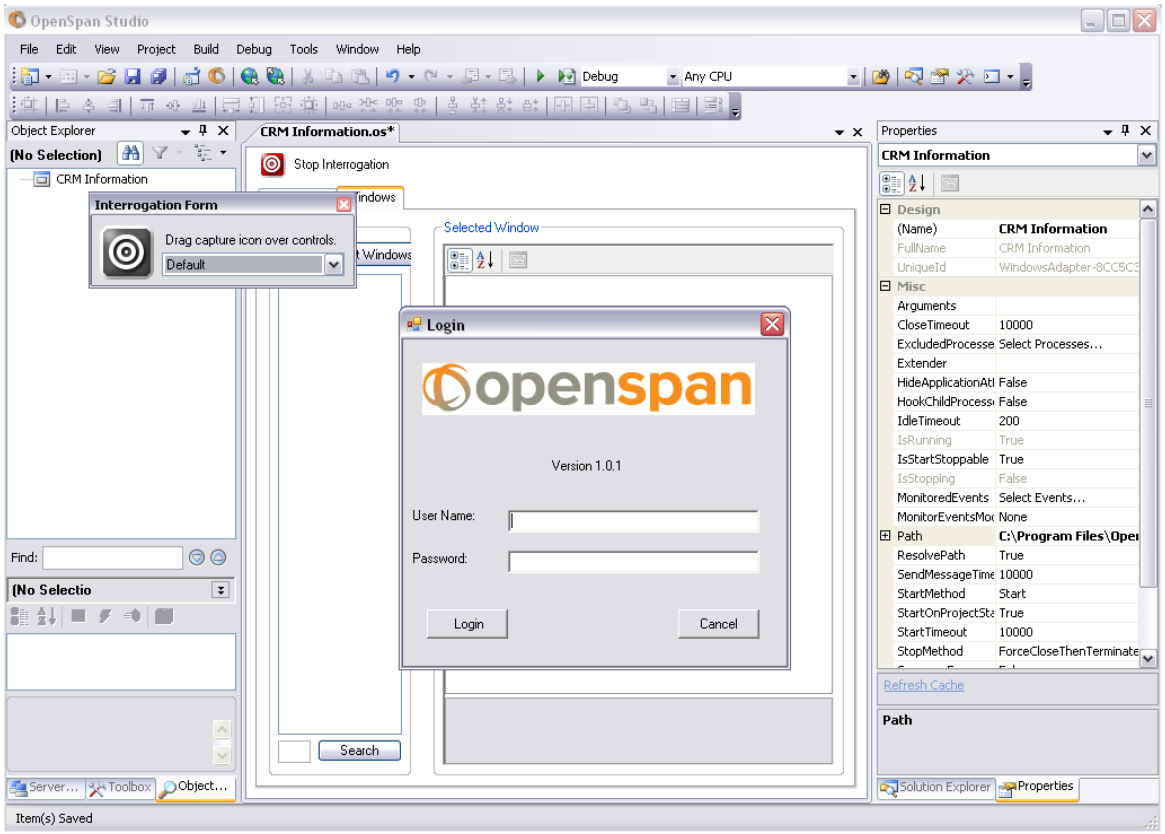


Interrogate Function

The Interrogate function is the primary function on the design tab for a Windows application. This function enables you to interact with a running version of the application through OpenSpan Studio, and select the controls in the application that you want to include in the solution. In order for an object to be included in a solution, the object must be matched through a sequence of *Match Rules*.

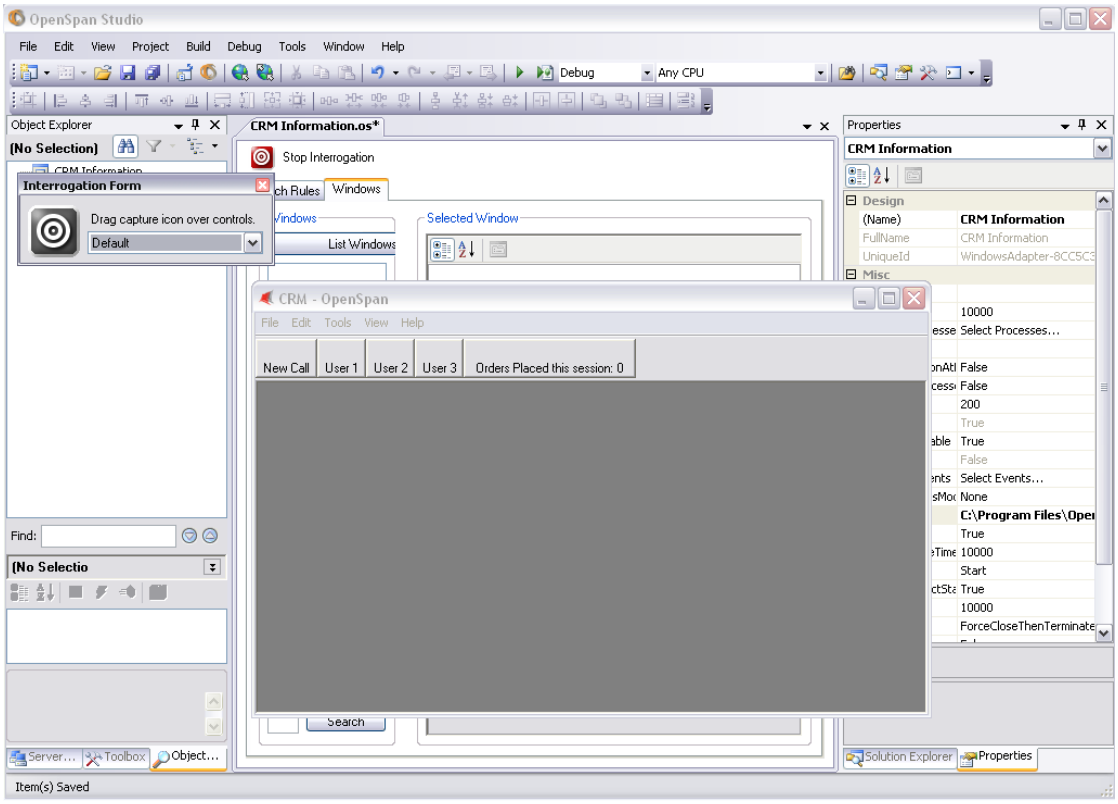
These match rules are based on the properties of the selected item, such as the item's class name, create sequence, or window style. Since many controls on a window or dialog can share similar attributes, multiple rules are usually required to uniquely match a single object.

The **Interrogation Form** dialog and the **CRM** application **Login** screen open on top of the OpenSpan Studio.



Select the **Login** button in the CRM application **Login** screen and the **CRM** window opens.

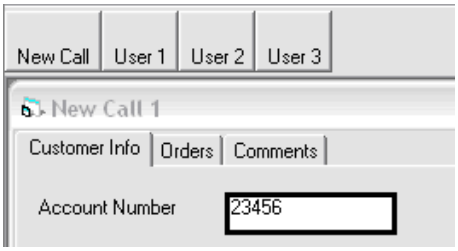
Note: It is not necessary to enter a **User Name** or **Password** in the **Login** screen.



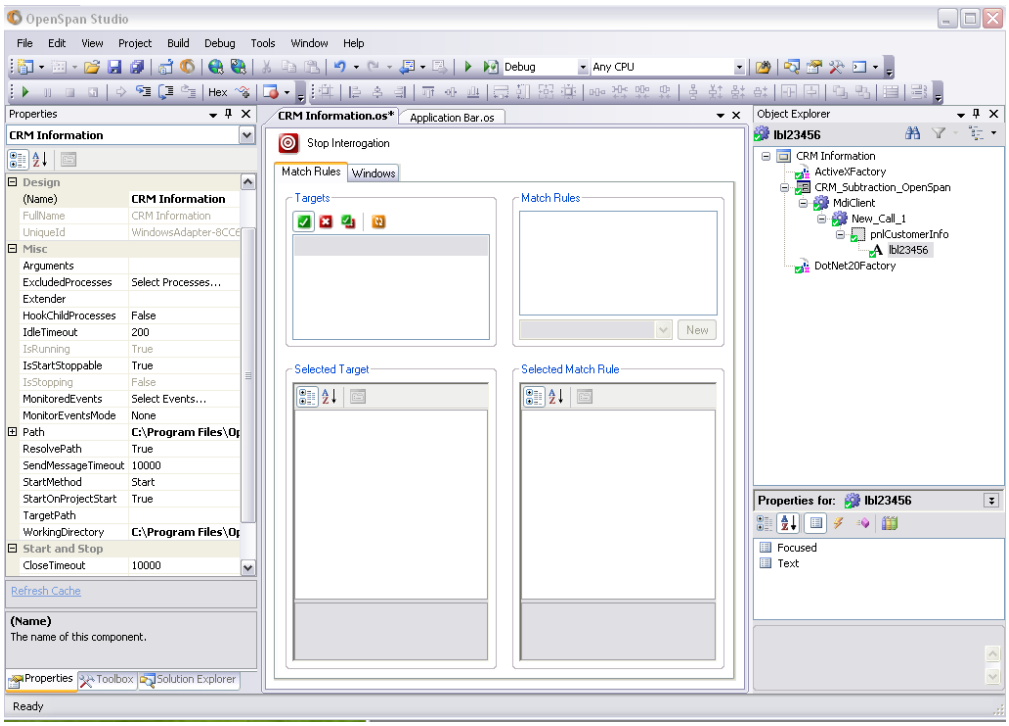
2. In the **CRM - OpenSpan** window, select the **New Call** button to open a new window.

   To select targets for interrogation, click the **bulls-eye** icon on the **Interrogator Form** dialog and drag it to the object on the CRM application. When a rectangle highlights the object that you want to include, release the mouse button.

3. Click and drag the bulls-eye icon over the **Account Number** text box and release when the field has a rectangular highlight around it.



The target object displays in the **Object Explorer**:

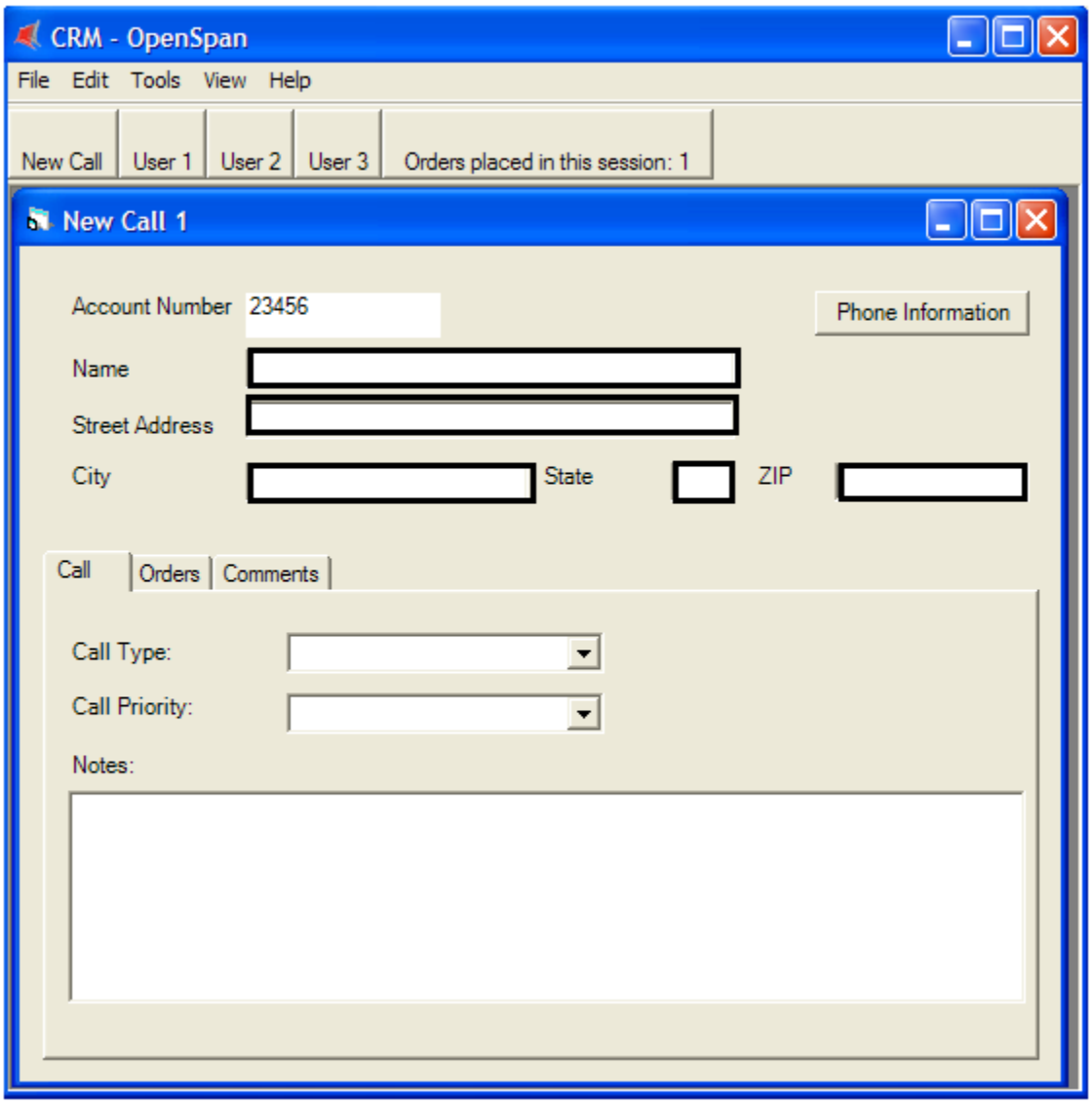


Note: Even though only the field was selected during the Interrogation process, all of the parent controls are automatically identified and matched by OpenSpan Studio. In this example, the following controls are identified:

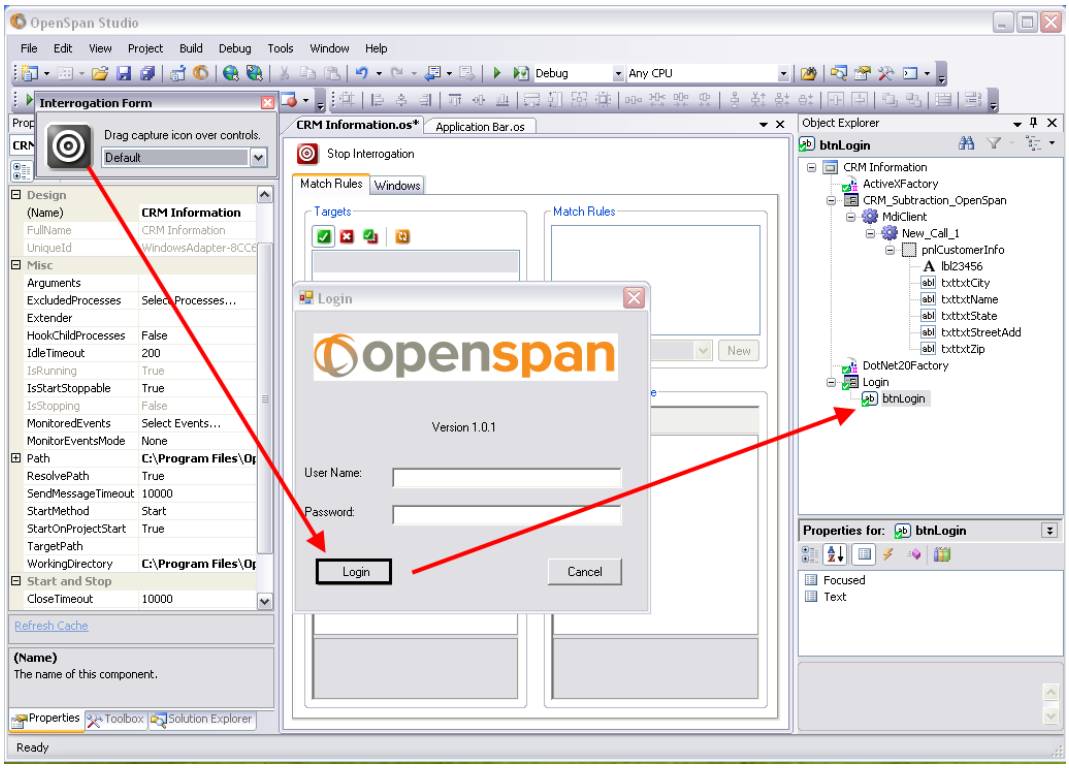
- **CRM**
- **ActivexFactory**
- **CRM_Subtraction_OpenSpan**
- **MdiClient**
- **New_Call_1**
- **pnlCustomerInfo**
- **lbl23456**
- **dotnet20Factory**

All of these controls are matched as indicated by the green arrow subscript.

4. Repeat this process for the following text boxes:
    - **Name**
    - **Street Address**
    - **City**
    - **State**
    - **ZIP**



5. The next control to integrate will be the login button from the CRM login screen. Stop the current interrogation which will close the CRM application. Select the Start Interrogation button to launch the CRM application and interrogate the Login button.



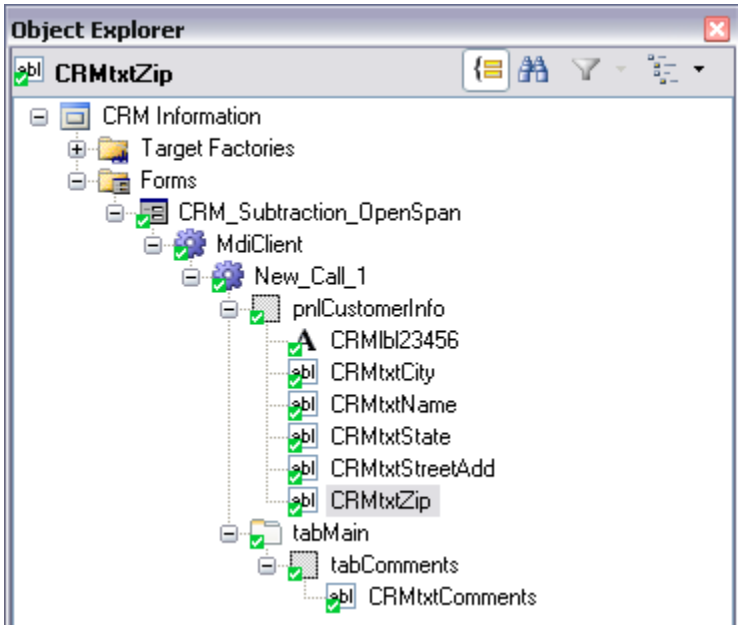
6. Select View | Properties Window.

7. In the Properties window, rename each of the interrogated objects:

Existing Name	New Name
lbl23456	CRMlbl23456
txttxtName	CRMtxtName
txttxtStreetAdd	CRMtxtStreetAdd
txttxtCity	CRMtxtCity
txttxtState	CRMtxtState
txttxtZip	CRMtxtZip
btnLogin	CRMbtnLogin

8. Select the Login button to open the CRM application. Select New Call button to open a call. In the New Call 1 window, select the Comments tab and interrogate the Comments text box. Rename the txttxtComment control to CRMtxtComments.

