PEGA PLATFORM

Pega Robotic Automation Architect Essentials

8.0

Exercise Guide

Pega® ACADEMY

Trademarks

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.
One Rogers Street
Cambridge, MA 02142-1209
USA
Phone: 617-374-9600
Fax: (617) 374-9620
www.pega.com

DOCUMENT: Pega Robotic Automation Architect Essentials Exercise Guide
SOFTWARE VERSION: Pega 8.0
UPDATED: 02 2019

COURSE INTRODUCTION

This lesson group includes the following lessons:

- Before you begin
- Completing the exercises

Before you begin

Completing the exercises

When learning new concepts or skills, there is no substitute for learning by doing. This course includes exercises that provide practical, hands-on experience to help you apply your new skills immediately. The exercises help reinforce the learning objectives of each lesson.

Exercise environment

Download the following files from the related content section:

File	Purpose
Pega Robotic Automation Training Exercise Files	Download and use the files to complete the exercises in the course
Pega Robotic Automation Architect Essentials Exercise Guide	PDF of all exercises in the course for use as reference material
Pega Robotic Automation Architect Essentials Student Guide	PDF of course content for use as reference material
Pega Robotic Automation Studio 8.0 SP1 Installation Instructions	Use to install Pega Robotic Automation Studio Standalone and to confirm the minimum requirements for installation
Pega Robotic Automation Studio 8.0 SP1 Plug In Installation Instructions	Use to install Pega Robotic Automation Studio Plug In and to confirm the minimum requirements for installation

When you click the download the exercise software links, you agree to the terms of this agreement.

You must have administrative rights to your laptop to install the software and its pre-requisites correctly.

If Microsoft Visual Studio is already installed on your computer you cannot install the standalone version of Pega Robotic Automation Studio used for this training course. You must download the Pega Robotic Automation Studio Plug In version. Read the Pega Robotic Automation Studio Plug In 8.0 SP1 Installation Instructions to confirm your version of Microsoft Visual Studio is compatible with the Plug In version.

Download Pega Robotic Automation Studio 8.0 SP1.

Download Pega Robotic Automation Studio 8.0 SP1 Plug In .

Download Pega Robotic Automation OCR.

WINDOWS INTEGRATION

This lesson group includes the following lessons:

- Getting started with Pega Robotic Automation Studio
- Developing solutions and projects
- Working with windows adapters

- Working with automations
- Working with multiple app instances

Getting started with Pega Robotic Automation Studio

Exercise: Organizing the workspace

Your assignment

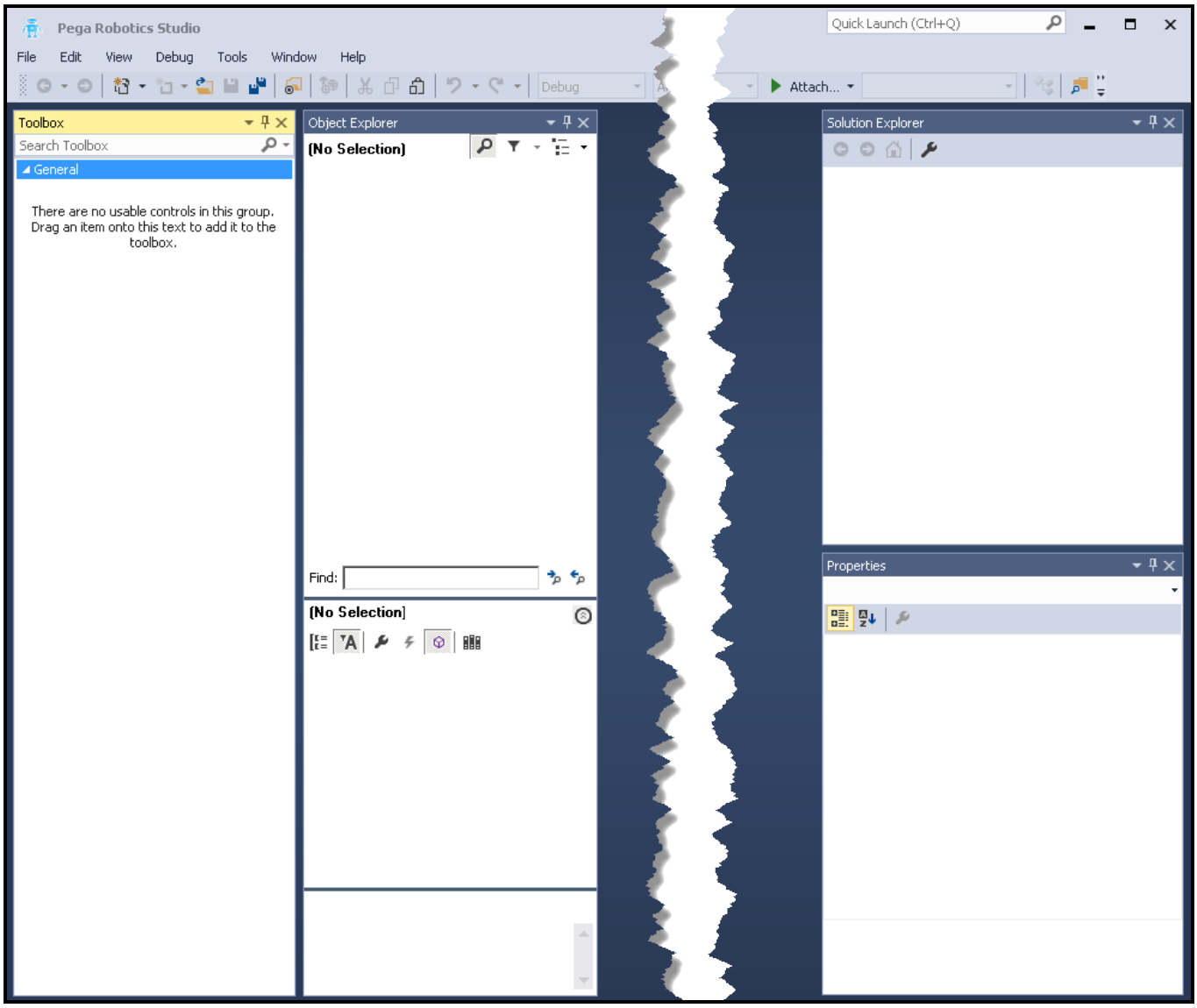
In this exercise, you organize the main Tool windows used for this course: Solution Explorer, Object Explorer, Properties, and Toolbox. You also enable the Auto Hide for the Toolbox window.

Detailed steps

Follow these steps to organize the workspace.