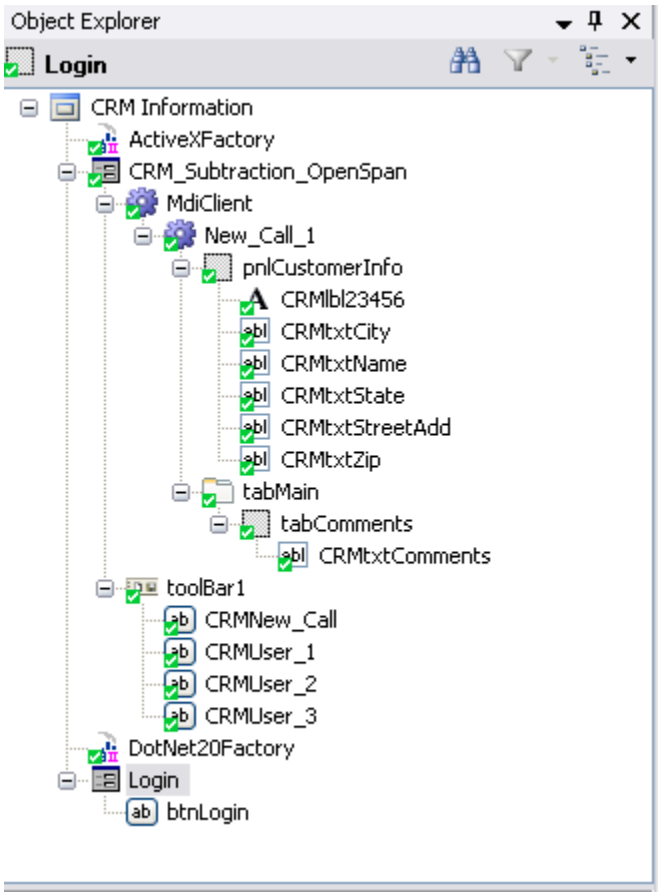
   When this object is returned to OpenSpan Studio, a new Windows control is added for **tabComments** in the hierarchy as follows:



9. The last objects to include in the solution are the toolbar and the associated buttons. Interrogate the following buttons:

   - **New Call**
   - **User 1**
   - **User 2**
   - **User 3**

10. When this object is returned to OpenSpan Studio, a button object is added under the toolBar1 object in the hierarchy for each button you interrogate. Rename the buttons CRMNew_Call, CRMUser_1, and so on. Your Object Explorer should look similar to the following:

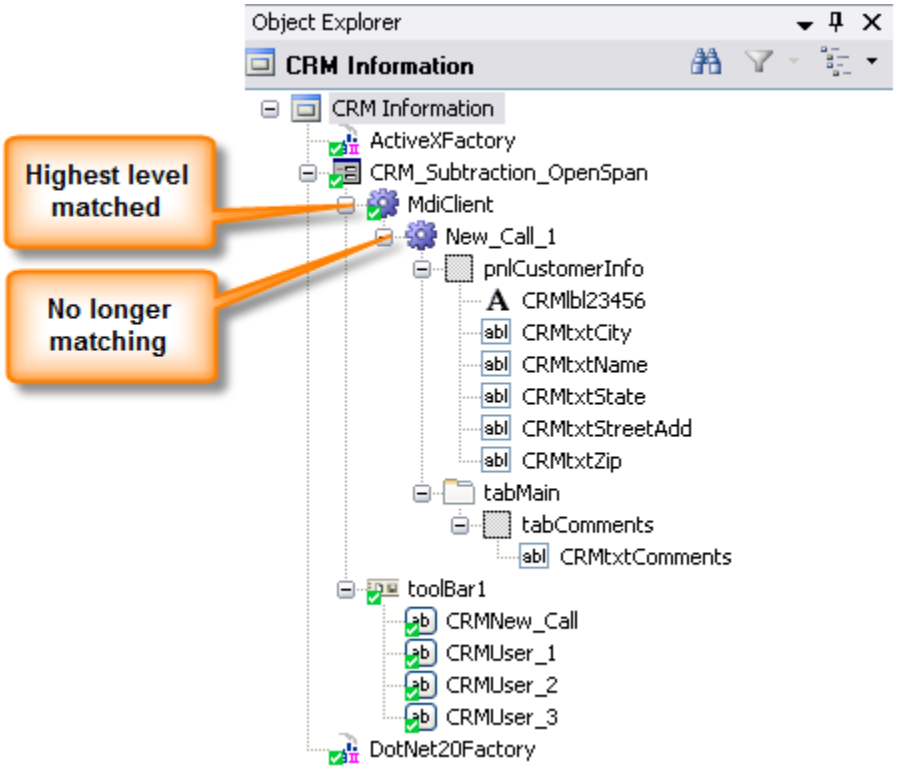


11. Select File | Save CRM Information.os.

Understanding Match Rules

1. Leave the **CRM** application and **Interrogation Form** dialog open, but close the **New Call 1** window.

   Note: Not all the controls match – the highest level that matches is the **MdiClient**. The green check next to a control indicates that it matches.

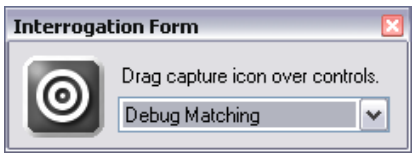


2. In the **CRM** application, open a new window by clicking the **New Call** button.

   Note: The rules remain unmatched.

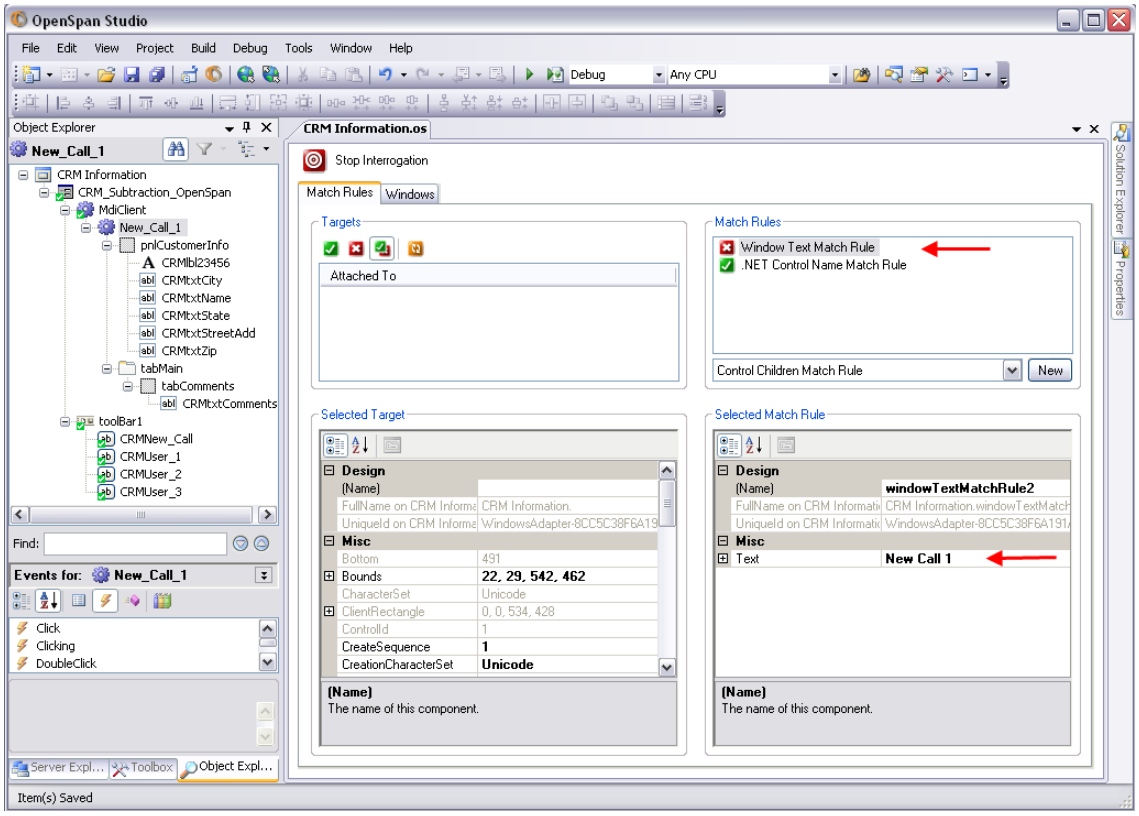
3. In the **Object Explorer**, select the **New_Call_1** object (the highest level object that no longer matches).

4. In the **Interrogation Form** dialog, select **Debug Matching** from the drop-down box.

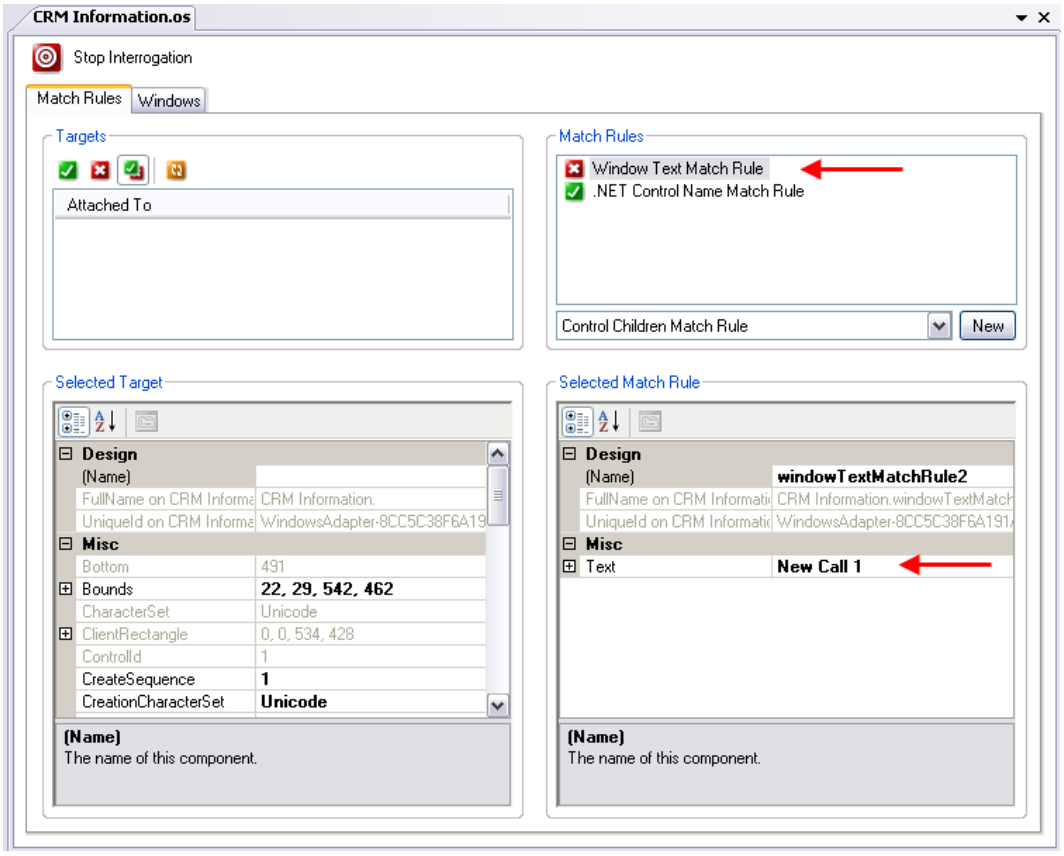


5. Drag the **bulls-eye** icon over the entire **New Call** window. When the entire window is surrounded by a thick black line, release the mouse button.

OpenSpan Studio uses a red x box to indicate the rules that are not matched. In this example, the **Window Text Match Rule** is not matched.

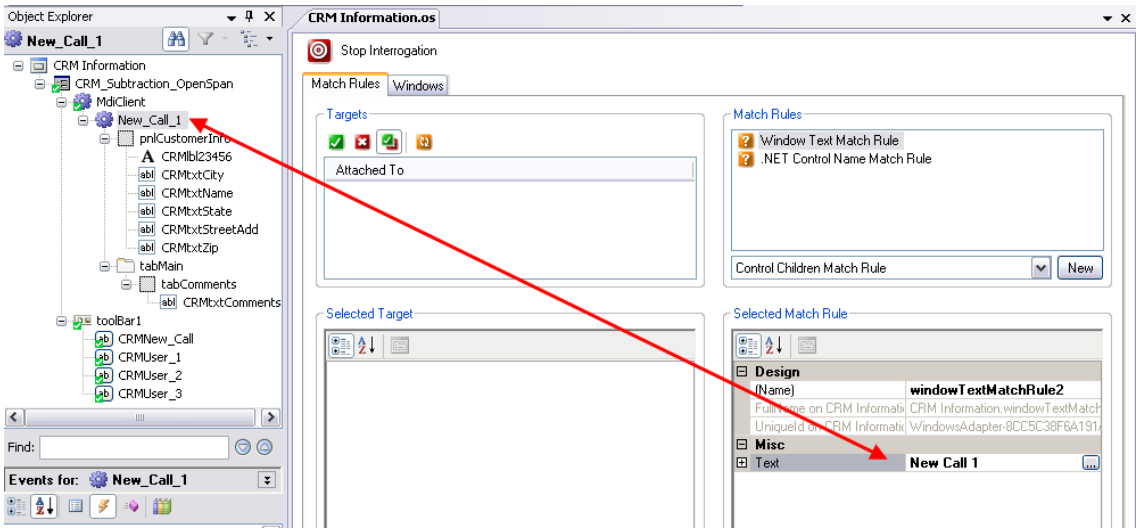


6. Select the **Window Text Match Rule** and note that the **Text** property in the **Selected Match Rule** grid shows **New Call 1**.

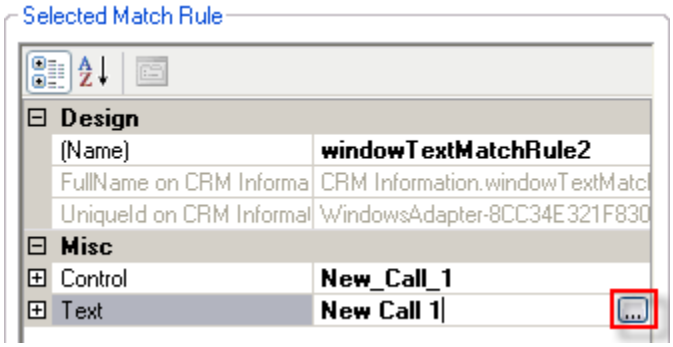


Note: New Call 1 is the text from the window when the control was originally interrogated and matched. Any subsequent windows that open will have a different number appearing after the text and will not be matched. For example: New Call 2, New Call 3, New Call 4, etc…

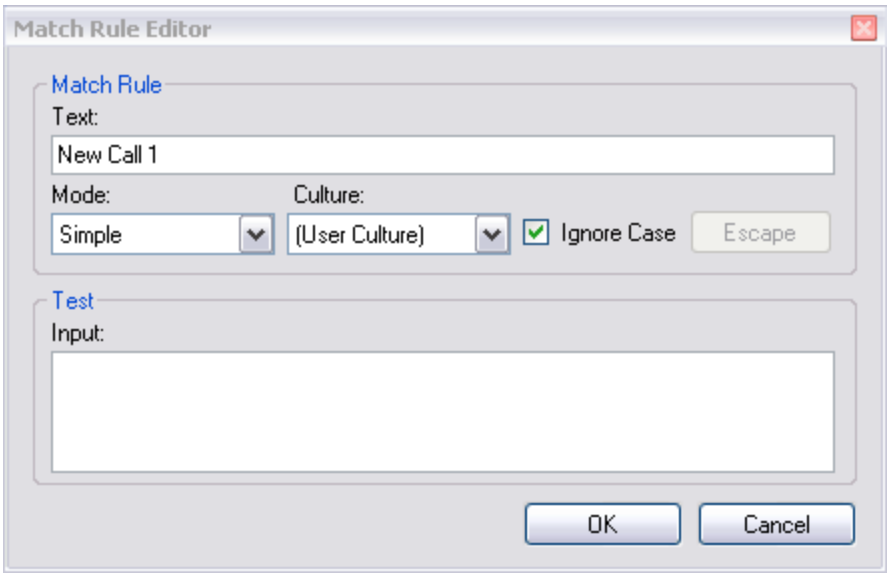
7. The Text property needs to be changed so that any window starting with the text New Call is matched. Highlight the Window Text Match Rule for the New Call 1 control.