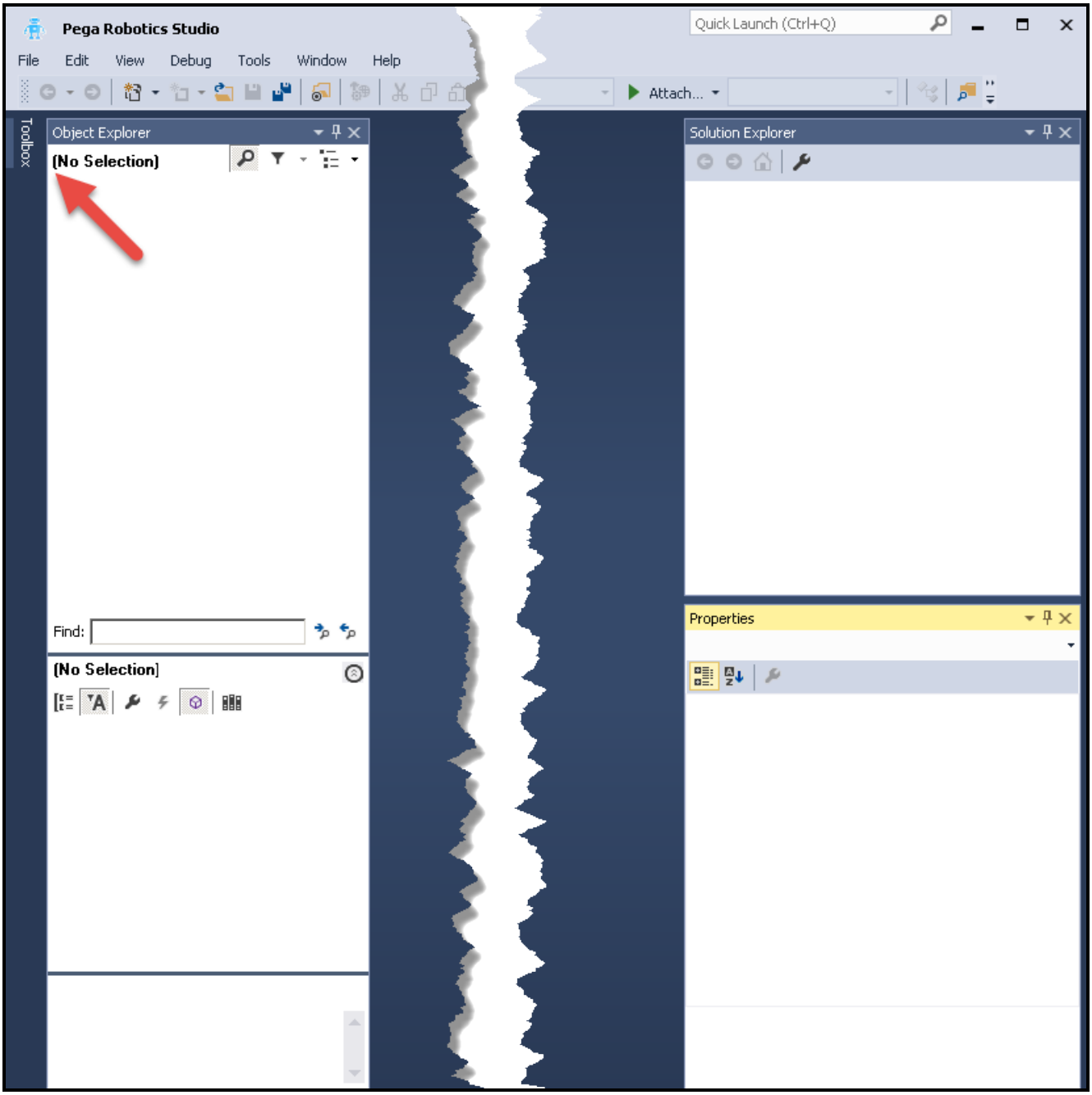
1. With Studio running, make sure the following tool windows are open: Solution Explorer, Object Explorer, Toolbox, and Properties. If any of these windows are not displaying, use the View menu to select which windows are missing. If other windows are displayed, close them.
2. On the right-side of the workspace, dock the Solution Explorer and the Properties windows.
3. On the left-side of the workspace, dock the Object Explorer and the Toolbox windows.

4. Position the Object Explorer and Toolbox as shown in the following image.



5. Auto hide the Toolbox window.

Your workspace and tool window placement should look like the following image.



Developing solutions and projects

Exercise: Creating a solution and project

Scenario

Based on the business case, you create a project and a corresponding solution. The project name is CRMAdapter, and the solution name is TrainingCertification.

Your Assignment

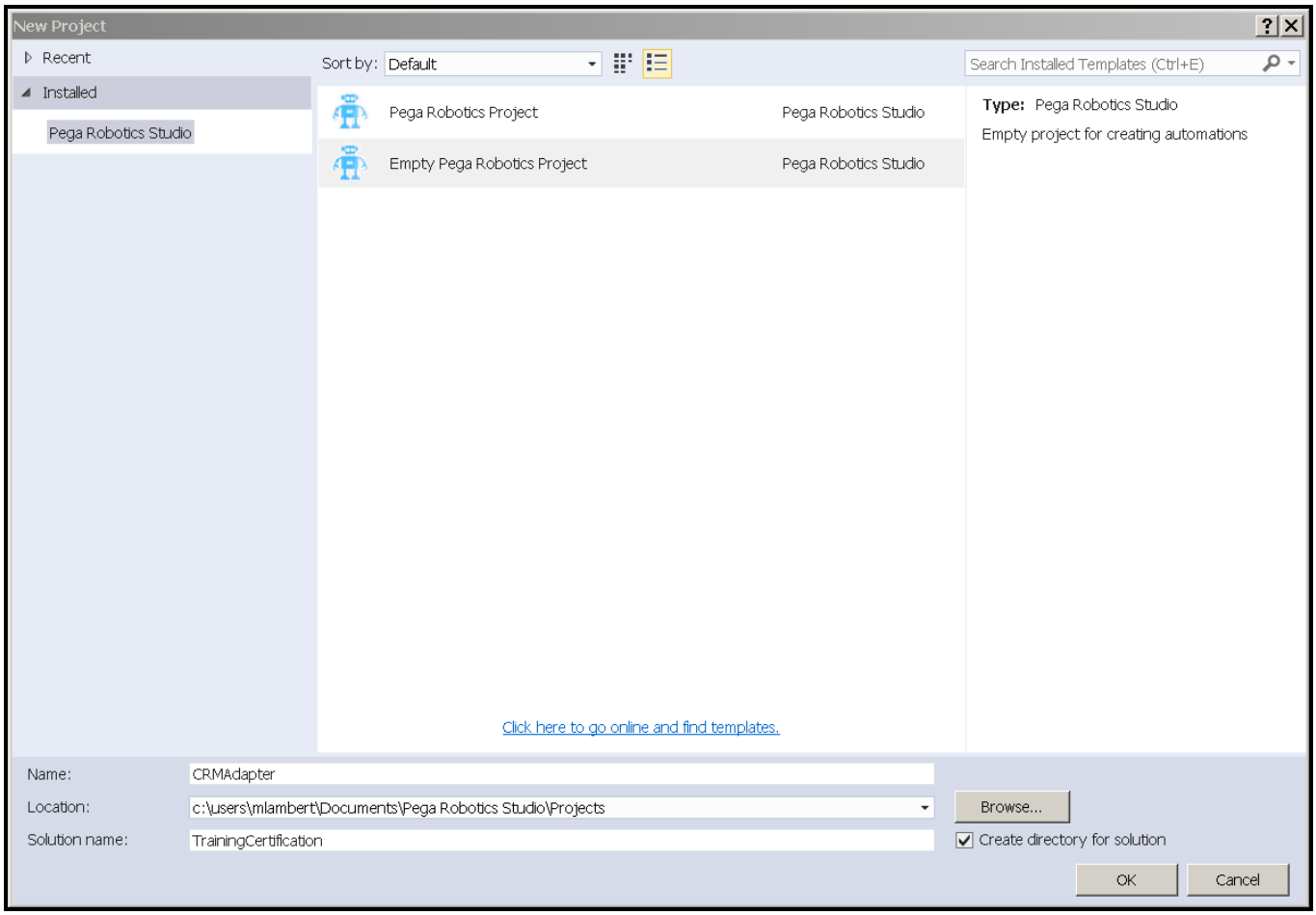
Develop automations for the business case by creating a project and solution.

Detailed Steps

Follow these steps to create a project and solution.

1. Start Studio.
2. Select **File > New > Project**.
3. Select **Empty Pega Robotics Project**.
4. In the **Name** field, enter **CRMAdapter** .
5. In the **Solution Name** field, enter **TrainingCertification**.
6. Accept the default folder location.

7. Click **OK**. The new solution and the project display in the Solution Explorer window.



Working with windows adapters

Exercise: Adding a windows adapter

Scenario

Based on the business case, you have successfully added the CRMAdapter project and TrainingCertificaiton solution. Now, you are ready to add the CRM application to your solution by adding a windows adapter to launch the CRM application.