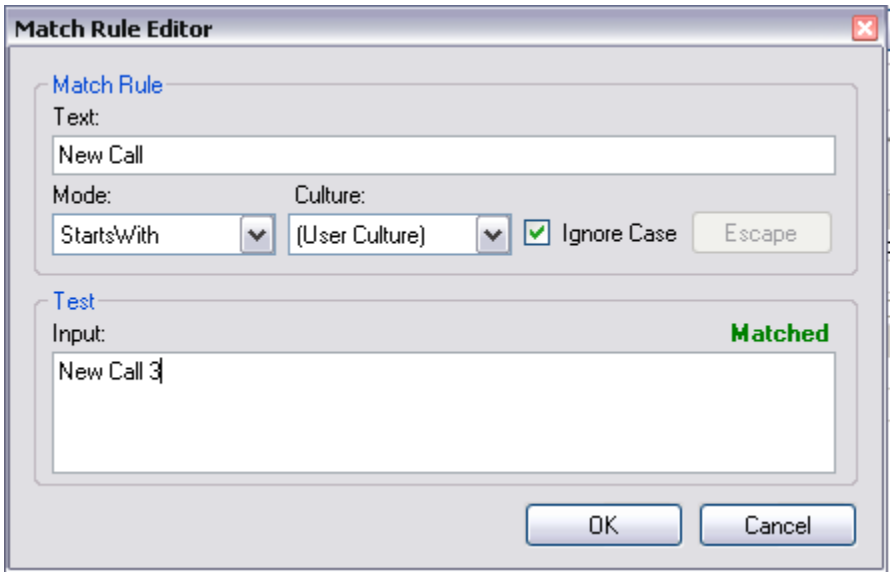
8. In the **Selected Match Rule** window, click on the browse button for the Text property:



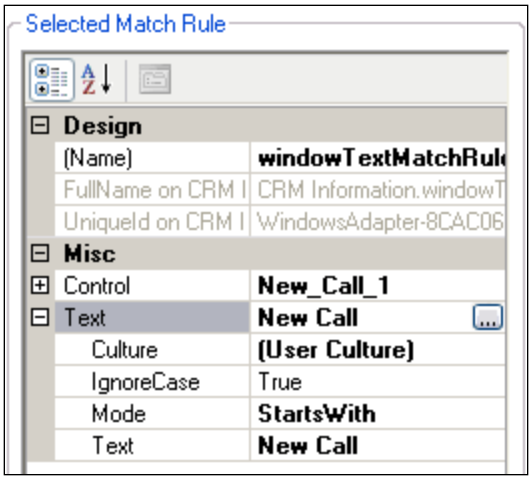
The Match Rule Editor displays:



9. The **Match Rule Editor** displays the Window Title "New Call 1" as the **Text** to match and the **Mode** as simple, which means an exact match. Change the **Mode** to *StartsWith* and change the **Text** to **New Call**. In the **Test Input** box, type New Call 3, you should see a message **Matched** as shown in the following image:

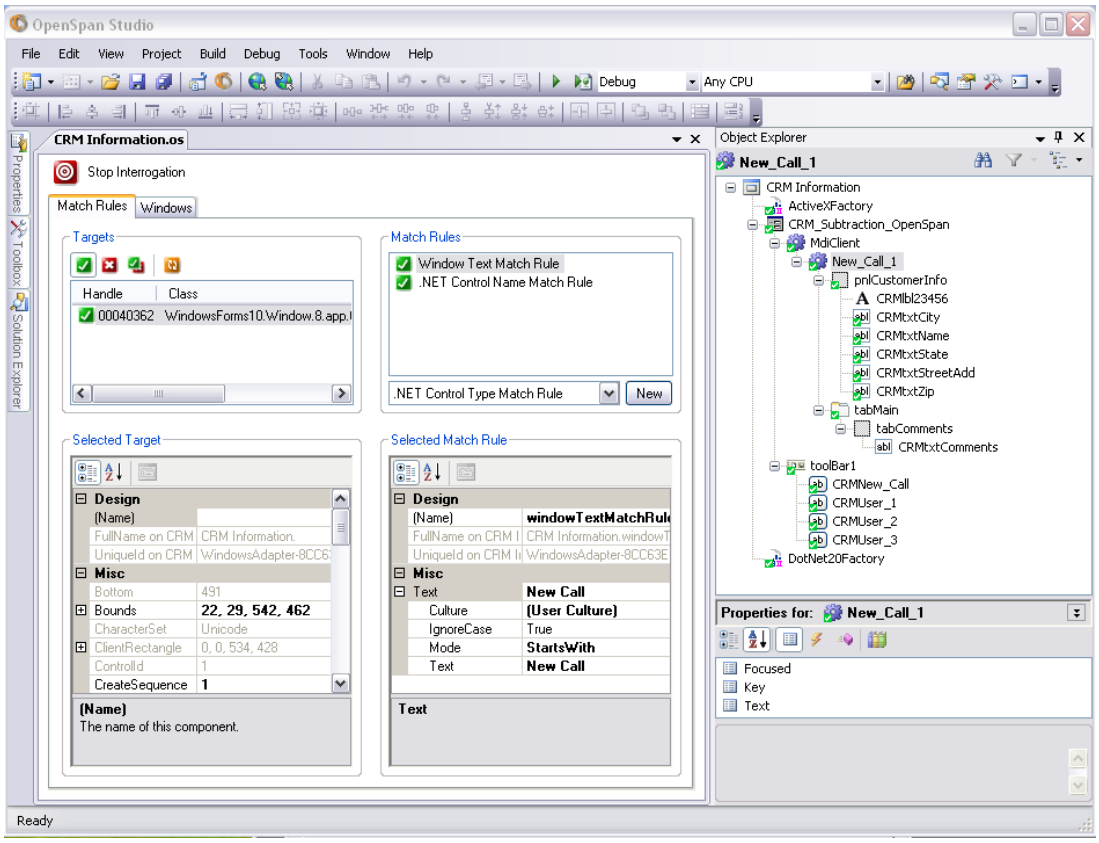


10. Click **OK** in the **Match Rule Editor**. The Text and Mode properties are updated as follows:



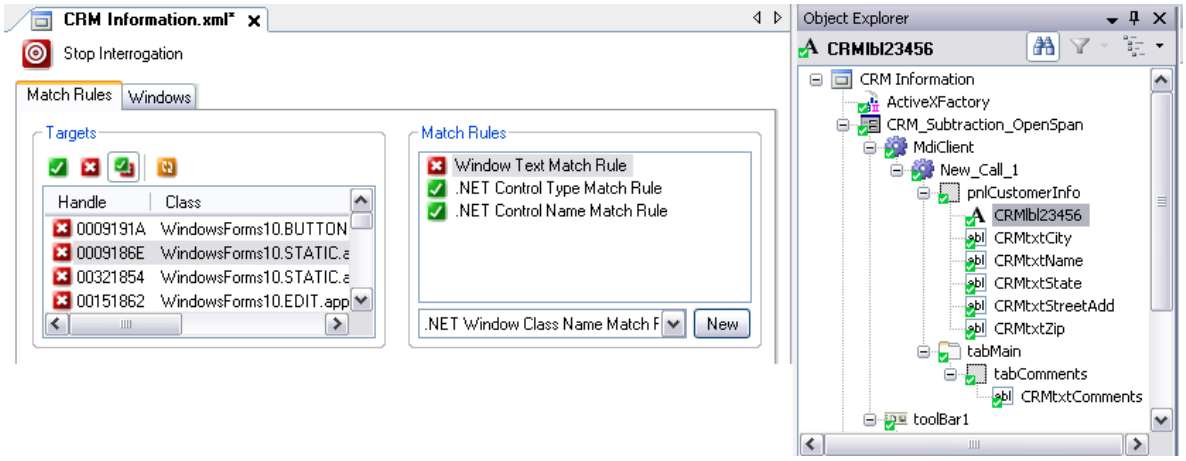
11. Select the **Refresh Matching** icon to continue.

OpenSpan Studio then matches any Window text starting with **New Call** and will not include the appended number.

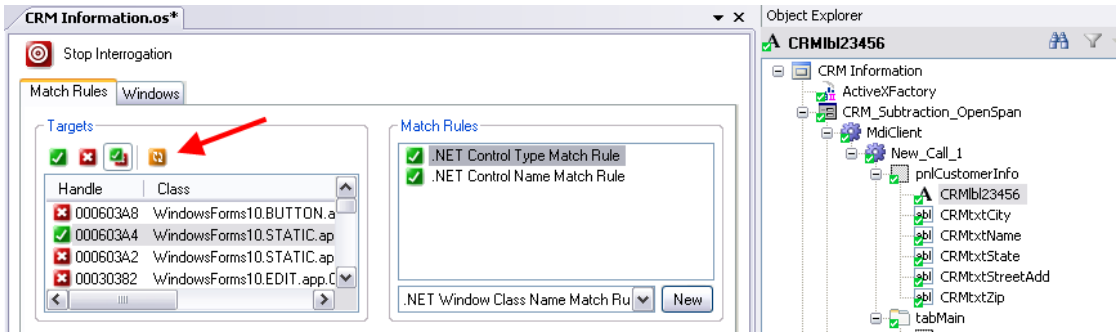


12. Select **File | Save CRM Information.os**.
13. Notice that the label for the account number control, **CRMlbl23456,** still appears in the Object Explorer as unmatched (there is no green check displaying beside it). Highlight the label, select **Debug Matching** in the **Interrogation Form** dialog and drag the bull's-eye over the account number on the **New Call** window.

14. The result of the Debug Matching shows that the Window Text Match Rule is broken:



15. The label is being matched by the text property. Since this text will be changing with every new account, delete the **Window Text Match Rule** by right clicking on the rule and selecting delete from the context menu. Select the **Refresh Matching** icon to match the item and clear any matching issues.



16. Stop interrogation and save the solution.

Exercise Summary

This exercise introduced you to some of the basics working with the Windows applications and using Windows adapters in solutions:

- Adding Windows Adapters to Solutions and Setting Path Property
- Interrogating Windows applications
- Troubleshooting Match Rules

Building Automations

The controls for both the Windows CRM application and Application Bar are now fully defined in OpenSpan Studio. The next step is to establish the logic which ties these applications together.

The automation (logic) is developed by establishing connections between Windows and Application Bar properties, methods, and events. You establish the connections by placing design blocks on OpenSpan Studio automations. Design blocks are visual representations of the properties, methods and events for the adapters, controls, and components.

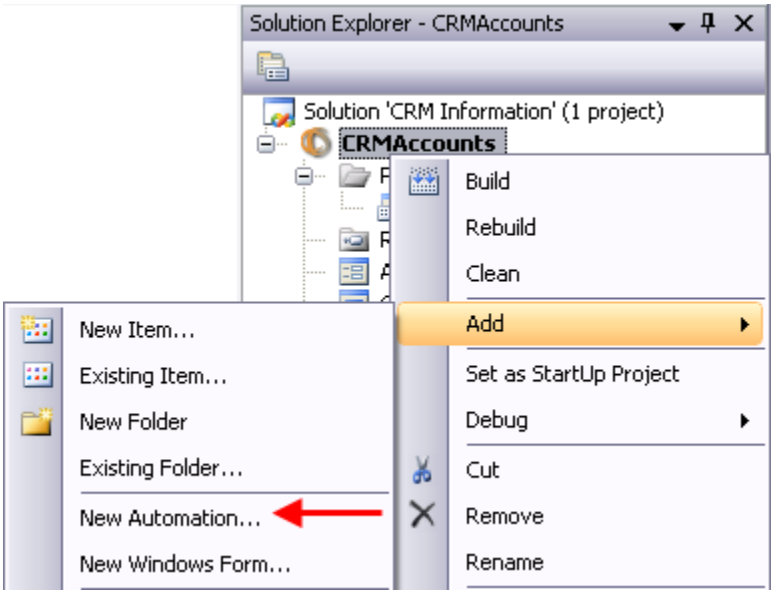
Exercise 3 - Building the Automation

In this exercise, you will build two automations:

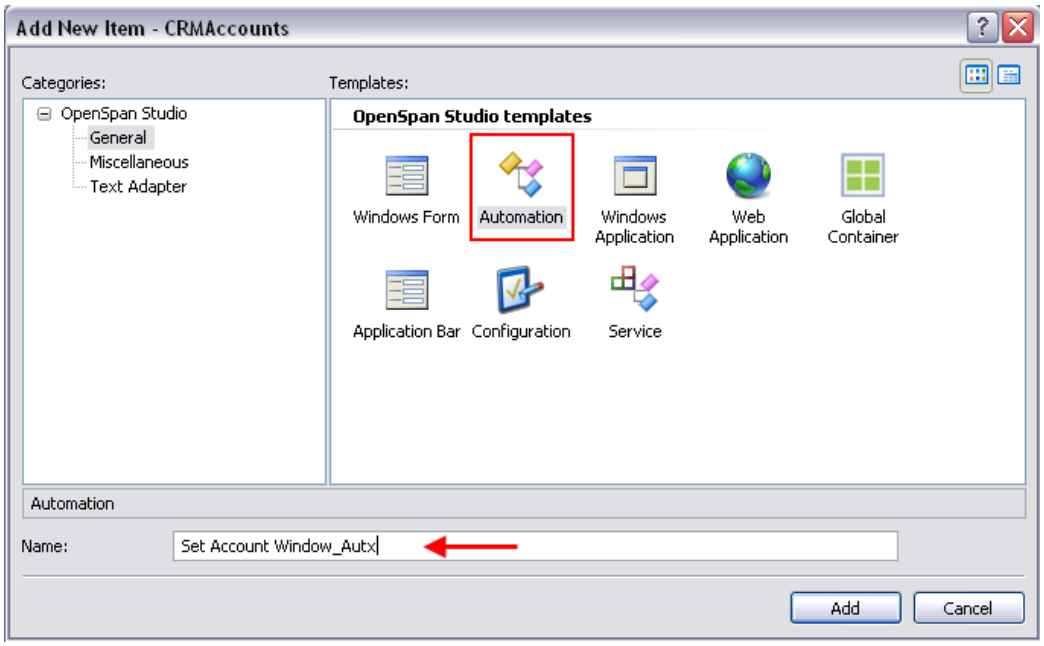
Set Account Window_Autx	Opens User 1 – New Call window, identifies all CRM windows with account numbers, and returns the Account number to CRM application.
Get Account Window_Autx	Returns Name and Address information for a selected Account.

Automation 1: Set Account Window_Autx

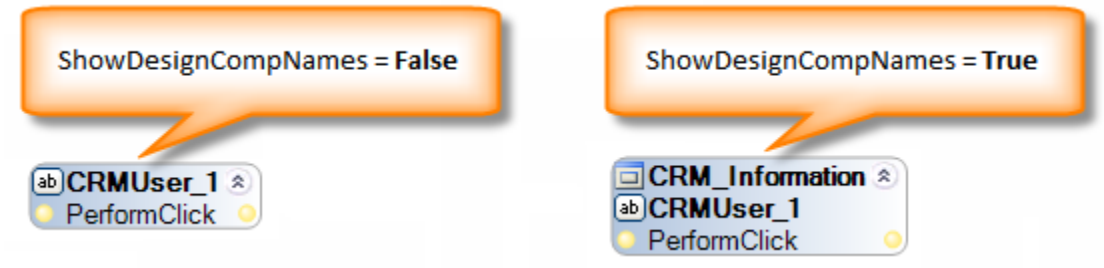
1. In the **Solution Explorer** window, **right-click** the **CRMAccounts** project and select **Add | New Automation** from the context menu.



2. The **Add New Item** dialog opens. This dialog displays all the available Design Components that you can add to a solution. The Automation Icon is highlighted allowing you to type in the name of the automation. Rename **Automation1** to **Set Account Window_Autx** by entering the new name in the **(Name)** field, then select the **Add** button:

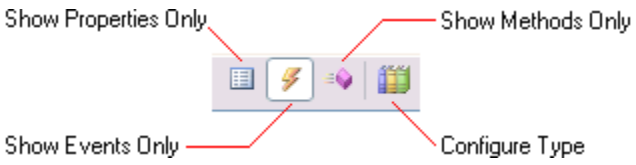


3. In the **Properties** window for the **Set Account Window_Autx** automation, set the **ShowDesignCompNames** property to **True** so that the project item name (i.e., CRM_Information) displays as the top row in the automation design blocks.

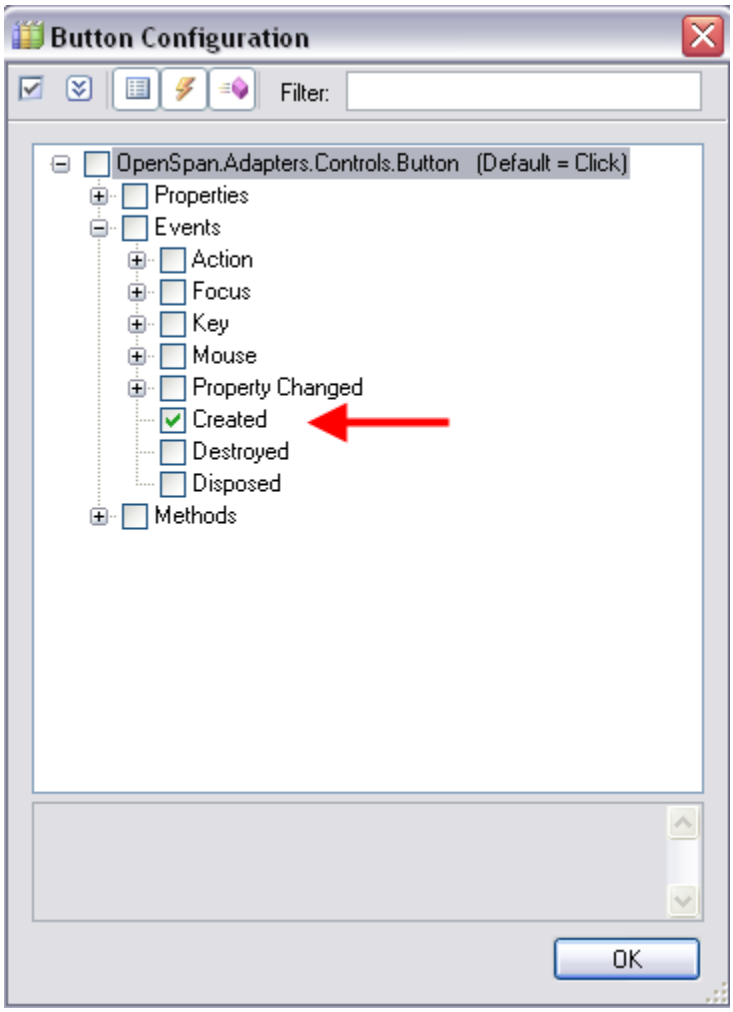


The first logic stream will click the **Login** button which will load the CRM application. Once the CRM application has started it will open the **New Call** window. To do this, the **Created** event for the CRM **Login** button will be used to PerformClick the **Login** button. We will then use the **PerformClick** method from the New Call button to open the New Call window.

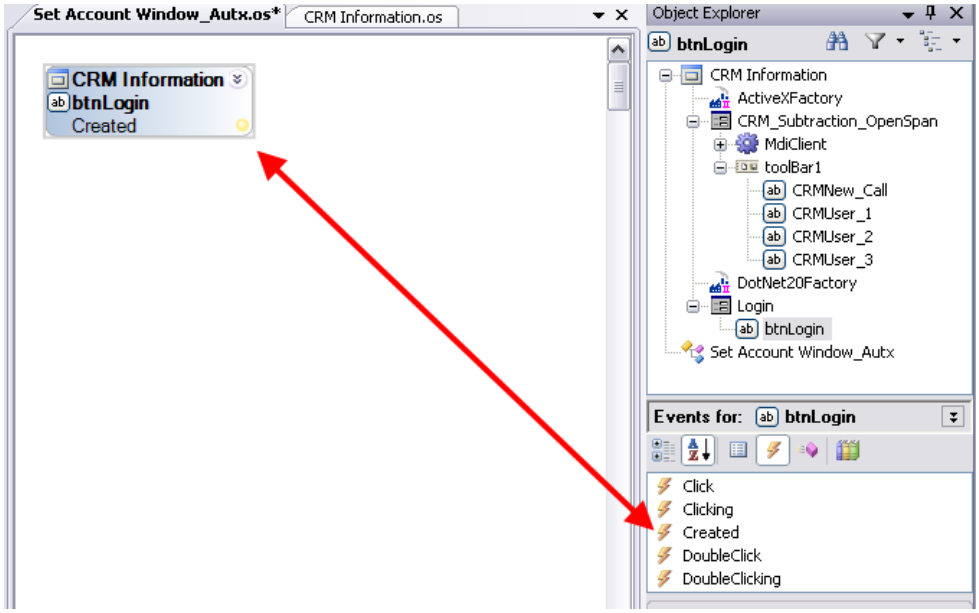
4. In the **Object Explorer** window, select the **CRMbtnLogin** project item.
5. Select the **Configure Type** icon to open the **Button Configuration** dialog.



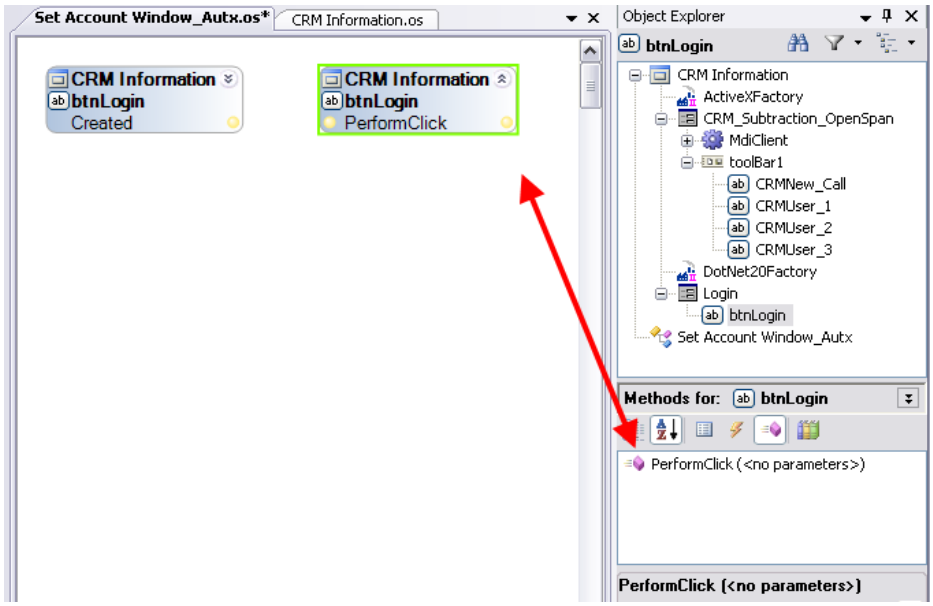
6. In the **Button Configuration** dialog, expand the **Events** item in the tree view and select the **Created** check box, and then click **OK** to save your selection.



7. With the **Show Events Only** button selected, drag and drop the **Created** event to the upper-left corner of the automation design pane.



8. Return to the Object Explorer and select the **Show Methods Only** button. Drag and drop the **PerformClick** method for the **CRMbtnLogin** to the Automation.

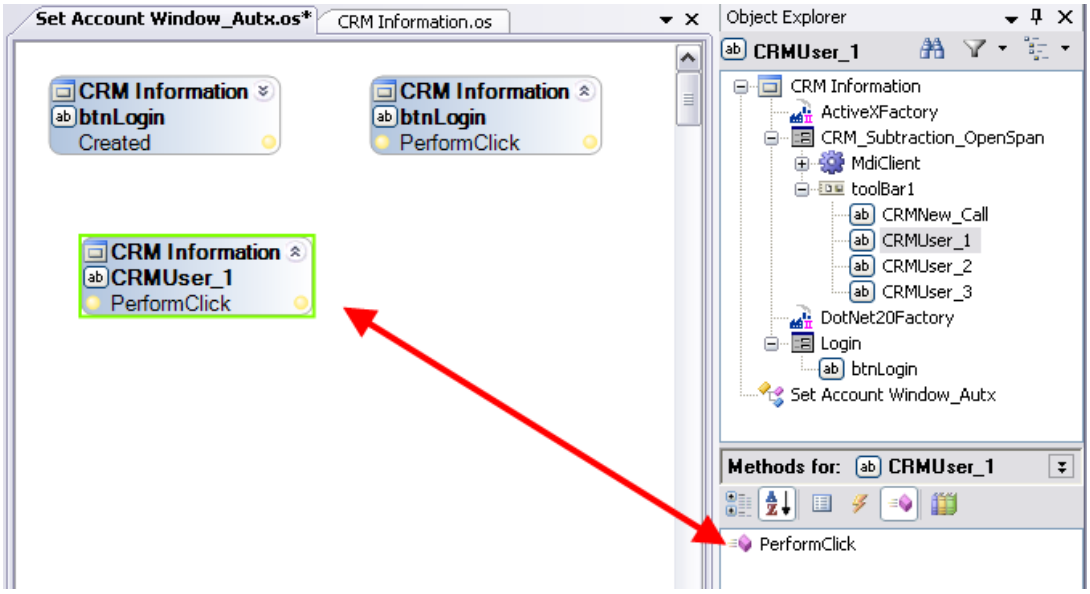


9. Return to the **Object Explorer** and select the **CRMUser_1** button under the **toolBar1** object.

10. Click the **Show Methods Only** icon in the lower pane.

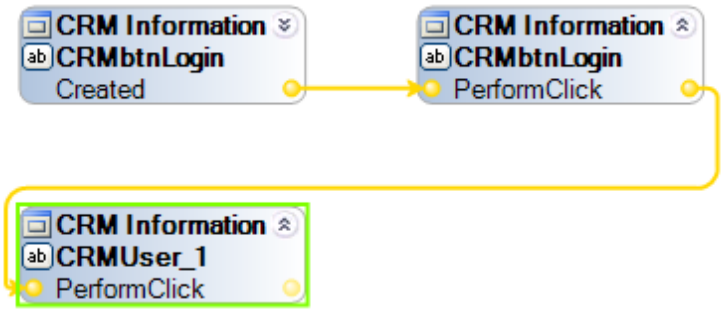
11. Drag and drop the **PerformClick** method to the automation design pane.

Note: If the method does not display, with the CRMUser_1 selected in the object explorer, click the **Configure Type** icon. Use the **ToolBarButton Configuration** dialog to add the **PerformClick** method.



Connect the execution path from the **Created** event to the input of the **PerformClick** method by drawing a line between the yellow event ports. Finish the automation logic by connecting the exection path from the **CRMbtnLogin PerformClick** method to the **CRMUser1 PerformClick**.

Note: To draw lines between design blocks, click a colored dot on one object and drag to the target object. Release the mouse button when the line connects the controls.



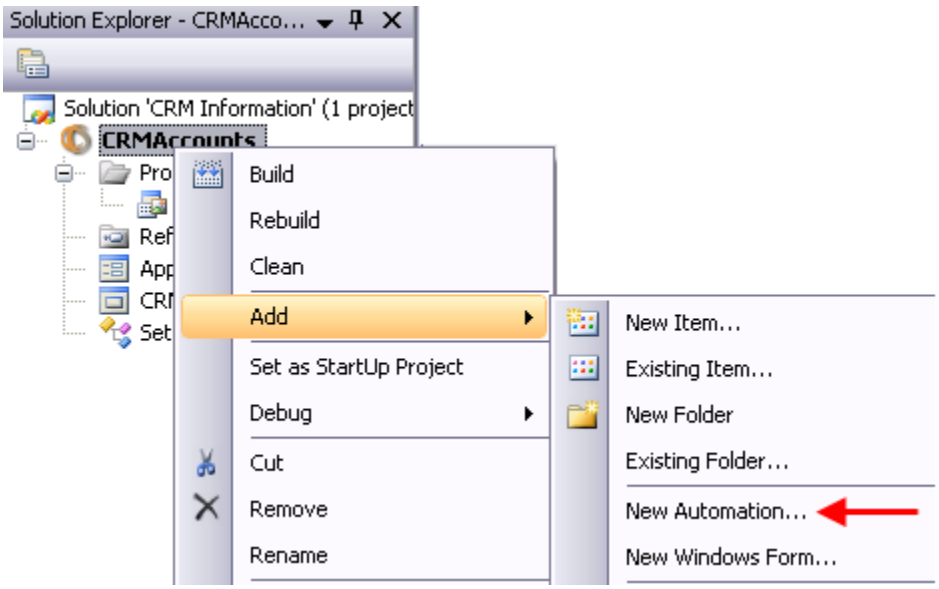
11. Select the Run Project icon  from the toolbar.
12. To ensure the **Set Account Window_Autx** automation is working as designed, confirm that both the application bar and the CRM application open.
13. Additionally, in the CRM application confirm that a **User 1** window opened automatically.

14. Select the **Stop** icon  from the toolbar. The **Stop** icon will stop the project and close both the Application Bar and CRM Information application.

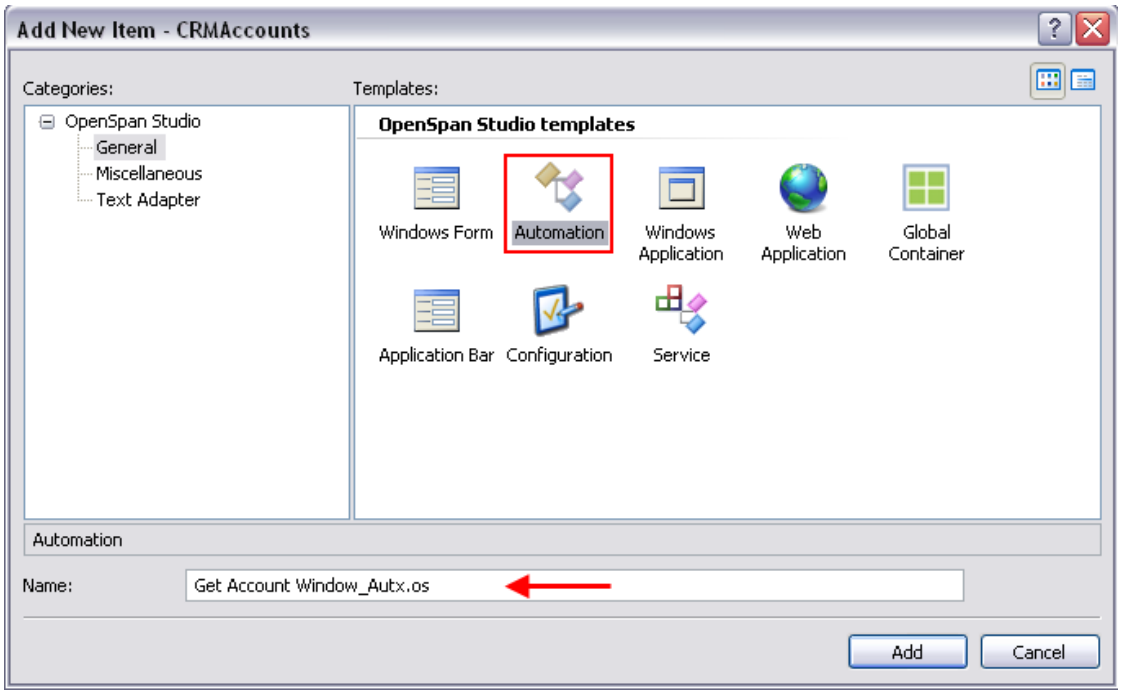
Automation 2: Get Account Window_Autx

The next part of the solution will push the information entered in the **New Call** window in the CRM application to the Application Bar fields. Begin by adding a new automation to the solution:

1. In the **Solution Explorer** window, right-click the **CRMAccounts** project and select **Add | New Automation** from the context menu.



2. **Add New Item** dialog opens. This dialog displays all the available Design Components that you can add to a solution. The Automation Icon is highlighted allowing you to type in the name of the automation. Rename **Automation1** to **Get Account Window_Autx** by entering the new name in the **(Name)** field, then select **Add.**

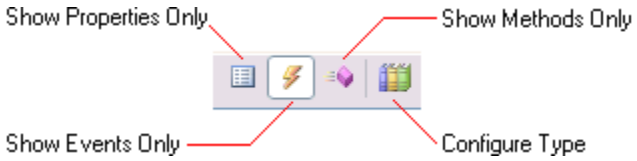


3. In the **Properties** window for the **Get Account Window_Autx** automation, set the **ShowDesignCompNames** property to **True** so that the project item name displays as the top row in the automation design blocks.

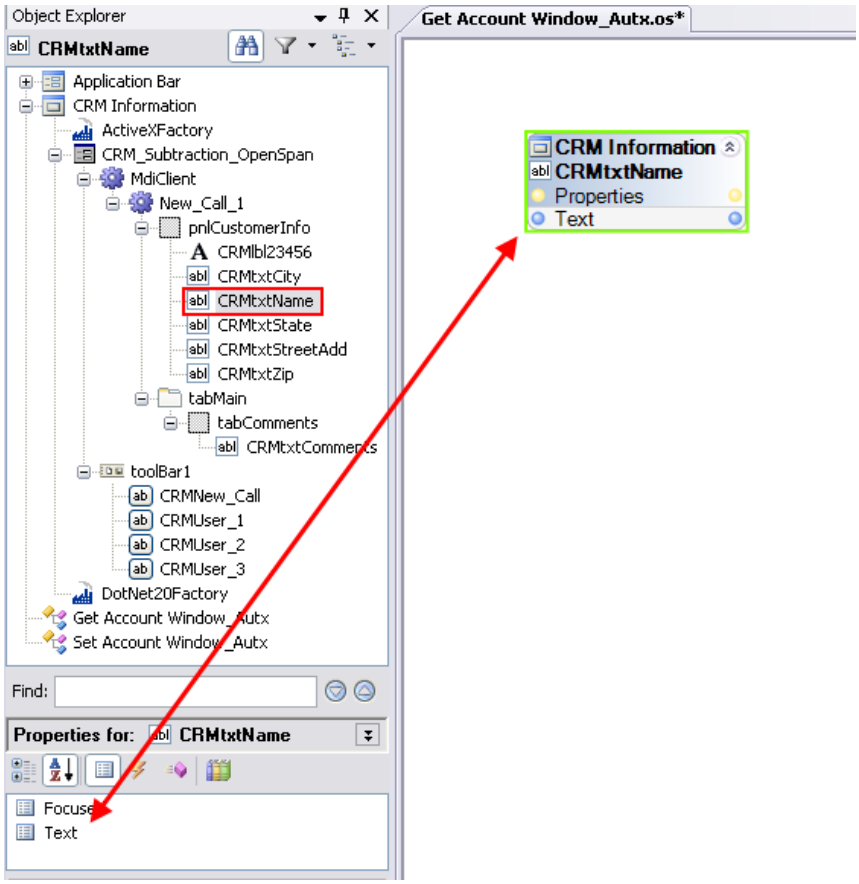
Note: The text (data) moves from the **CRM** application to the **Application Bar** using the following controls:

From - CRM Control	To - Application Bar Control
CRMtxtName	txtName
CRMtxtStreetAdd	txtAddress
CRMlbl23456	txtCurrentAcct

4. Open the **Object Explorer** window and select the **CRMtxtName** item.
5. Click the **Show Properties Only** icon on the lower pane.



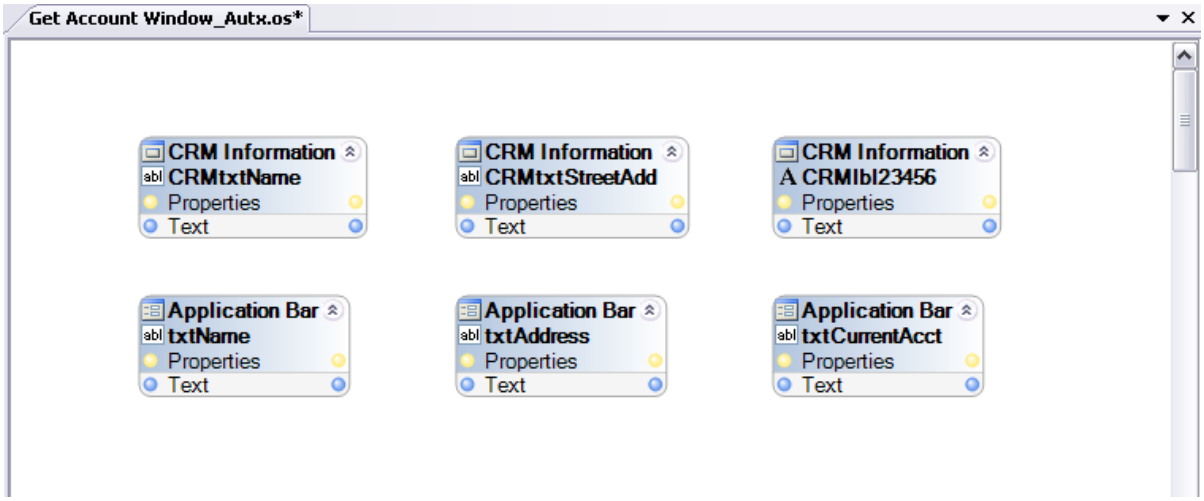
6. Drag and drop the **Text** property for the **CRMtxtName** to the upper left corner of the **Get Account Window_Autx** automation design pane.



7. Return to the **Object Explorer** and drag and drop the **Text** property for the following CRM application and Application Bar controls to the automation design pane:

CRM Controls	Application Bar Controls
CRMtxtName	txtName
CRMtxtStreetAdd	txtAddress
CRMlbl23456	txtCurrentAcct:

Your automation should look similar to the following:



Connecting the Data Paths for the Automation

To move the data from the CRM application to the application bar, you must first connect the data path from the CRM application text fields to the Application Bar text fields.

1. Connect the **text** output from the **CRMtxtName** control to the **text** input of the Application Bar **txtName.**

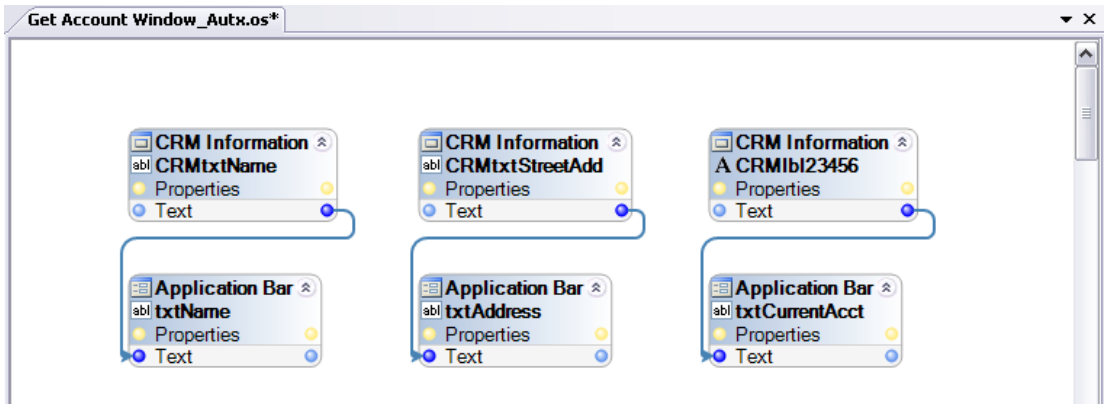
   To connect the data paths, draw a line from the blue data port on the right side (the origination point) of the CRM application object to the blue data port on the left side of the Text property for the application bar label object.

   Note: **Blue** lines represent the **data path** and **Yellow** lines represent the **execution path**. To draw lines between design blocks, simply click a colored dot on one object and drag to the target object and release the mouse button.