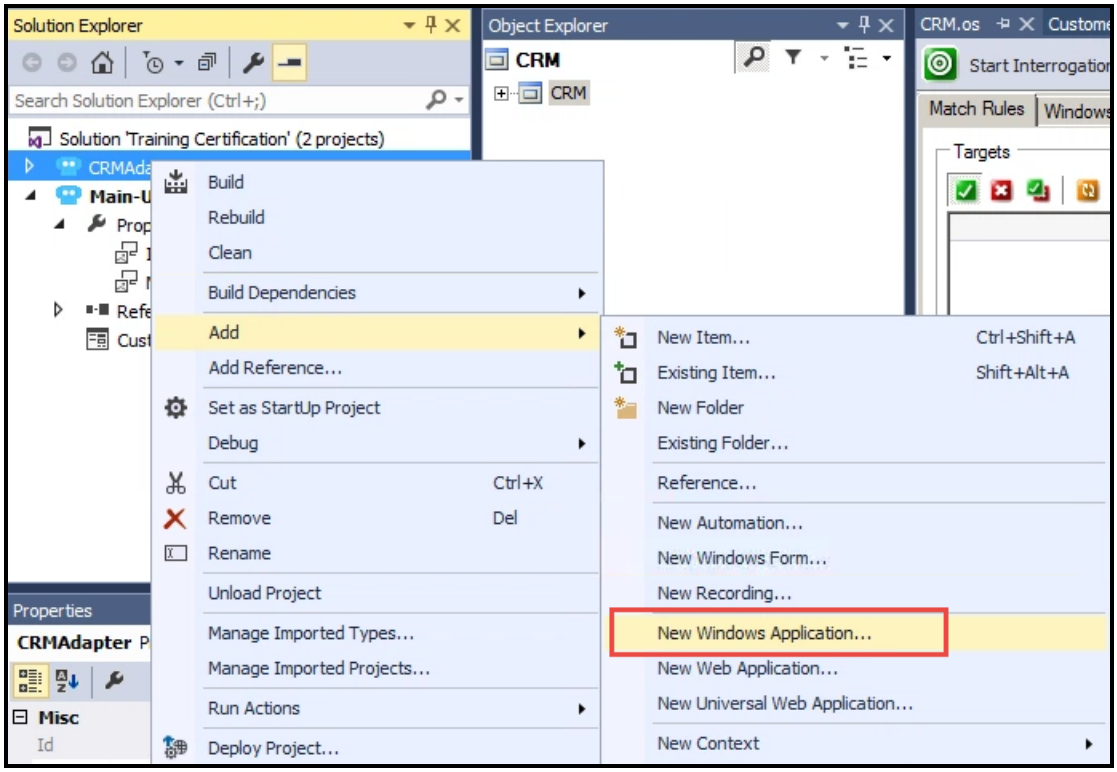
Your assignment

Now, you need to add a Windows adapter (application) to your solution. You will set the adapter properties and use the installed training CRM application for the windows application. If you have not installed the application, refer to the Before you begin lesson for instructions.

Detailed steps

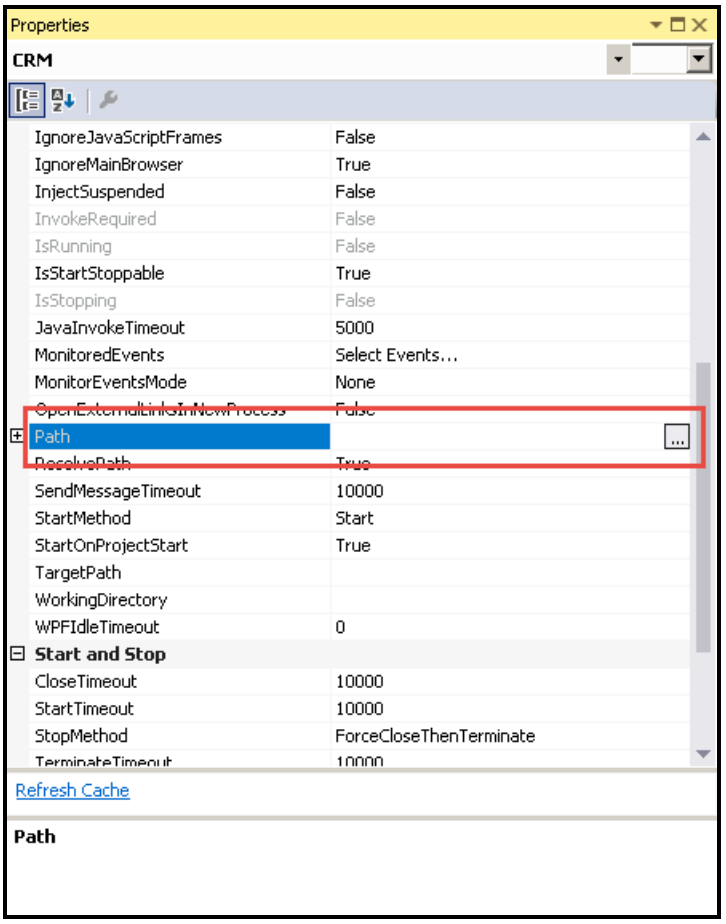
Follow these steps to add a new project and a windows adapter.

1. In Solution Explorer, right-click the **CRMAdapter** project.
2. Select **Add > New Windows Application**. The Add New Item dialog appears.



3. Enter **CRM** in the Name field.
4. Click **Add**. The CRM.os adapter displays in the Solution Explorer and the CRM.os designer window opens in the IDE.
5. In the Object Explorer, click the **CRM** object. The Properties window displays the adapter properties.
6. In the Properties window, locate the Path property.
7. Click in the blank property field. An ellipses button appears on the right of the blank property field.

8. Click the **Ellipses** button.



9. Browse to the installation location of the training CRM application.
10. Click **Open**.
11. Select **File > Save All.** This saves the adapter property value edits.

Exercise: Interrogating a windows application

Scenario

The business case states that the solution login to the application and display the CRM account information on a user interface. The first step of gaining access to the information in CRM is interrogating the required CRM account information and login controls.

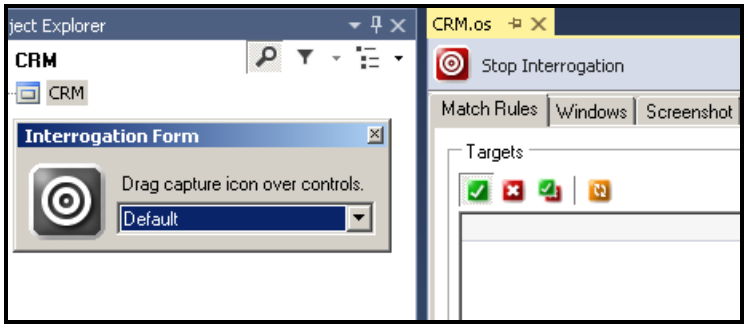
Your assignment

Interrogate the login controls, toolbar buttons, customer account information, and menu items in the CRM application using the three methods: Interrogation target, Create Control, and Add Menu items. Rename the objects in the Properties window during the interrogation process.

Detailed steps

Interrogate using the Interrogation target

1. With the CRM adapter project item open in the Designer Window, click the **Start Interrogation** button. Studio launches the Training CRM application (as specified in the Path property) and opens an Interrogation Form dialog.

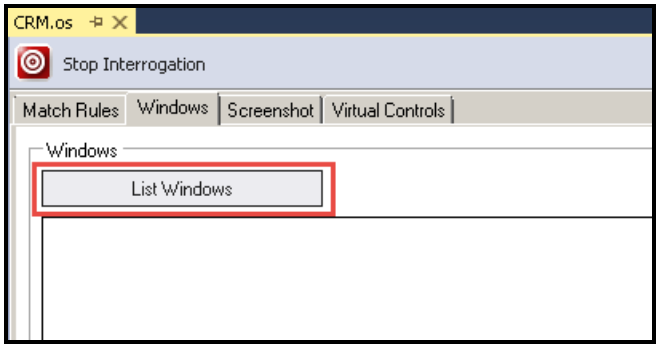


2. Click the **Target** icon on the Interrogator Form dialog and drag it to the **Login** button.

3. Release the mouse button when the field has a rectangular highlight around it.

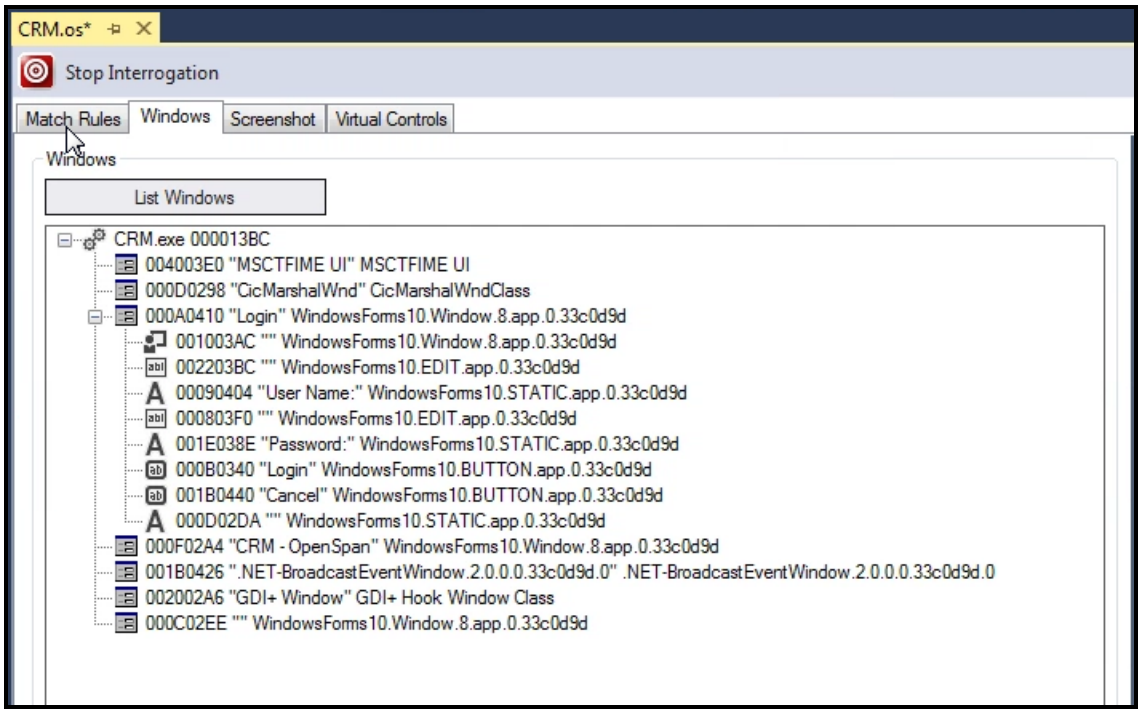
4. Interrogate the **User Name** and **Password** fields.

5. Click the **Windows** tab on the **CRM.os Designer** window.
6. Click **List Windows**.

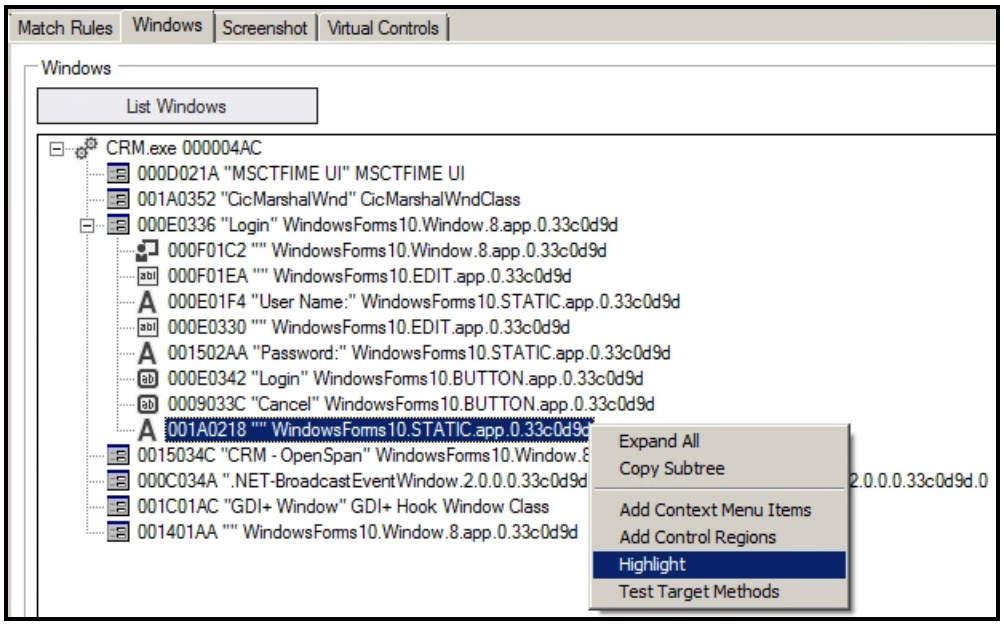


7. Expand the **CRM.exe** item in the tree view of the Windows group box.

8. Expand the **“Login” Window Form**.

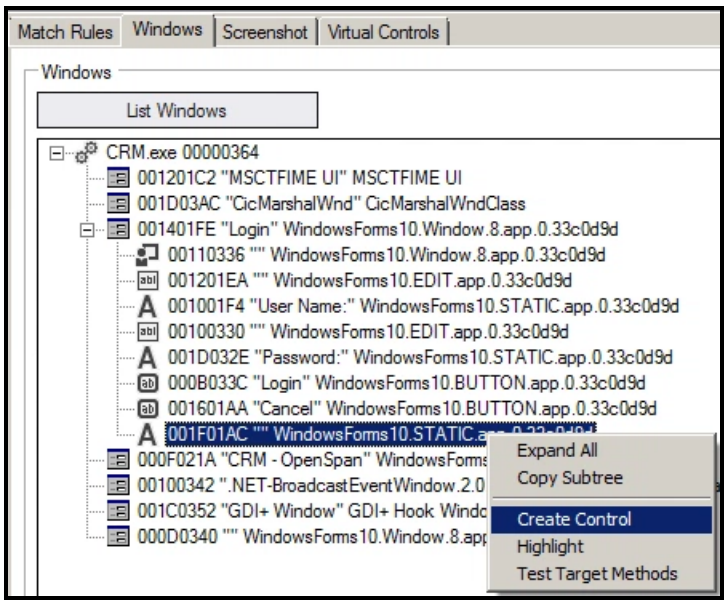


9. Select the last object in the list, which is the label assigned to the CRM application version information.
10. Right-click the **Label** and select **Highlight**.



Interrogate using Create Control

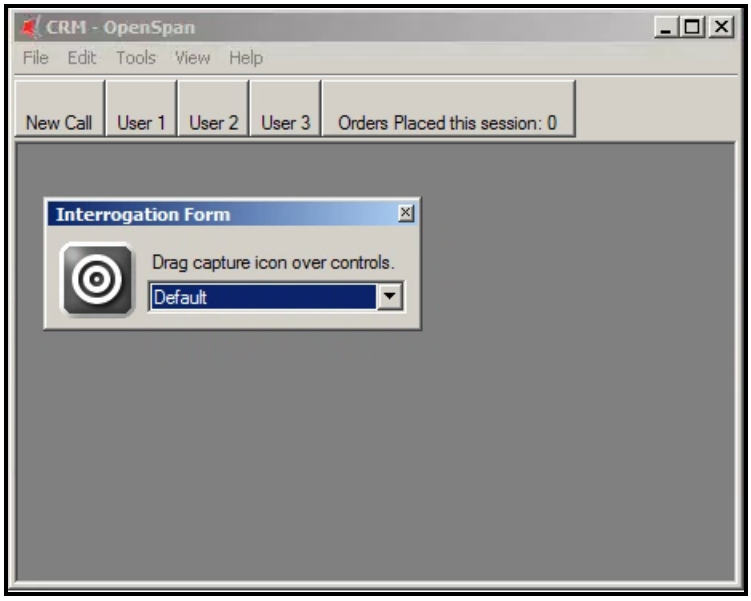
1. Return to the Windows list. Right-click **Label** and select **Create Control** to add this target to the project.



2. Highlight each object in the Object Hierarchy and rename the Design Name in the Properties window as follows:

Object Name	Design Name
Login	CRMfrmLogin
btnbtnLogin	CRMbtnLogin
lblgdilblVersion	CRMlblVersion
txttxtCredentials	CRMtxtUserName
txttxtPassword	CRMtxtPassword