2. Connect the remaining control boxes.

CRMtxtStreetAdd	→	**Application Bar txtAddress**
CRMlbl23456	→	**Application Bar txtCurrentAcct**

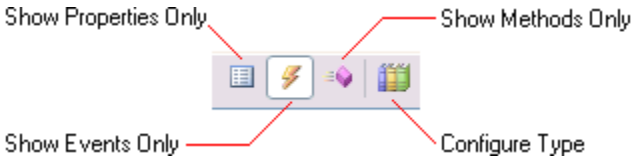
After you have connected all the data paths, the automation should look like this:



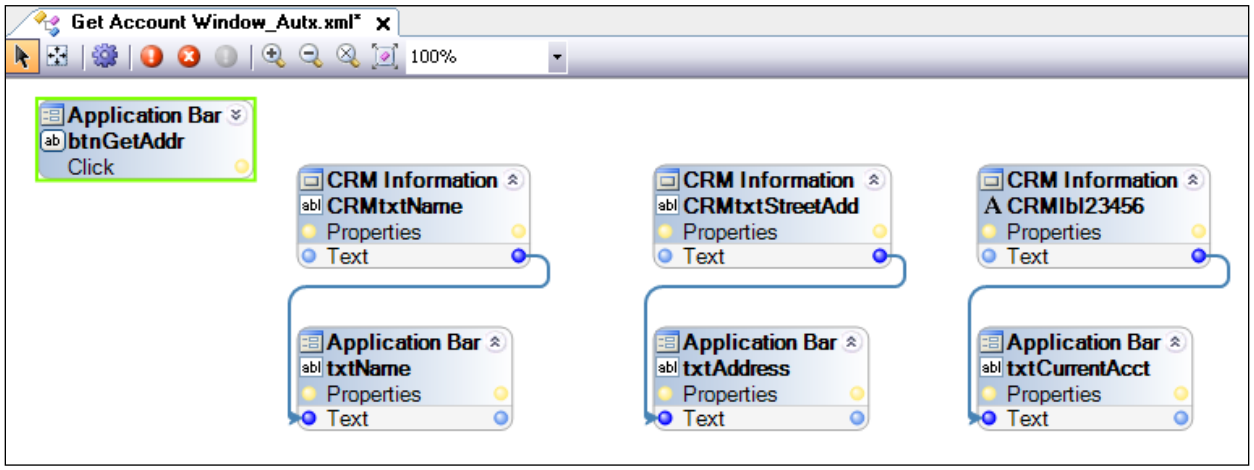
Connecting the Execution Path for the Automation

In order to trigger the moving of data, you can use several different events. For this solution, we will use the **Click** event for the **btnGetAddr** application bar button. The text moves from the CRM application to the application bar whenever you click the **Get Address** button on the application bar.

1. In the **Object Explorer** window select the Application Bar **btnGetAddr** item.
2. Click the **Show Events Only** icon in the lower pane.

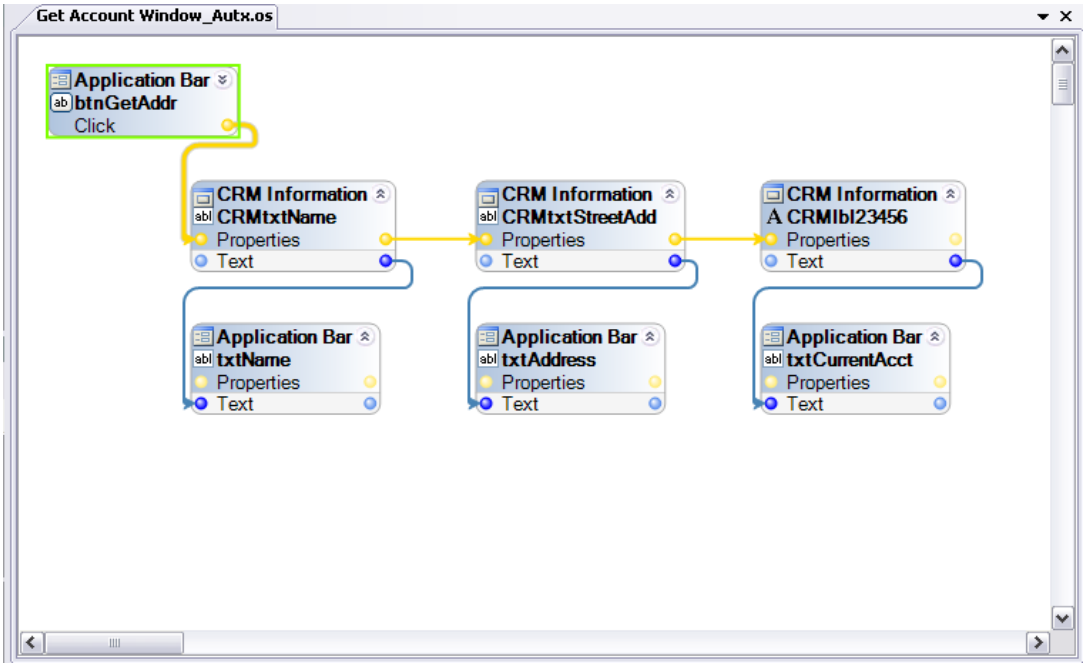


3. Drag and drop the **Click** event for **btnGetAddr** object to the upper left corner of the automation design pane.



4. Connect the execution path from the **btnGetAddr** button **Click** event to the **CRMtxtName Text** Property.

   To connect the execution paths that trigger the moving of data from the CRM application to the application bar, draw a line from the yellow execution port in the **btnGetAddr** design block to the yellow execution port **input** for the **CRMtxtName** design block.



5. Right-click the execution path (yellow line) and select **Asynchronous** from the context menu. Visually, the line changes from solid yellow to perforated.

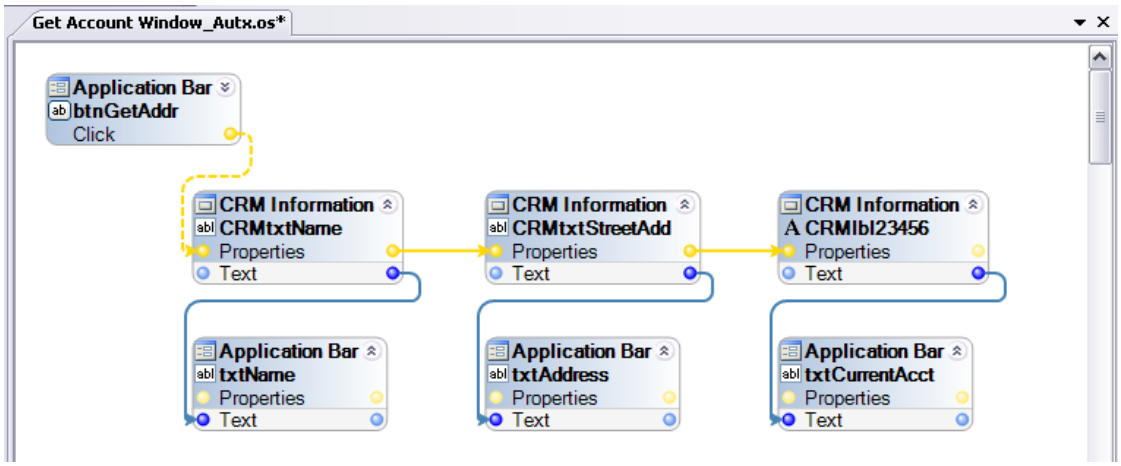
   Remember - When the OpenSpan starts a project, the first thread it creates is the user interface thread. The user interface thread is responsible for displaying any windows forms used in the project. Typically the user interface thread is either painting the user interface to the screen, or waiting for user input. When the user interacts with a windows form or its child controls, Windows sends messages describing the user activity to the user interface thread. The user interface processes hundreds of messages every second. In response to these messages the user interface thread raises windows forms events such as *Click*, *KeyPressed* or *TextChanged*. All windows forms events are raised **synchronously** and execute on the user interface thread

   Since the user interface thread is responsible for painting to the screen, any long-running activity that occurs on the user interface thread can block both painting and message processing. This gives the user the impression that the user interface is not responding. To avoid this, any automation that is triggered by a windows form event should execute an **asynchronous** link to release the user interface thread before it interacts with any adapter controls or non-windows forms components (e.g., web services, data access.).

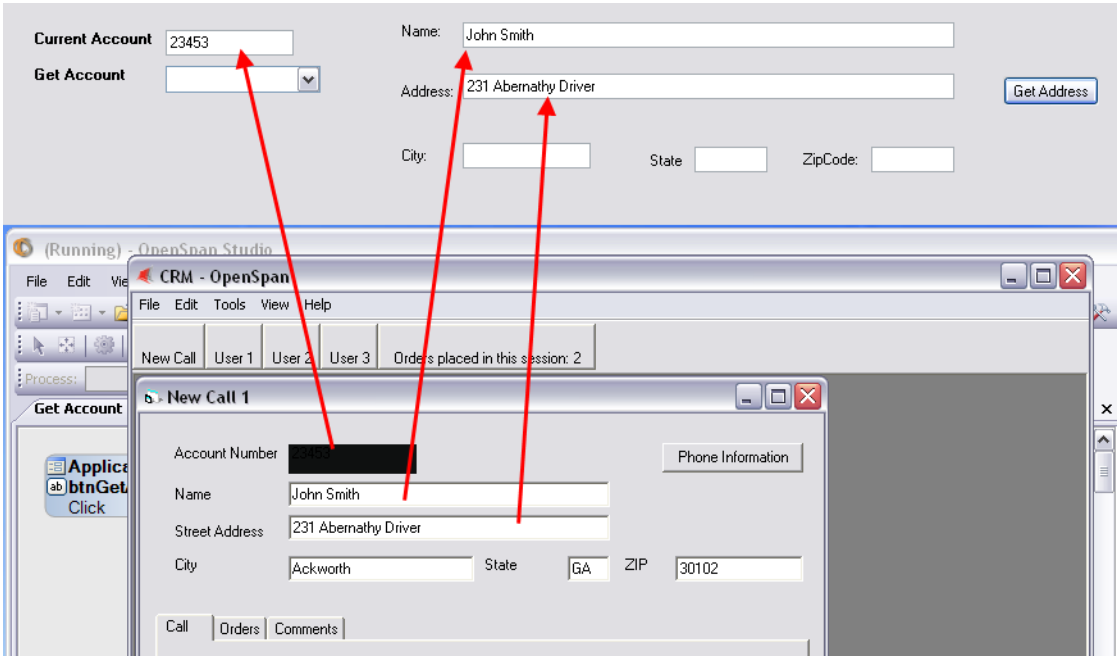
6. Complete the automation by connecting the remaining execution paths as follows:

CRMtxtName Properties	→	CRMtxtStreetAdd Properties
CRMtxtStreetAdd Properties	→	CRMlbl23456 Properties

7. When the execution paths have all been connected, the automation should look like this:



8. Click the Start Debugging icon  from the toolbar.
9. To ensure the automation is working as designed, make sure the New Call window is displaying information for a User then click the **Get Address** button to see data automatically move from the **CRM** application to the **Application Bar**.



Exercise Summary

This exercise introduced you to some of the basics working with the Windows applications and using Windows adapters in solutions:

- Adding Automations to Solutions
- Exposing and using Properties, Methods, and Events

This page intentionally left blank.

CHAPTER 4: WORKING WITH THE WEB ADAPTER

Integrating Web applications

As with Windows applications, OpenSpan Studio contains a special adapter for integrating with Web applications. In the same manner as the Windows adapter exercises you just completed, you add the Web adapter as an item to a project. The Web adapter exposes the functionality of the Web application and enables you, through the use of the OpenSpan Studio Interrogator, to isolate HTML controls with which you want to interact. There are 6 steps to using a Web application in an OpenSpan Studio solution:

1. Creating a Solution and Project.
2. Adding a Web adapter to the project.
3. Defining the Web adapter properties to identify the application.
4. Interrogating the Web application to identify the HTML controls for the solution.
5. Adding any required data and automations to create the work process flow desired.
6. Saving, deploying, and running the solution.

Interrogate Function

When interrogating a Web application, OpenSpan Studio uses the HTML tags of the selected targets as the basis for the Match Rules. In order for an object to be included in a solution, the object must be "matched" through a sequence of Match Rules. These match rules are based on the HTML tags of the selected item, such as the item's ID, name, width, etc.

Exercise 1 - Incorporating a Web Application into a Windows Solution

This exercise shows key concepts you need to know in order to integrate a Web application into an existing OpenSpan Studio Windows solution. This exercise builds on the project created in Chapter 3.

The following exercise leads you through the steps of creating a project that uses some of the key features of the Web adapter, such as:

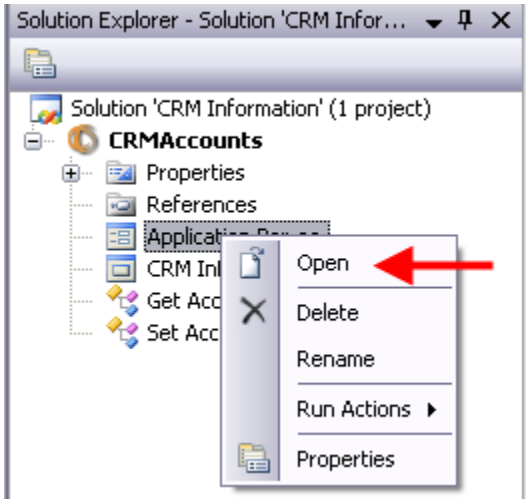
- Accessing web pages and checking for page creation
- Interrogating targets on a web page
- Sending data to a web page
- Sending events to a web page

The completed solution will do the following:

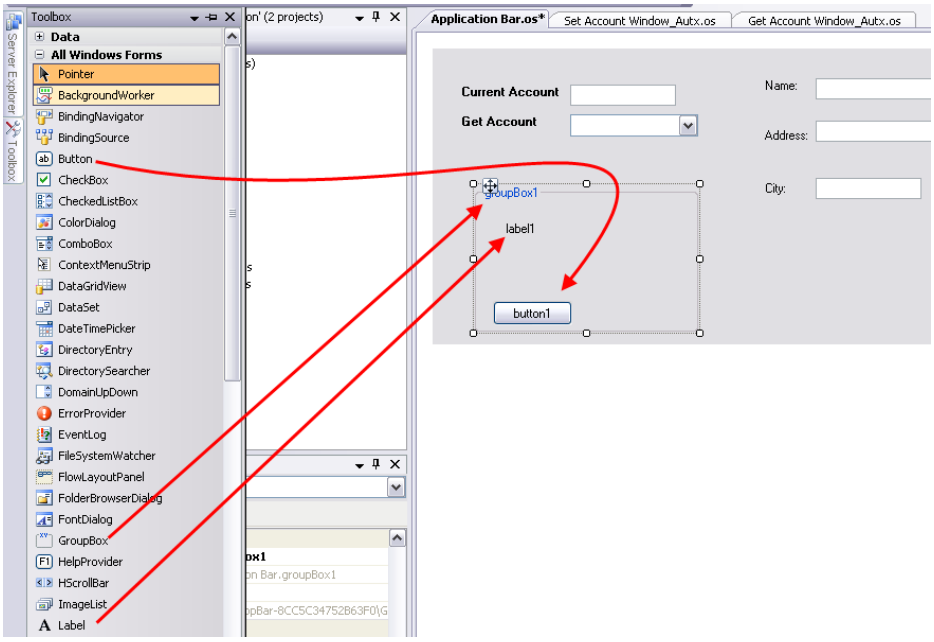
1. Open both a Web application and the Application Bar.
2. Push information from the Application Bar to a Web page.
3. Navigate to a specific web page and return data to the Application Bar.

Before you begin

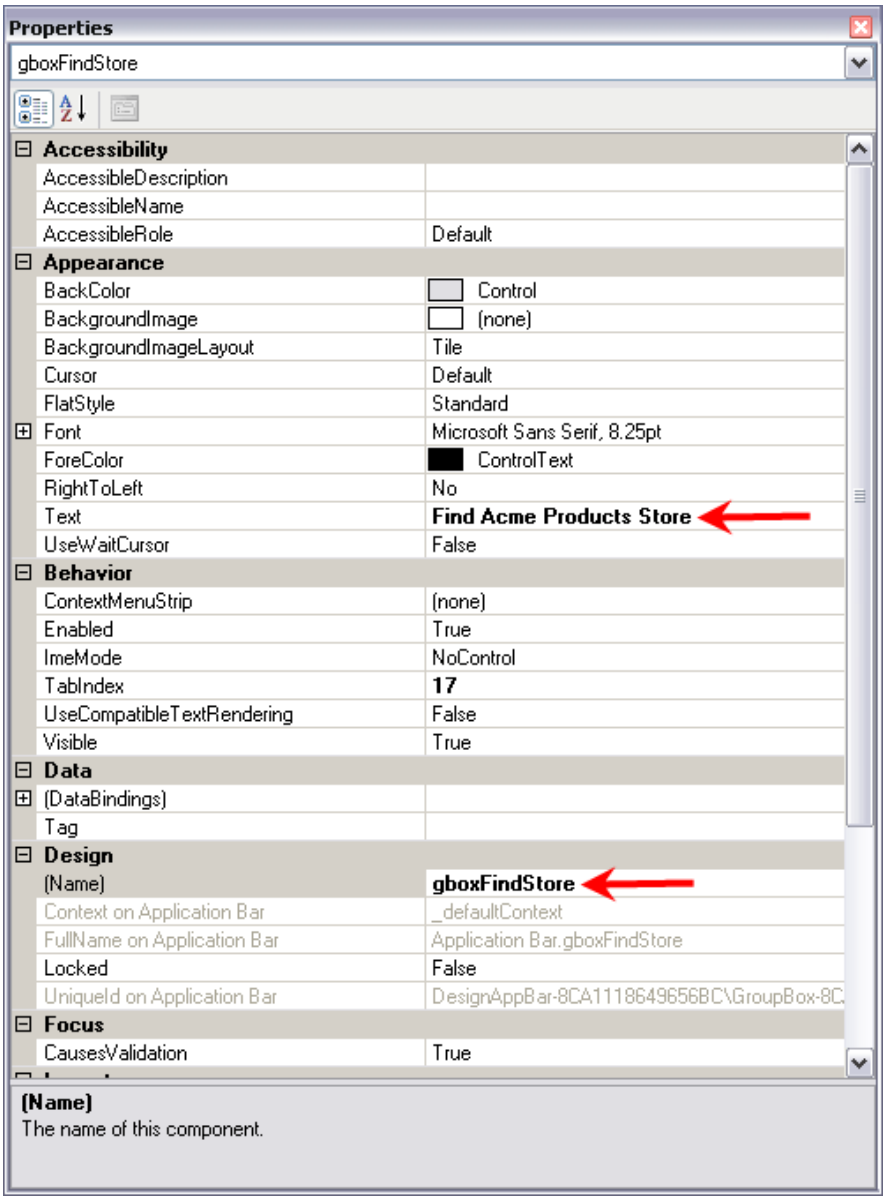
1. Open the **CRM Information** solution you created in **Chapter 3**.
2. In the **Solution Explorer** window, **right-click** the **CRMAccounts | Application Bar** project item and select **Open** from the context menu.