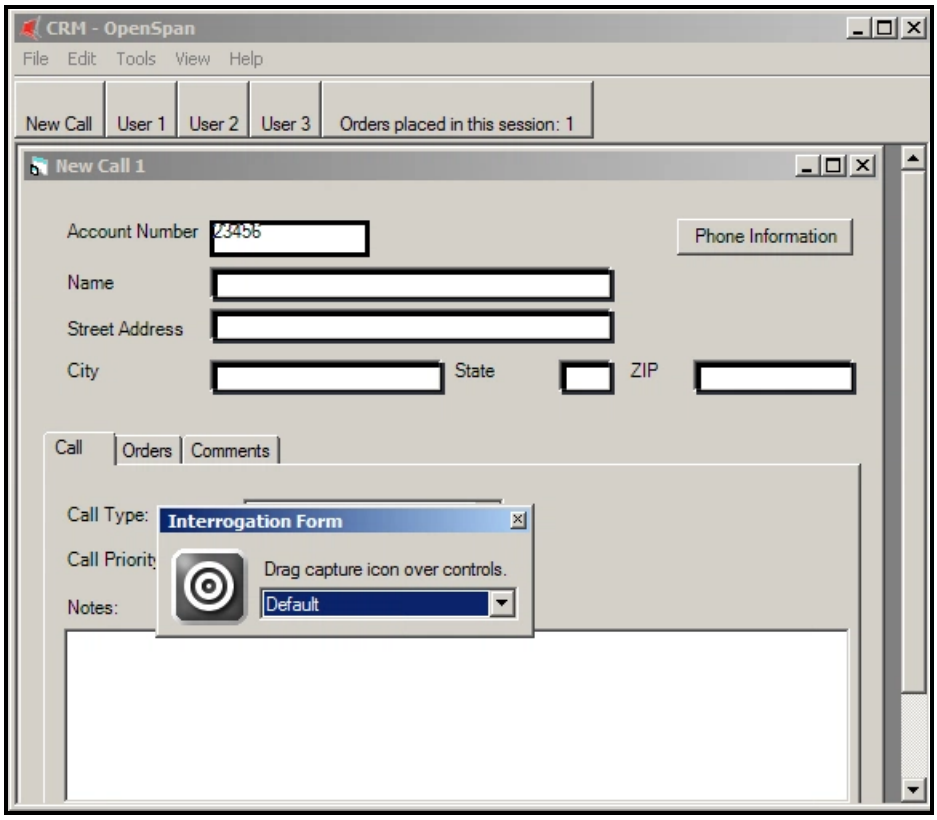
3. Click the **Login** button. The CRM window appears.
4. Interrogate these CRM buttons: New Call, User 1, User 2, and User 3.



5. Highlight the object and rename the Design Name as follows:

Object Name	Design Name
CRM_Subtraction_OpenSpan	CRMfrmMain
Toolbar1	CRMToolBar
btnbntUser1	CRMbtnUser1
btnbntUser2	CRMbtnUser2
btnbtnNewCall	CRMbtnNewCall
btnbtnUser3	CRMbtnUser3

6. Click the **New Call** button. This opens the New Call 1 window. Your display settings may black out the Account Number field. Change the display to Windows Classic, if desired.
7. Interrogate these text field targets:
   - Account Number
   - Name
   - Street Address
   - City
   - State
   - ZIP



8. In the Object Explorer, highlight each object and rename the Design Name as follows:

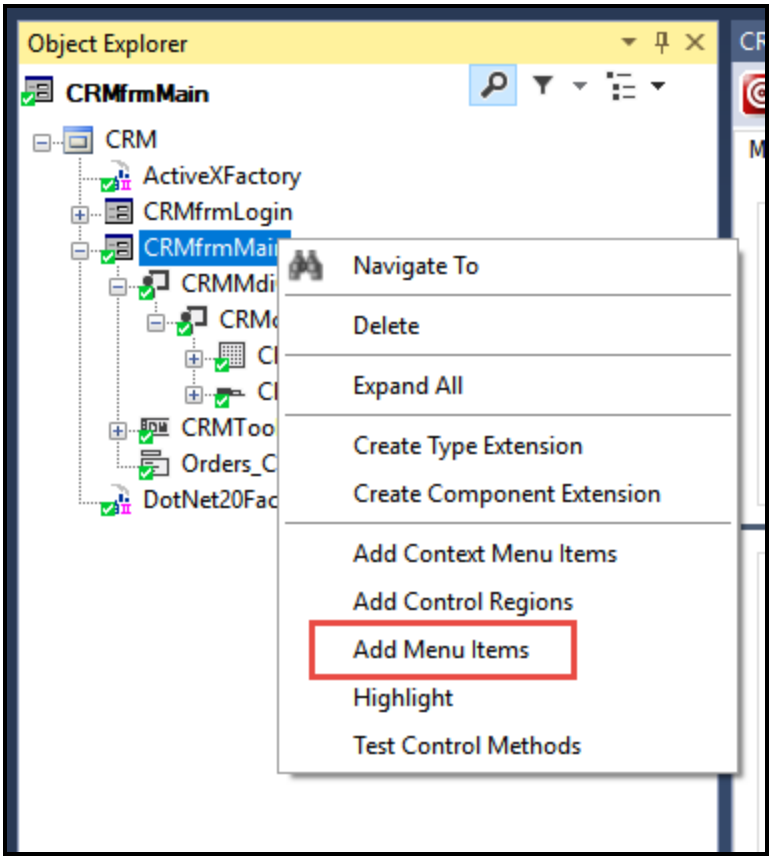
Object Name	Design Name
MdiClient	CRMMDIClient
CRMchild	CRMChild
pnlCustomerInfo	CRMpnlCustInfo
lbllblAcctNum	CRMlblAcctNum
txttxtCity	CRMtxtCity
txttxtName	CRMtxtName
txttxtState	CRMtxtState
txttxtStreetAdd	CRMtxtStreetAdd
txttxtZip	CRMtxtZip

9. Click the **Comments** tab.
10. Interrogate the Comments field.
11. In the Object Explorer, highlight the object and rename the Design Name as follows:

Object Name	Design Name
tabMain	CRMtabMain
tabComments	CRMtabComments
txttxtComments	CRMtxtComments

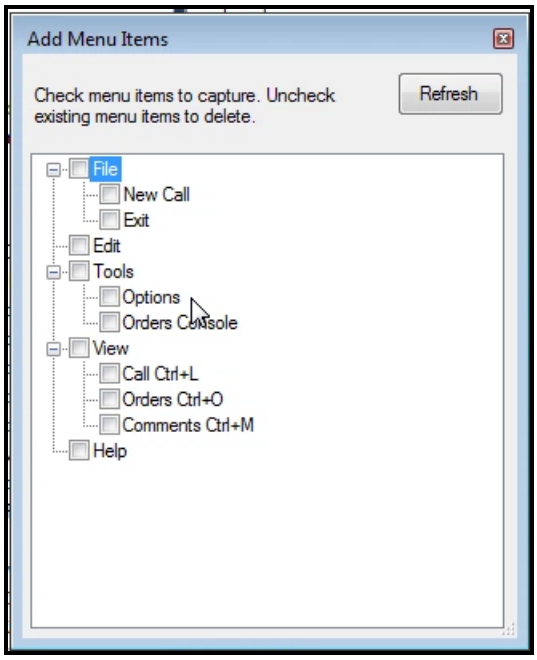
Interrogate using Add Menu Items

1. In Object Explorer, highlight the **CRMfrmMain** object.
2. Right-click and select **Add Menu Items**. The Add Menu Item window appears.



3. Select **Orders Console** from the **Add Menu Items** dialog.

4. Click the **X** to close the dialog and save your selection.



5. Stop the interrogation process.
6. Click **File > Save**. The asterisk on the CRM.os tab disappears.

Working with automations

Exercise: Creating an automation

Scenario

Based on the business case, you have created the necessary projects and items to begin the first portion of the process. The first goal was for the solution to login into the CRM application and accept the first call.

Your assignment

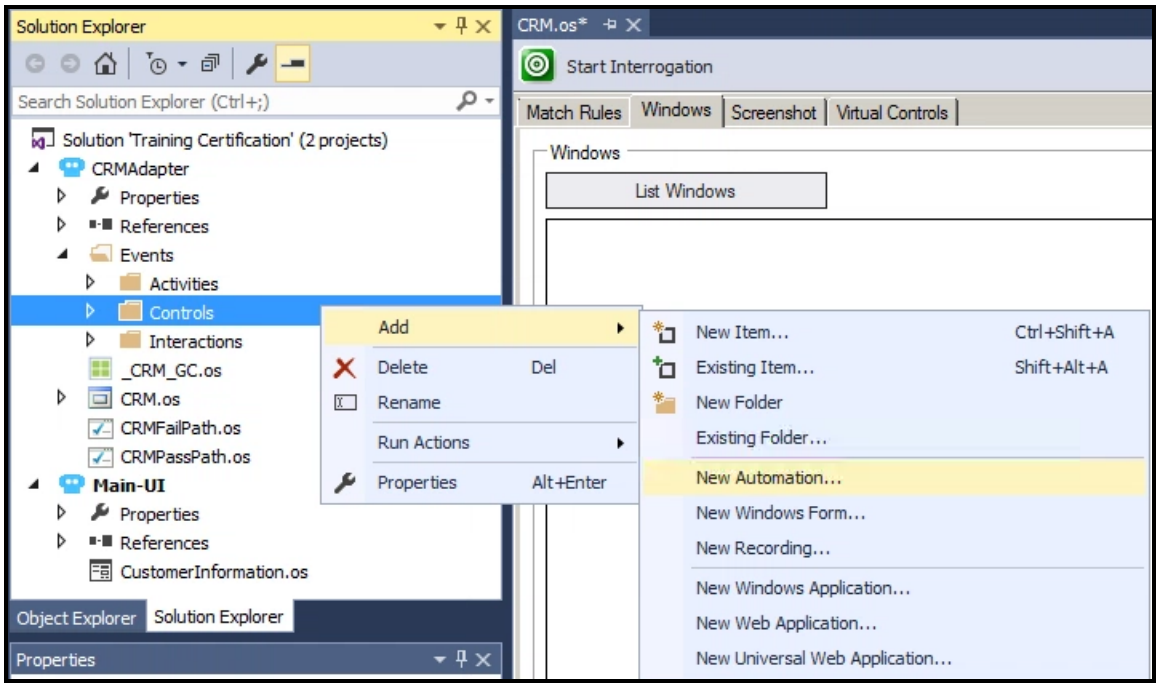
In this part of the solution, organize the CRMAdapter project with folders to manage the automations. Then create an automation to fulfill the business case to login and open the first call.

Detailed steps

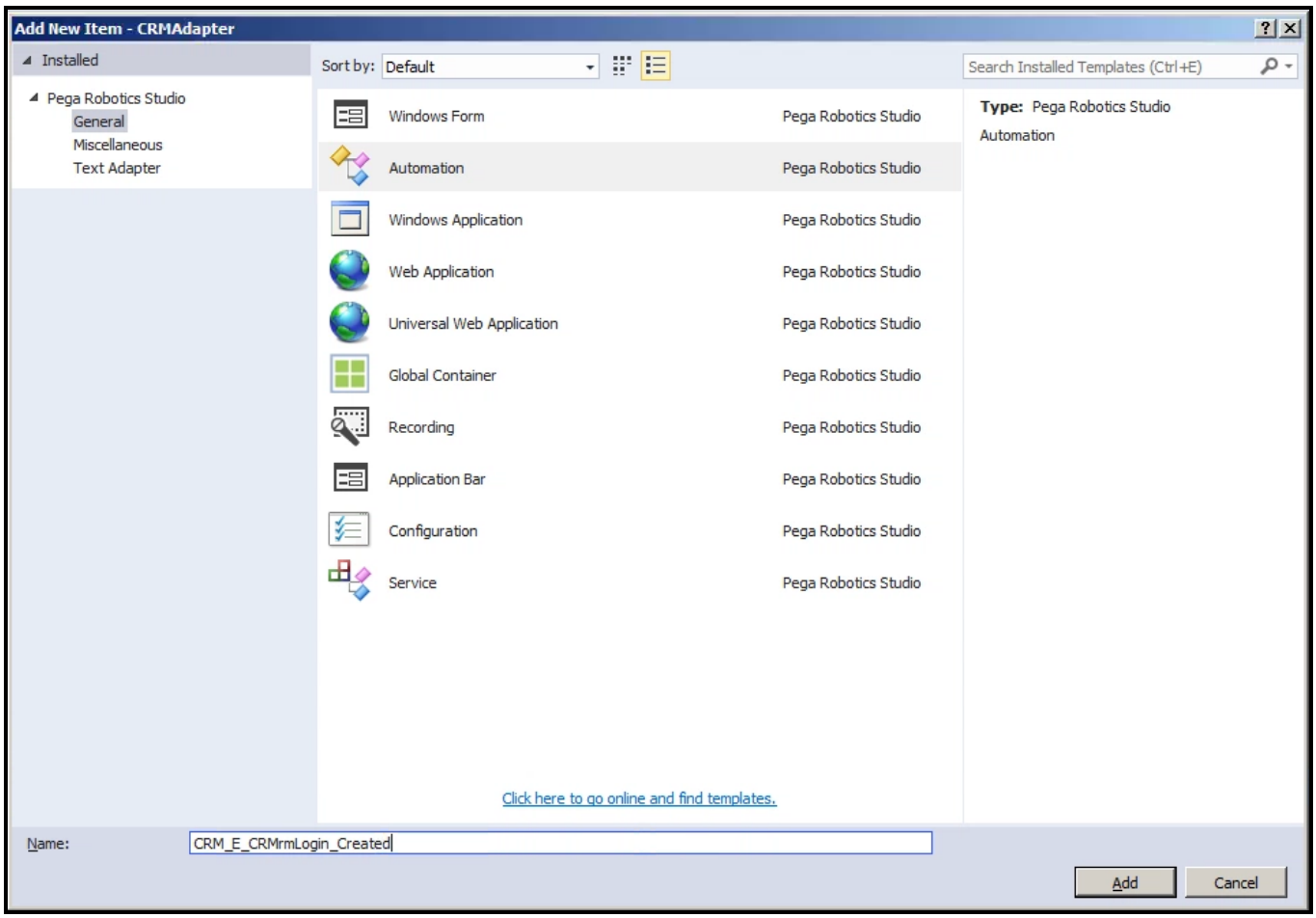
Follow these steps to create the CRM_E_CRMfrmLogin_Created automation.

1. In the Solution Explorer, right-click the **CRMAdapter** project and select **Add > New Folder**.
2. In the Solution Explorer, enter **Events** in the highlighted folder.
3. In the Solution Explorer, right-click the **Events** folder and select **Add > New Folder**.
4. In the Solution Explorer, enter **Controls** in the highlighted folder.

5. In Solution Explorer, right-click the **Controls** folder and select **Add > New Automation.**

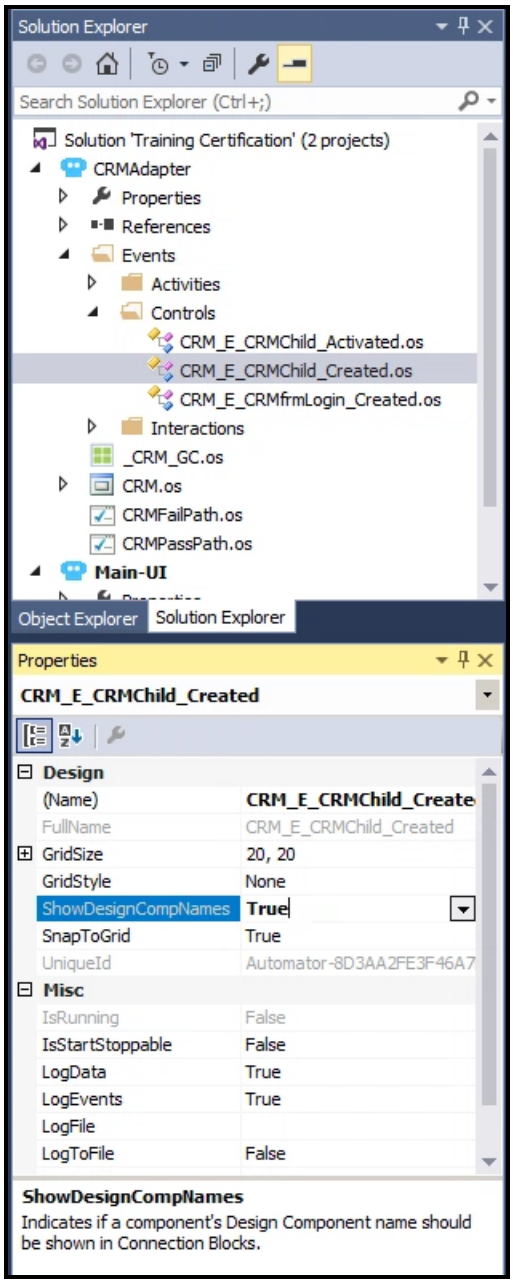


6. In the **Name** field, enter **CRM_E_CRMfrmLogin_Created**.
7. Click **Add.** The automation opens in a Designer window and the Properties window displays the properties for the automation.