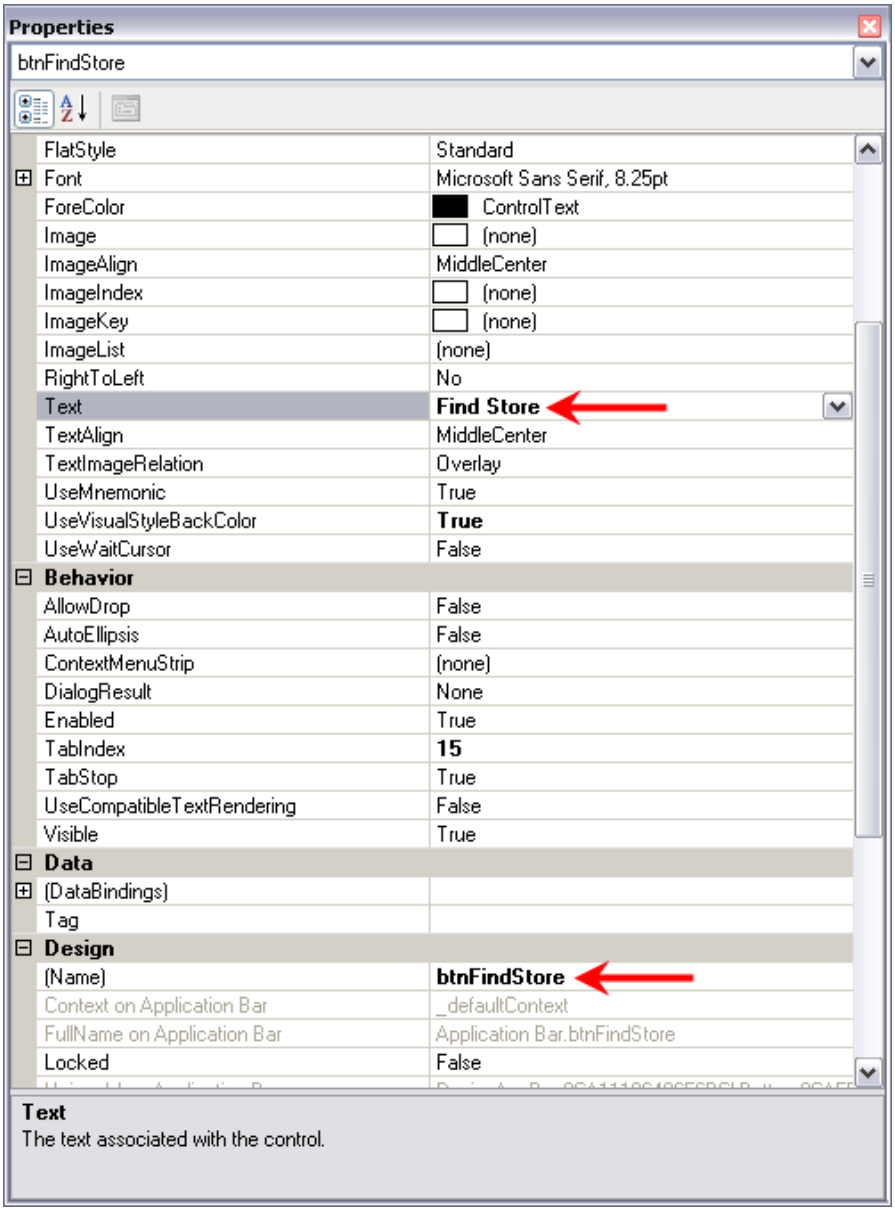
3. Select **View | Toolbox.**
4. In the Toolbox window, expand the **All Windows Forms** section.
5. Lengthen the **Application Bar** form and then drag and drop the following tools from the **Toolbox** on to the application bar.
   - GroupBox
   - Label
   - Button



6. Select **View | Properties Window.**
7. In the **Properties** window drop-down list, select **groupBox1**.
8. In the **Appearance** section, change the **Text** field entry to **Find ACME Products Store**.
9. In the **Design** section, change the **(Name)** field entry to **gboxFindStore**.

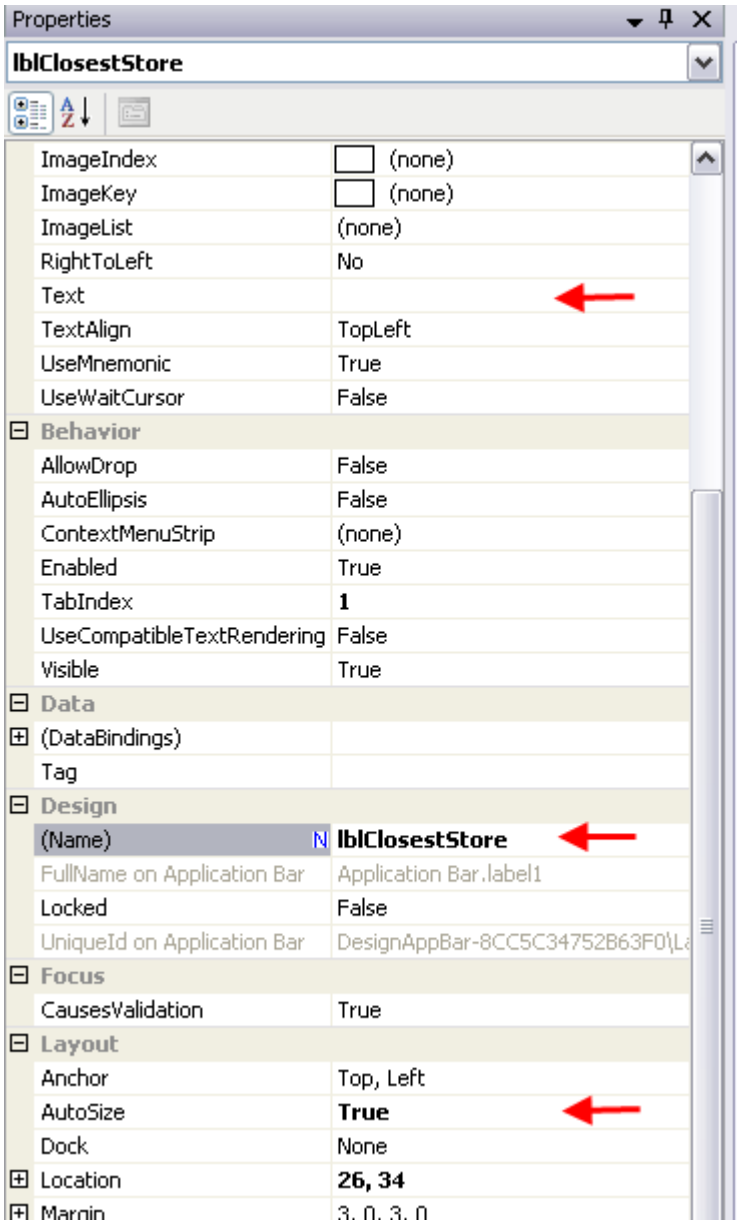


10. In the **Properties** window drop-down list, select **button1**.
11. In the **Appearance** section, change the **Text** field entry to **Find Store**.
12. In the **Design** section, change the **(Name)** field entry to **btnFindStore**.

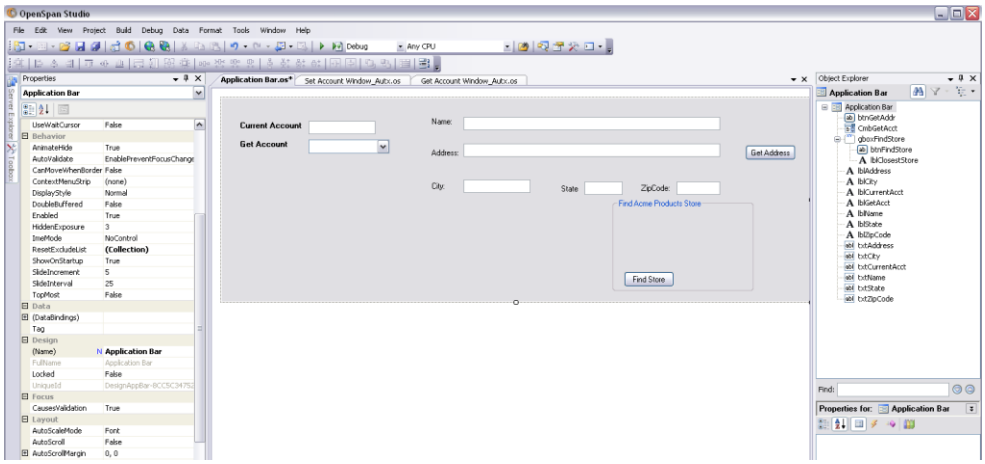


13. In the **Properties** window drop-down list, select **label1**.
14. In the **Appearance** section, delete the **Text** field entry so that no default label displays on the Application Bar Windows Form for this text label.
15. In the **Design** section, change the **(Name)** field entry to **lblClosestStore**.

16. In the **Layout** section make sure the (**AutoSize**) field is set to true.



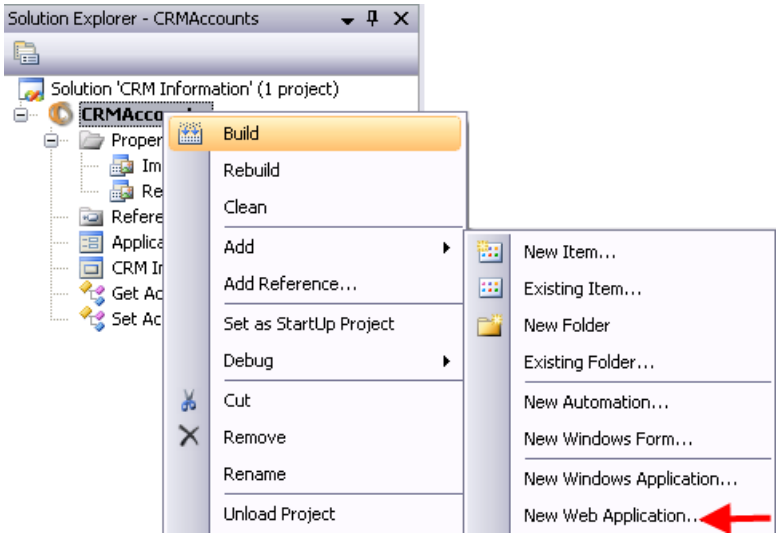
17. Resize and reposition the controls you just added so the **Application Bar** looks as follows:



18. Select **File | Save All**

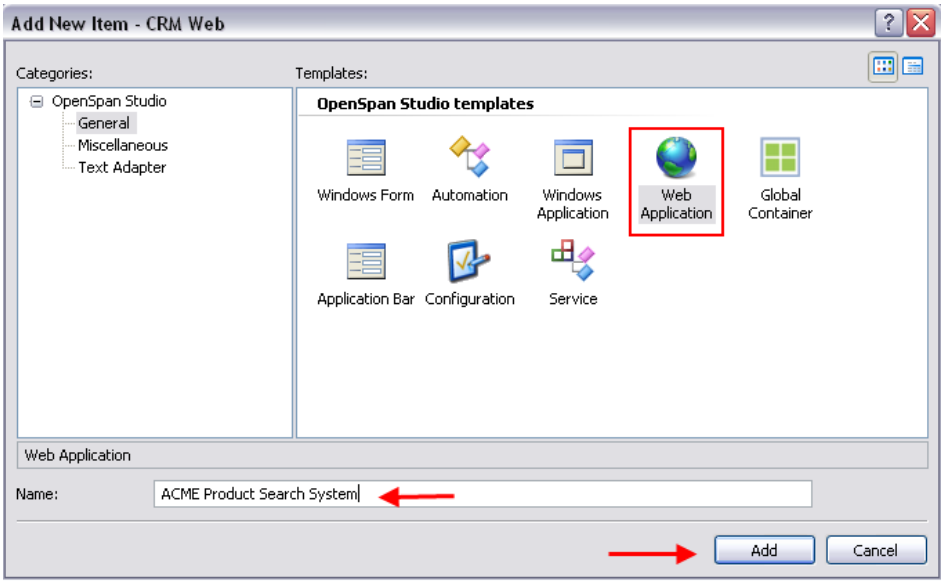
Adding Web Application to the Project

1. In the Solution Explorer window, right-click the CRMAccounts project and select Add | New Web Application from the context menu.

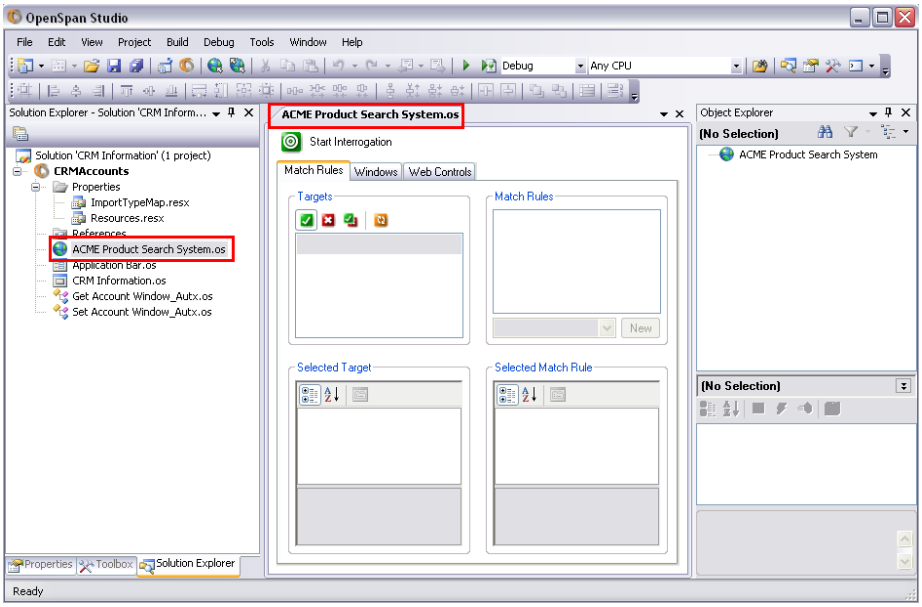


The **Add New Item** dialog opens.

2. Select the **General** category, and then click the **Web Application** template.
3. Type **ACME Product Search System** in the **Name** field, and then click **Add**.



The new Web Application item (**ACME Product Search System**) is added to the **Solution Explorer** and the **ACME Product Search System.os** design pane automatically opens.



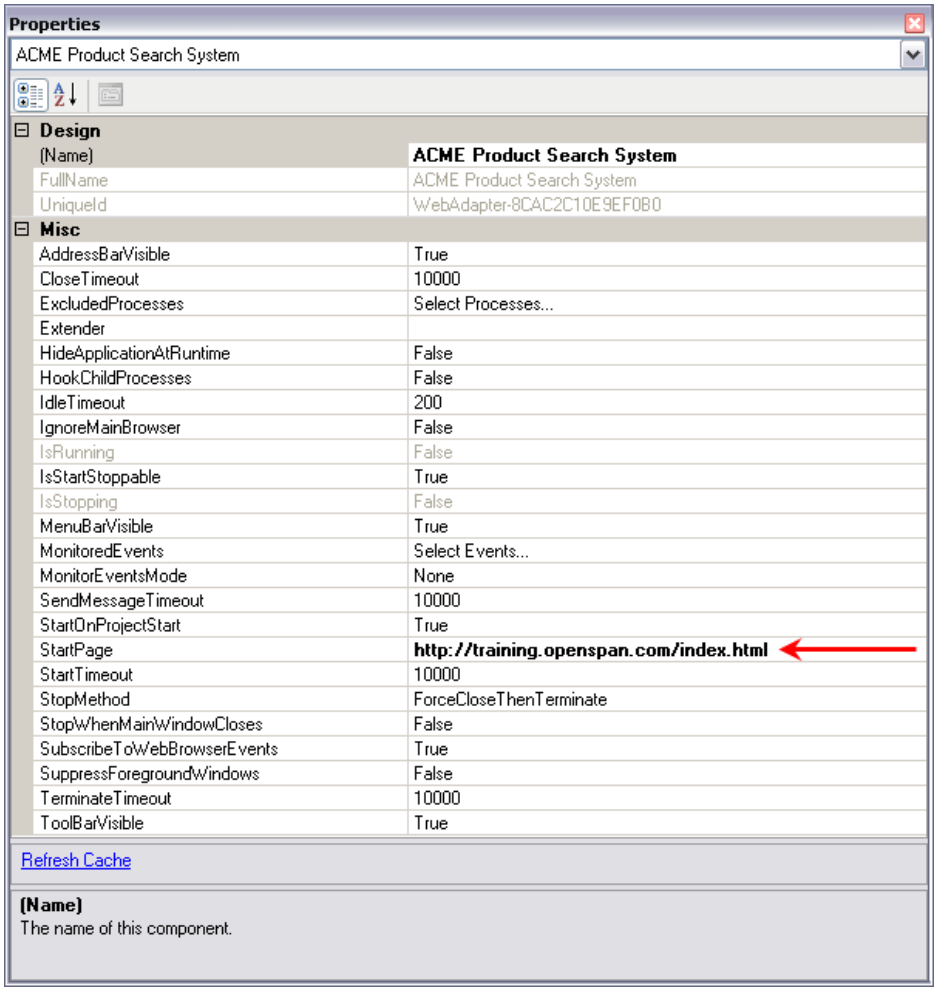
4. Select **File |Save All.**

Setting properties for the Web Application

1. Click the **ACME Product Search System.os** tab to bring the ACME Product Search System design pane into focus.

Note: If the design pane is not open, right-click the **ACME Product Search System** item in the **Solution Explorer** window, and select **Open** from the context menu.

2. Select **View | Properties Window**.
3. Select **ACME Product Search System** in the drop-down list at the top of the **Properties** window.
4. In the **StartPage** field type **http://training.openspan.com/index.html**

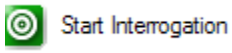


Note: By specifying the **StartPage** property, you are ensuring that the solution begins at the http://training.openspan.com/index.html web page. If no StartPage is specified, then the home page of the browser will open.

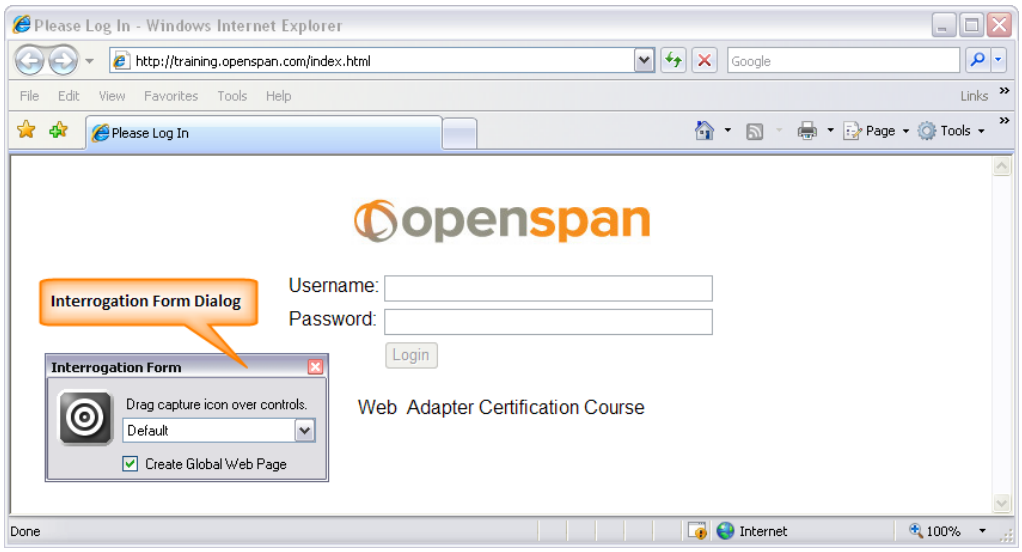
5. Select **File** | **Save All**.

Interrogating the Web Page

1. Open the **ACME Product Search System.os** design pane.
2. Click the **Start Interrogation** button at the **top** of the ACME Product Search System. os design pane to launch the Interrogate function.



The **Interrogation Form** dialog launches and the **Login** web page opens.

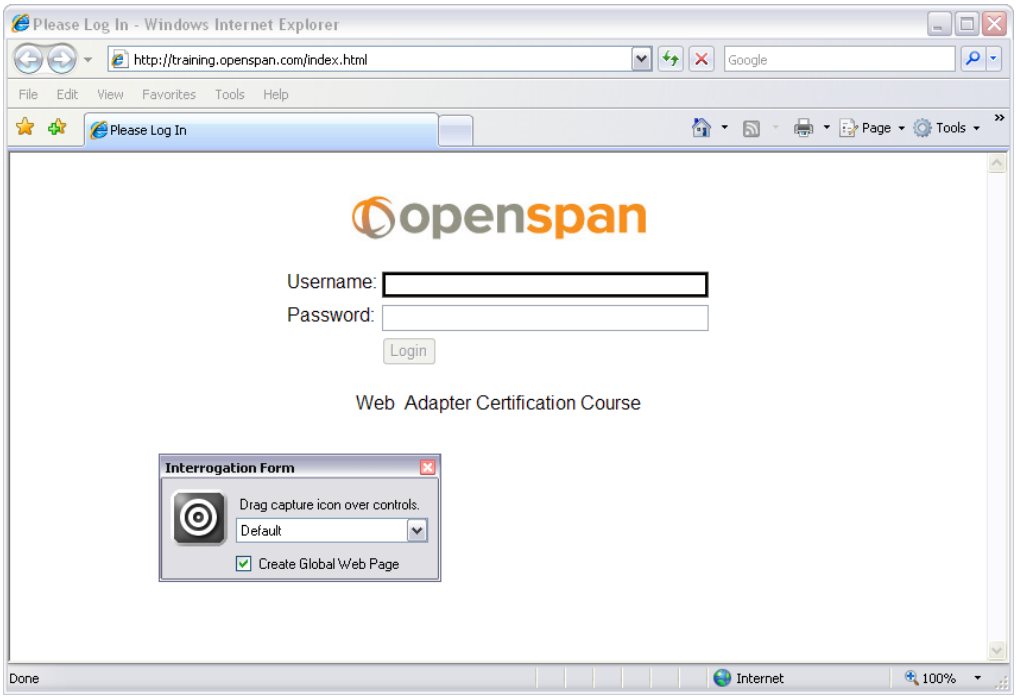


3. If it is not already checked, then select the **Create Global Web Page** check box on the **Interrogation Form** dialog box.

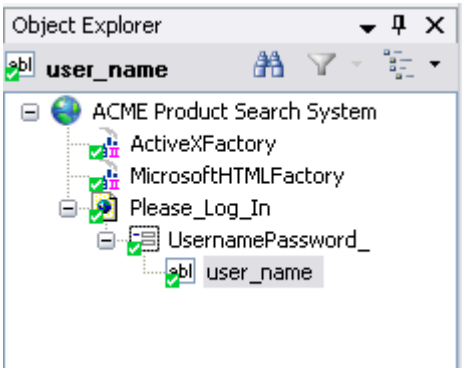
   Note: You should select the **Create Global Web Page** option when interrogating a web page, except in situations where the HTML frame controls, browser window, and/or IE7 tabs are required in your solution. When you interrogate a web page with the **Create Global Web Page** option selected the aforementioned controls are ignored (i.e., they are not interrogated and therefore are not displayed in the Object Explorer window nor are they available for your automations). Additionally, when you interrogate a web page with the **Create Global Web Page** option selected the resulting solution will be compatible with both IE6 and IE7, because the IE7 tabs are ignored (i.e., not interrogated).

4. For this exercise you will manually login to the ACME Product Search System. However, as part of the automation you create later in this exercise you will confirm if the Login page has been created so it is necessary that you interrogate at least one **Login** page control.

5. Click and hold the mouse button down on the bulls-eye shaped icon in the **Interrogation Form** dialog, and drag the icon over the **Username** textbox. When the textbox is surrounded by a black rectangle, release the mouse button.

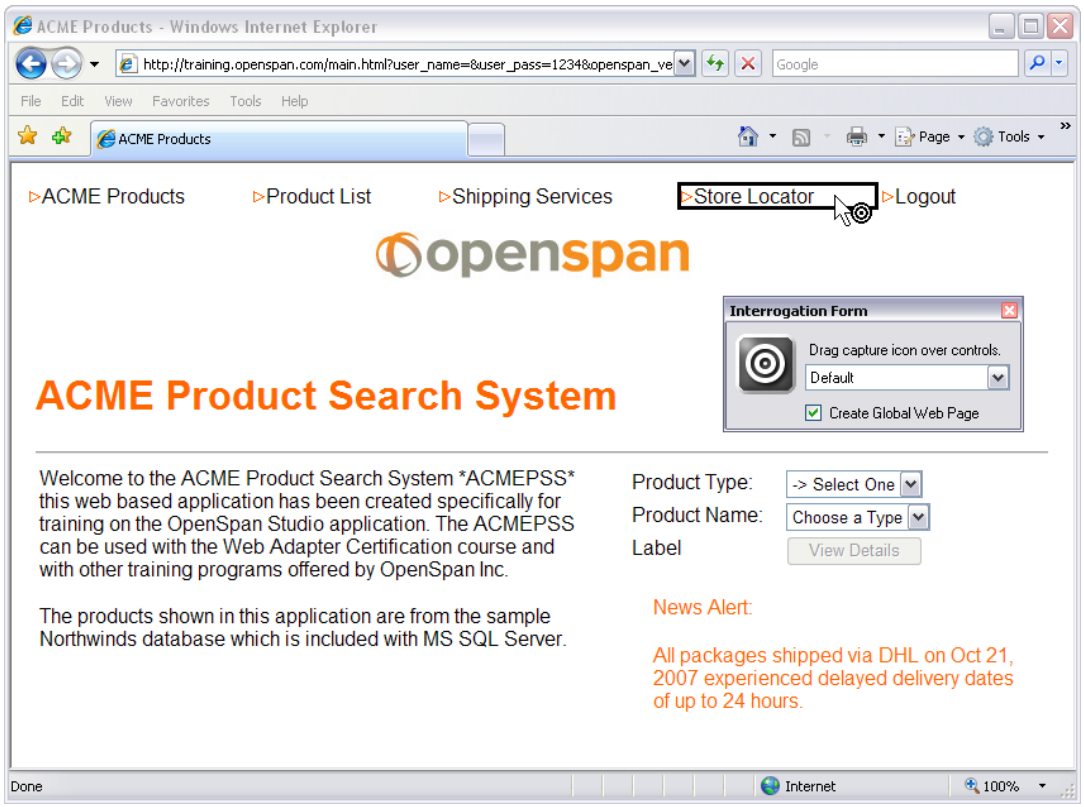


The **user_name** textbox and its parent, the **Please_Log_In** page, now appear in the Object Explorer.

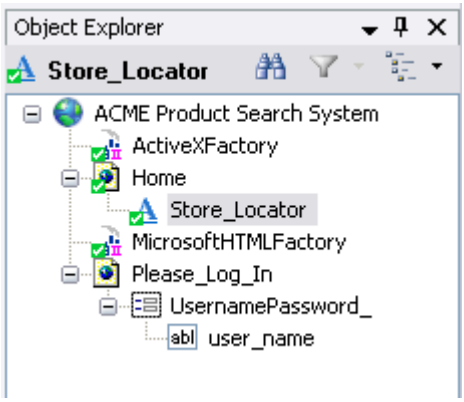


6. Enter any four alphanumeric characters in both the **Username** and **Password** text boxes on the training web page.
7. After the forth character is entered in the **Password** text box the **Login** command button becomes enabled. Click the **Login** command button to enter the ACME Product Search System.

8. Click and hold the mouse button down on the bulls-eye shaped icon in the **Interrogation Form** dialog, and drag the icon over the **Store Locator** menu option. When the menu option is surrounded by a black rectangle, release the mouse button.



The **Store_Locator** menu option now appears in the Object Explorer.

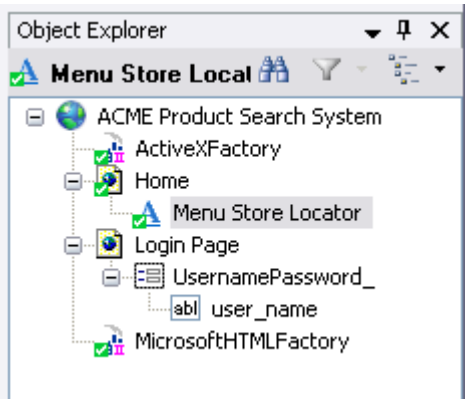


9. Select **View | Properties Window**.
10. In the **Properties** window, rename the following items you have interrogated and that you will use later in this exercise.

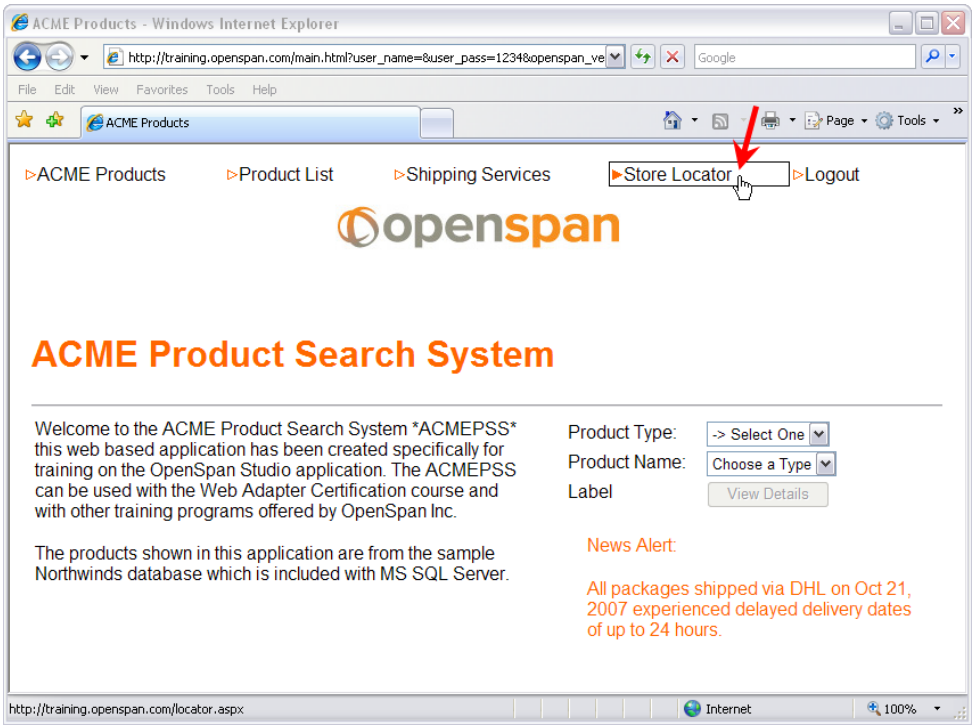
Interrogated Object	Rename To
Please_Log_In	Login Page
Store_Locator	Menu Store Locator

Note: As you are interrogating each object, you can switch to the **Properties** window and rename the objects so you can more easily identify them during the automation design process.

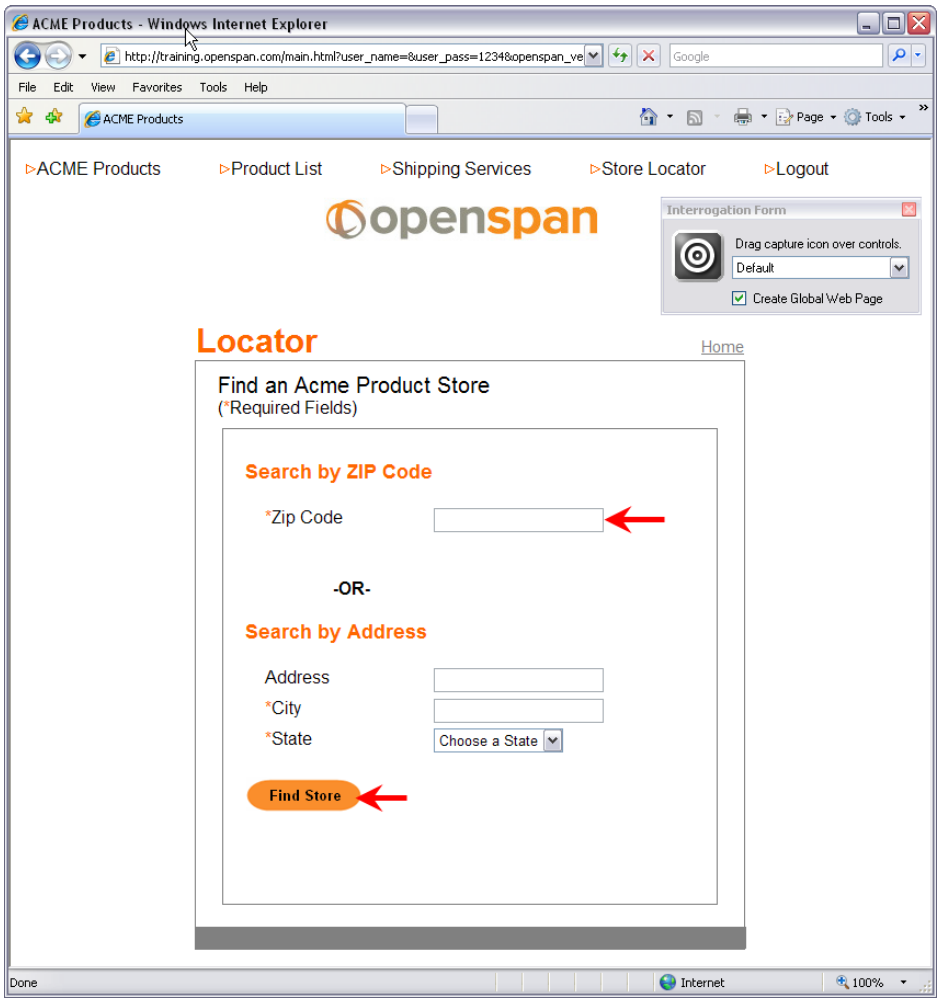
The **Object Explorer** window for the ACME Product Search System should look like this:



11. Click the **Store Locator** menu option to advance to the **Locator** page.

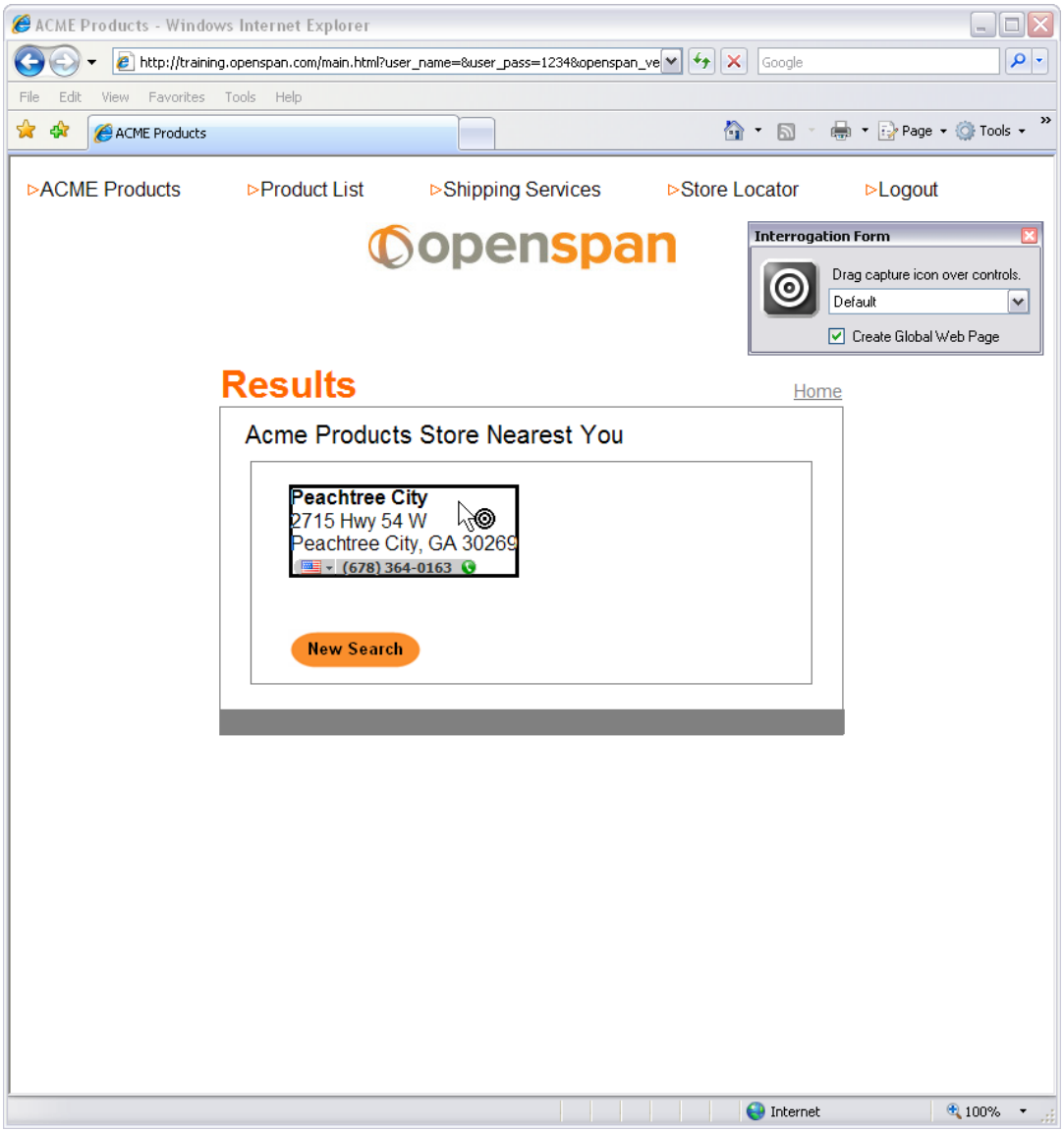


12. On the **Locator** page, interrogate the **Zip Code** text box and the **Find Store** command button.



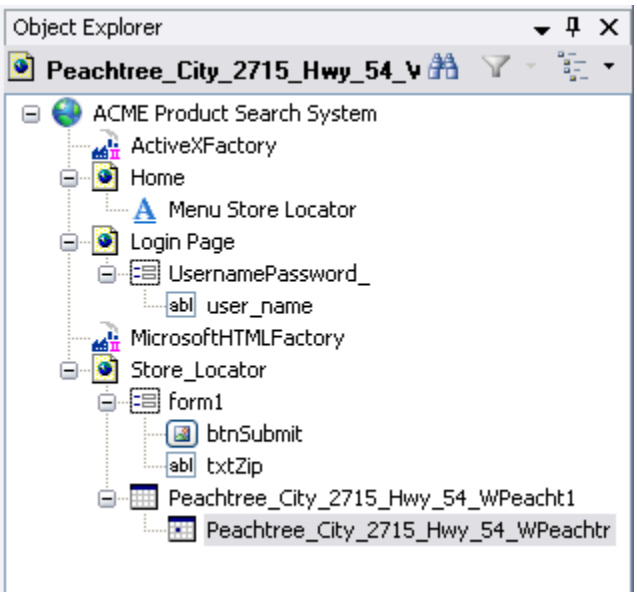
13. Enter the zip code **30022** in the **Zip Code** field of the **Locator** page and then select the **Find Store** command button.
14. Selecting the **Find Store** command button initiates the store search and opens the **Results** page.