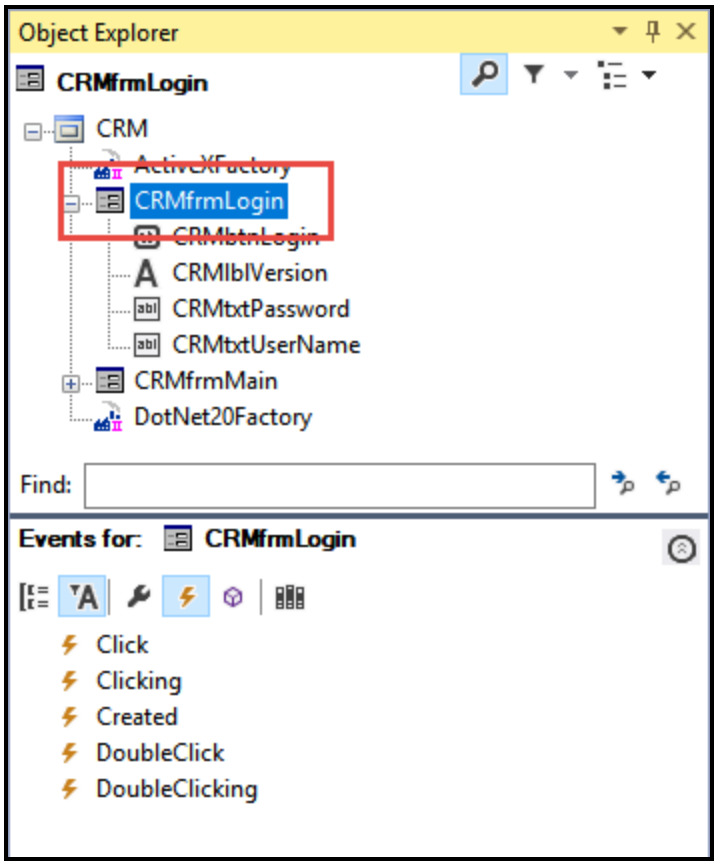


8. In the Properties window, change the **ShowDesignCompNames** from False to **True**.
9. In the Properties window, right-click the **ShowDesignCompNames** text and select **Default Value**... from the context menu.
10. On the Automator popup window, select **True**.

11. Click **OK**. This sets default value to true for all automations from this point forward.

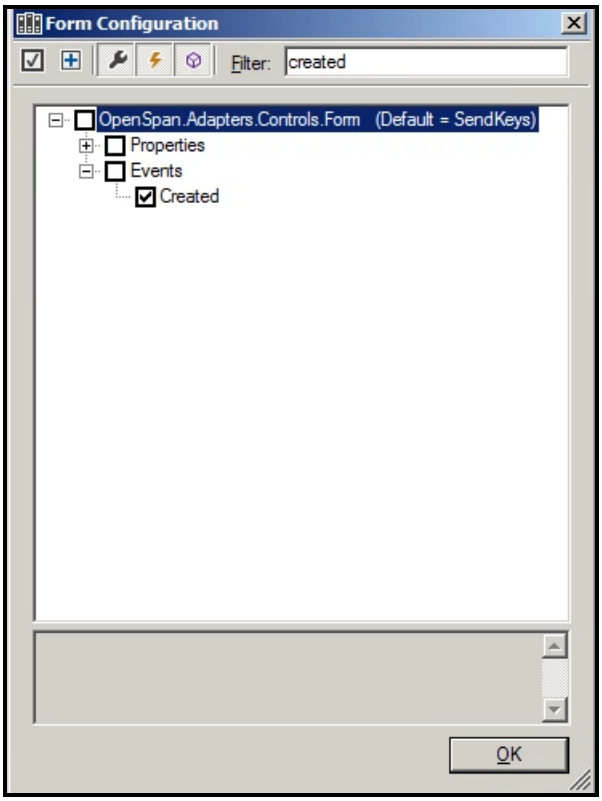


12. In Object Hierarchy, select the **CRMfrmLogin** control.

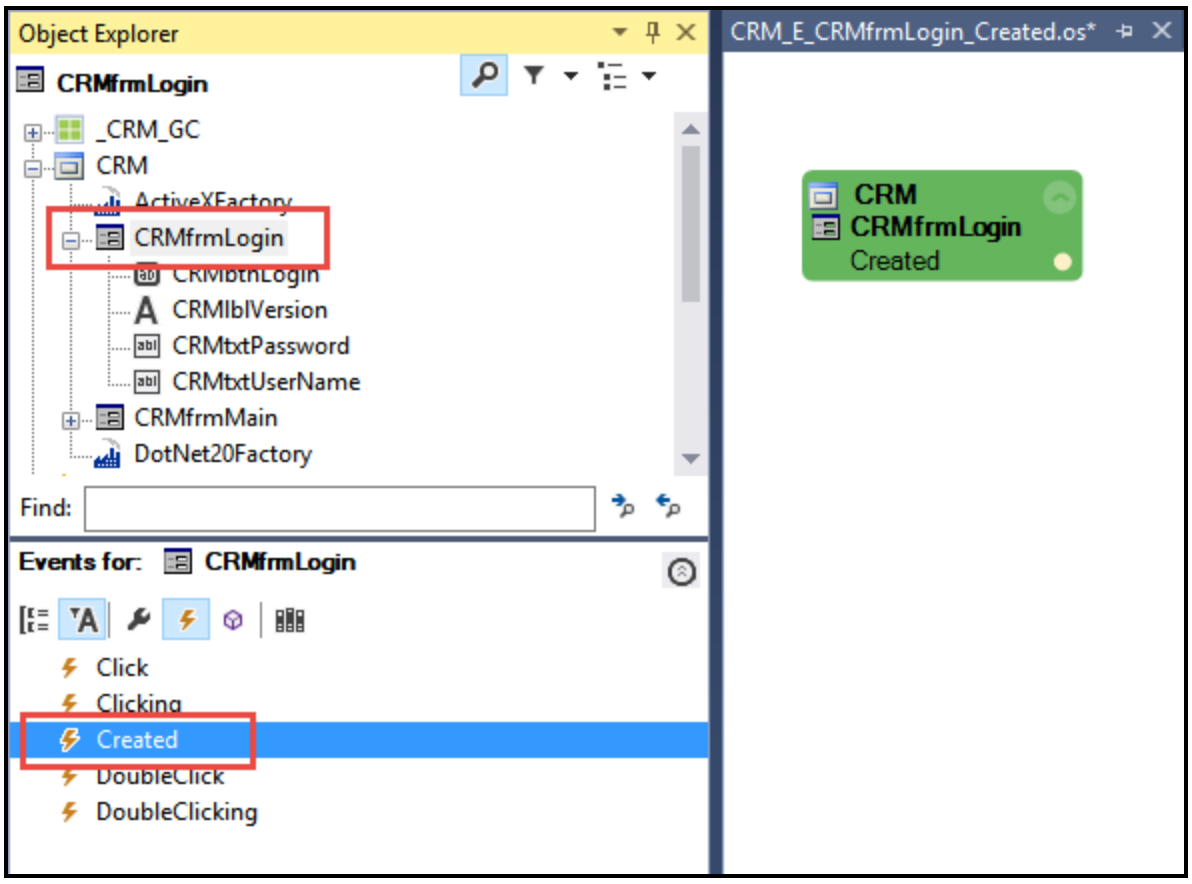


13. In Object Inspector, click the **Show Events Only** (lightning bolt) icon. The available events for the CRMfrmLogin control display.
14. Click the **Configure Type** (books) icon. The Form Configuration window displays.
15. In the **Filter** field, enter **Created**. The options filter to display only those containing the entered text.
16. Expand **Events**.
17. Check **Created**.

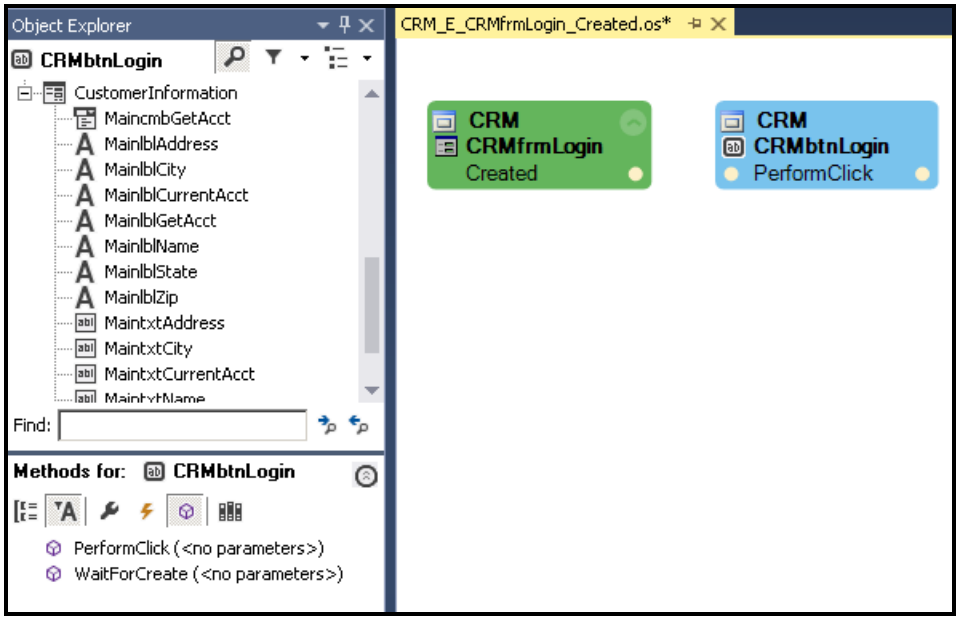
18. Click **OK** to add the event to the Object Inspector.