15. The **Results** page displays the address and telephone number of the store nearest to the zip code you entered. Now Interrogate the **Results** table. It is important to interrogate just the table to obtain a unique match. When you drag the bulls-eye icon over the table control release it in the upper right-hand corner of the area surrounded by the black rectangle (see below).



16. Click the **Stop Interrogation** button.
17. Select **File | Save All.**

The **Object Explorer** should look like this:

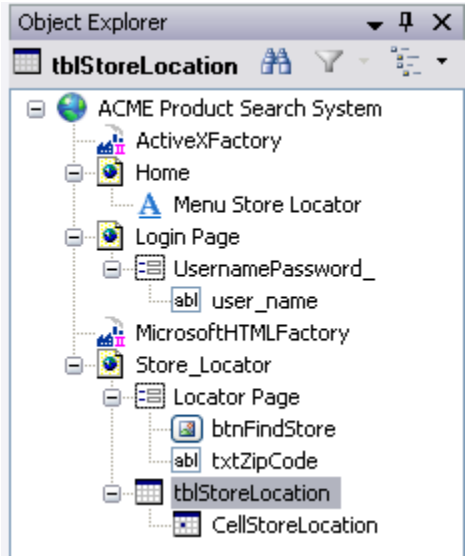


When you interrogated the **Results** table, OpenSpan identified the cell where the store location is displayed in the table. The parent control represents the table and the child control represents the specific cell where the store location information is displayed.

In the **Properties** window, rename the items you just interrogated.

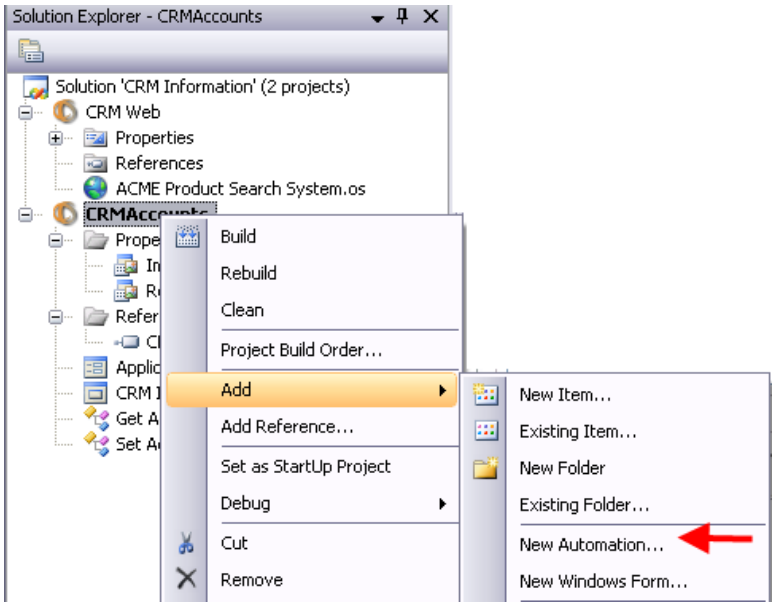
Interrogated Object	Rename To
form1	**Locator Page**
btnSubmit	**btnFindStore**
txtZip	**txtZipCode**
Peachtree_City_2715_Hwy_54_WPeacht1	**tblStoreLocation**
Peachtree_City_2715_Hwy_54_WPeachtr	**CellStoreLocation**

The **Object Explorer** should now look like this:

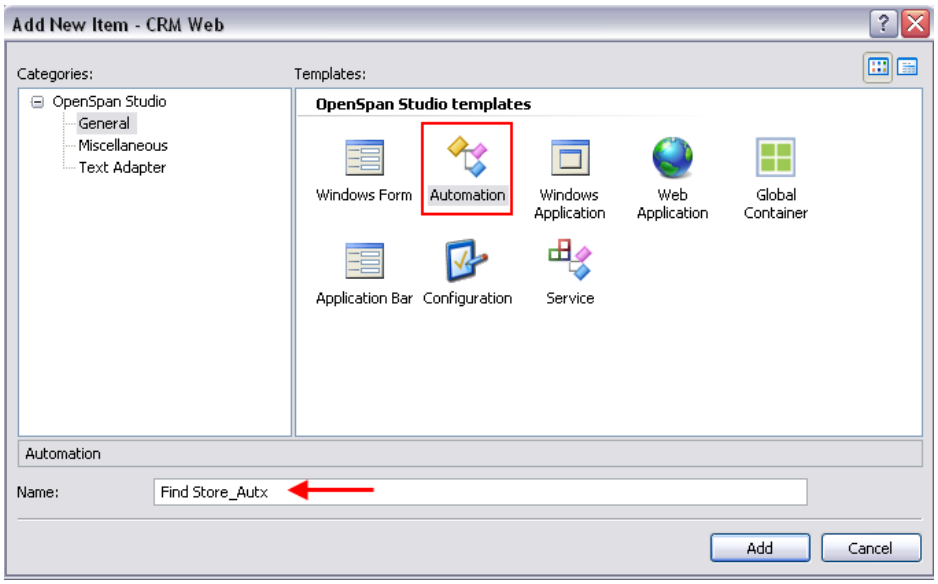


Adding the Automation

1. In the **Solution Explorer** window, **right-click** the **CRMAccounts** project and select **Add | New Automation** from the context menu.



2. **Add New Item** dialog opens. This dialog displays all the available Design Components that you can add to a solution. The Automation Icon is highlighted allowing you to type in the name of the automation. Rename **Automation1** to **Find Store_Autx** by entering the new name in the **(Name)** field, then select Add



3. In the **Properties** window for the **Find Store_Autx** automation, set the **ShowDesignCompNames** property to **True** so that the project item name (e.g., ACME Product Search System) displays as the top row in the automation design blocks.

Designing the Automation Flow

This automation will set the logic to perform the store location search.

This logic stream navigates the browser to the Locator page, enters the zip code, clicks the Find Store button on the Locator page, and pulls the store location from the Results page and displays it on the Application Bar.

4. Select **View | Object Explorer**.

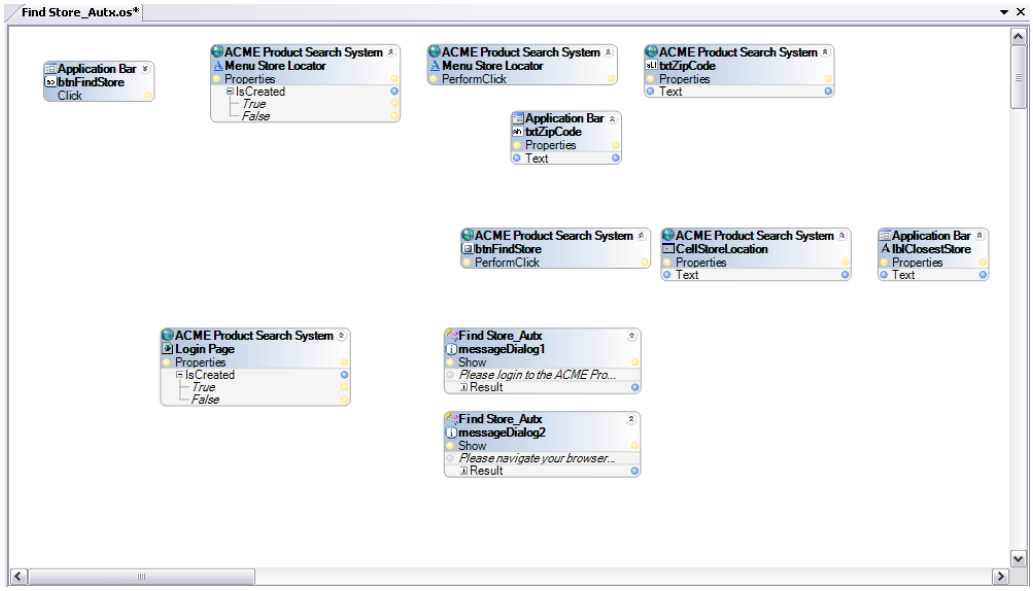
   In the **Object Explorer** window, drag and drop the **properties**, **methods** and/or **events** on to the **Find Store_Autx** design pane for each of the following objects.

   When you complete this step there should be eleven components on the **Find Store_Autx** design pane. One for each property, method and event, plus the two Message dialogs.

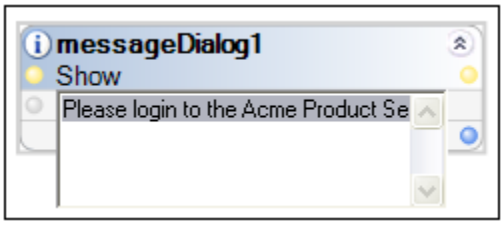
Object	Source Project Item	Property	Event	Method
btnFindStore	Application Bar		Click	
txtZipCode	Application Bar	Text		
lblClosestStore	Application Bar	Text		
Menu Store Locator	ACME Product Search System	IsCreated		PerformClick
txtZipCode	ACME Product Search System	Text		
btnFindStore	ACME Product Search System			PerformClick
cellStoreLocation	ACME Product Search System	Text		
Login Page	ACME Product Search System	IsCreated		
messageDialog*	FromToolbox/Advanced tab – added as MessageDialog1			
messageDialog*	FromToolbox/Advanced tab – added as MessageDialog2			

*When you drag and drop the messageDialog component from the Toolbox window onto the Find Store_Autx design pane, the following dialog box displays. Select the second method (radio button) for this exercise, which allows you to enter the message text.

The **Find Store_Autx** design pane should now contain the following components.

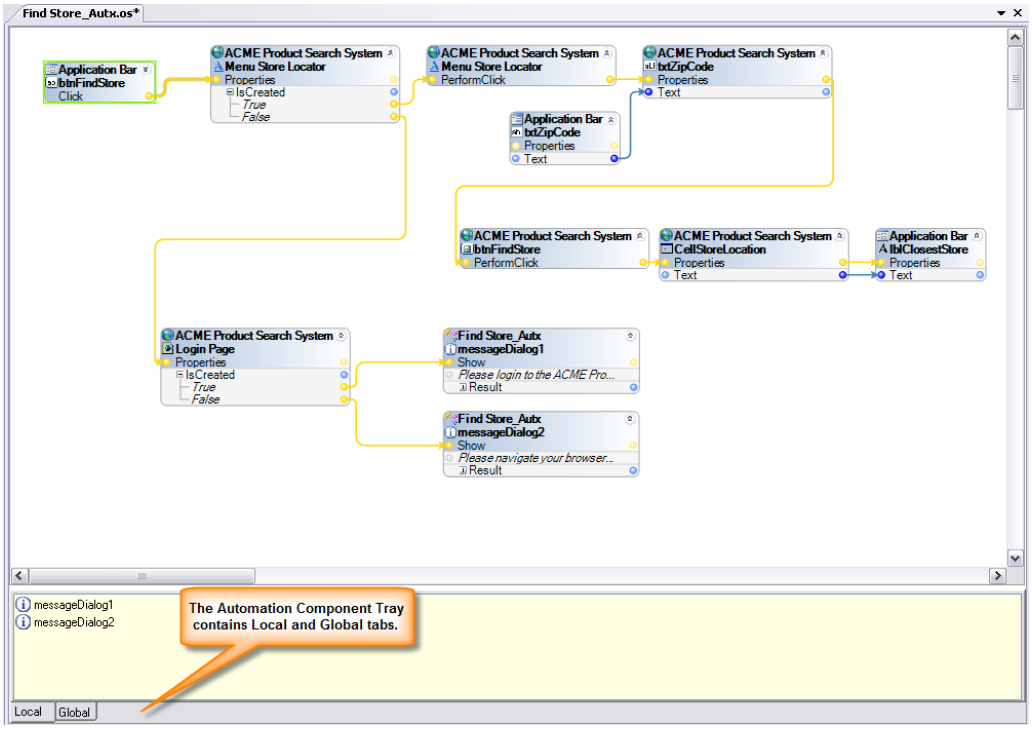


5. Enter the message text for messageDialog1 and messageDialog2.
6. **messageDialog1** will be issued if the Application Bar **Find Store** button is selected and the user is on the **Login** page of the ACME Product Search System. Click the message property on the **messageDialog1** component to open the edit box and then copy and paste the following message text: **Please login to the ACME Product Search System website before attempting to find a store.**



7. **messageDialog2** will be issued if the Application Bar **Find Store** button is selected and the user has navigated the browser outside of the ACME Product Search System. Copy and paste the following message text into the **messageDialog2** component: **Please navigate your browser to the ACME Product Search System and login to the website before attempting to find a store.**

8. Complete the automation by connecting the event and data paths for the automation as shown in the illustration below:



When a reusable component, such as a Message Dialog, is added to an automation it is placed in the **Automation Component** tray.

The Component tray has two tabs: **Local** and **Global**. Global components are visible to all automations in a project via the Object Explorer. Local components are exposed only to the automation to which they belong. Most components are by default added to the Local tab. However, you can easily change a Local component to Global by right-clicking it and selecting **Make Global** from the context menu.

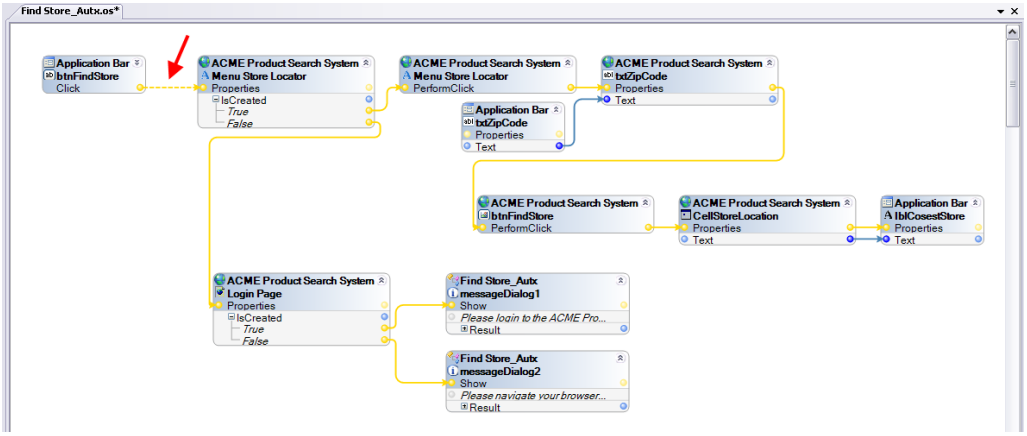
Global components are created when the OpenSpan project starts and exist for the entire time the project is running. They can be used by any automation at any time. These differ from Local components that belong to a specific automation and are created when the automation starts, and exist only while the automation is running.

When to use Local or Global components is sometimes tricky. You need to think whether or not an external automation, actually needs direct access to the component or variable. Making a component Local prevents other automations from using it. In the case of a Global component, its value can be set by an automation and then changed by a second automation prior the first automation finishing with the component. This could result in unexpected behavior of the first automation.

9. Right-click the execution path (yellow line) and select **Asynchronous** from the context menu. Visually, the line changes from solid yellow to perforated.

Remember - When the OpenSpan starts a project, the first thread it creates is the user interface thread. The user interface thread is responsible for displaying any windows forms used in the project. Typically the user interface thread is either painting the user interface to the screen, or waiting for user input. When the user interacts with a windows form or its child controls, Windows sends messages describing the user activity to the user interface thread. The user interface processes hundreds of messages every second. In response to these messages the user interface thread raises windows forms events such as *Click*, *KeyPressed* or *TextChanged*. All windows forms events are raised **synchronously** and execute on the user interface thread

Since the user interface thread is responsible for painting to the screen, any long-running activity that occurs on the user interface thread can block both painting and message processing. This gives the user the impression that the user interface is not responding. To avoid this, any automation that is triggered by a windows form event should execute an **asynchronous** link to release the user interface thread before it interacts with any adapter controls or non-windows forms components (e.g., web services, data access.).



10. Select **File | Save All**.
11. Click the **Start Debugging** icon  from the toolbar.
12. Login to the CRM application and the ACME Product Search System.

To ensure the automation is working as designed:

13. Enter a zip code in the **Zip Code** field on the Application Bar.
14. Click the **Find Store** button on the Application Bar.
15. The zip code is pushed from the Application Bar to the **Locator** web page, the **Find Store** button on the **Locator** page is clicked, and a search for the store begins. The **Results** page displays the store information and that information is pulled from the **Results** page and displayed on the Application Bar.
16. Navigate your browser to a website outside of the ACME Product Search System and then press the Application Bar **Find Store** button - **messageDialog2** is issued.

Exercise Summary

This exercise introduced you to some of the basics of working with Web applications and using the Web Grid in solutions:

- Accessing web pages and checking for page creation
- Activating web links
- Pulling and pushing data to/from web pages

This page intentionally left blank.