19. In the Object Inspector with Show Events Only selected, click the **Created** event, and drag and drop it to the CRM_E_CRMfrmLogin_Created automation.

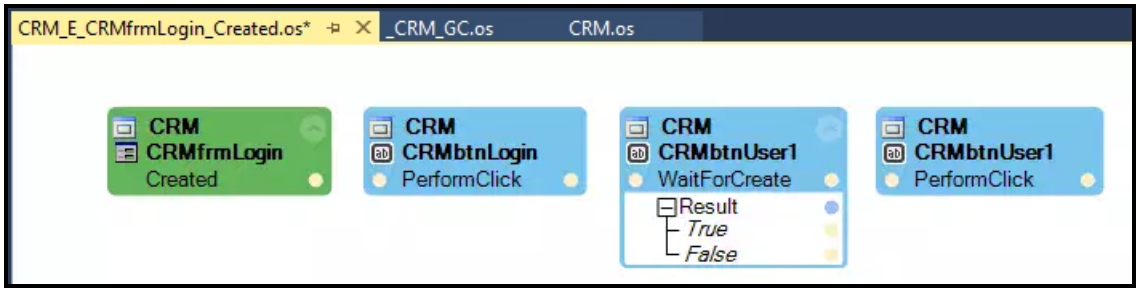


20. In the Object Hierarchy, highlight the **CRMbtnLogin** control.
21. In the Object Inspector, click the **Show Methods Only** (box) icon.
22. Select the **PerformClick** method, and drag and drop it to the automation.

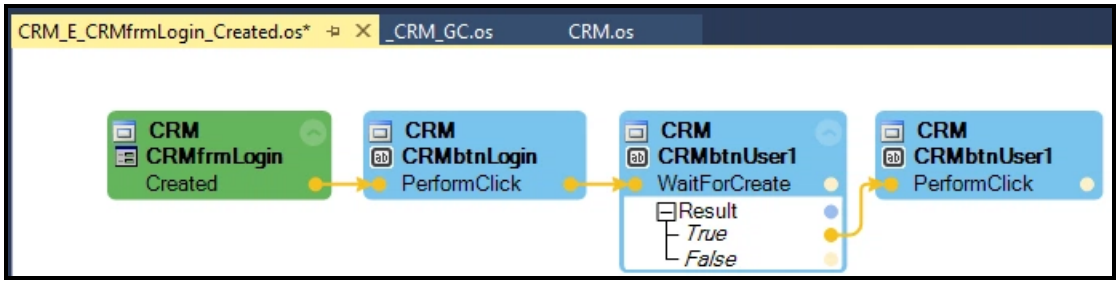


23. In the Object Hierarchy, highlight the **CRMbtnUser1** control.
24. In the Object Inspector, click the **Configure Type** icon. The Button Configuration window displays.

25. Add the WaitForCreate method to the Object Inspector.
26. In the Object Inspector with the Show Methods Only selected, click and drag the **WaitForCreate (<no parameters>)** method to the automation.
27. On the WaitForCreate design block, expand the **Result** to display a Boolean response.
28. In the Object Hierarchy, highlight the **CRMbtnUser1** control.
29. Select the **Show Methods Only** (box) icon, and drag and drop the **PerformClick** method to the automation.
30. Select **File > Save All** to save your work. The automation should be ordered like the following image:



31. Connect the design blocks as shown in the following image:



32. Press **F5**. The debugging process starts. The customer information window displays; the CRM application begins, with the log in window closing and the User 1 button clicked to display John Smith.

Working with multiple application instances

Exercise: Setting the UseKeys Property

Scenario

Based on the business case, end-users may work with multiple customers simultaneously. Since the CRM application contains our customer and account information, you must remember that end users may open various customer windows to ensure data integrity.

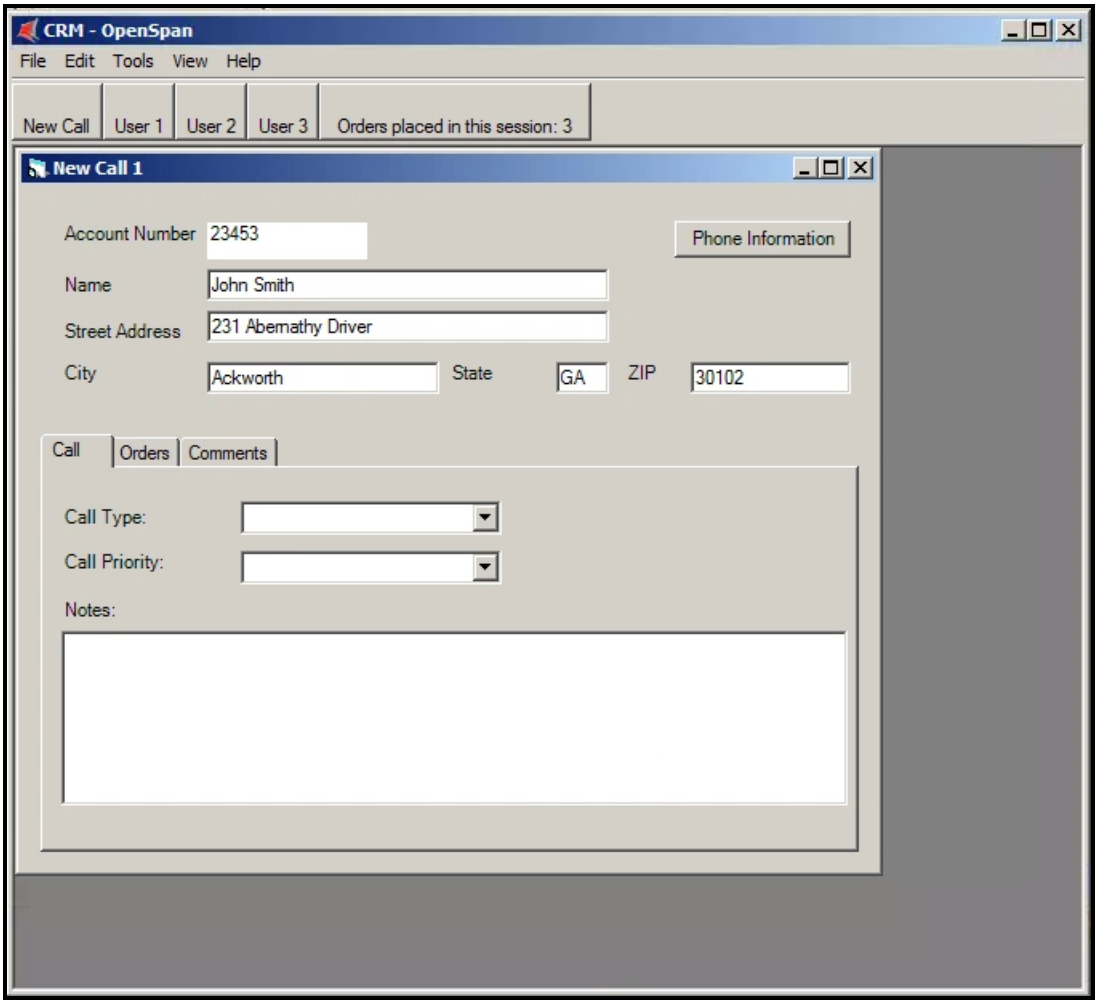
Your assignment

In this part of the solution, set the UseKeys property of the CRMChild control.

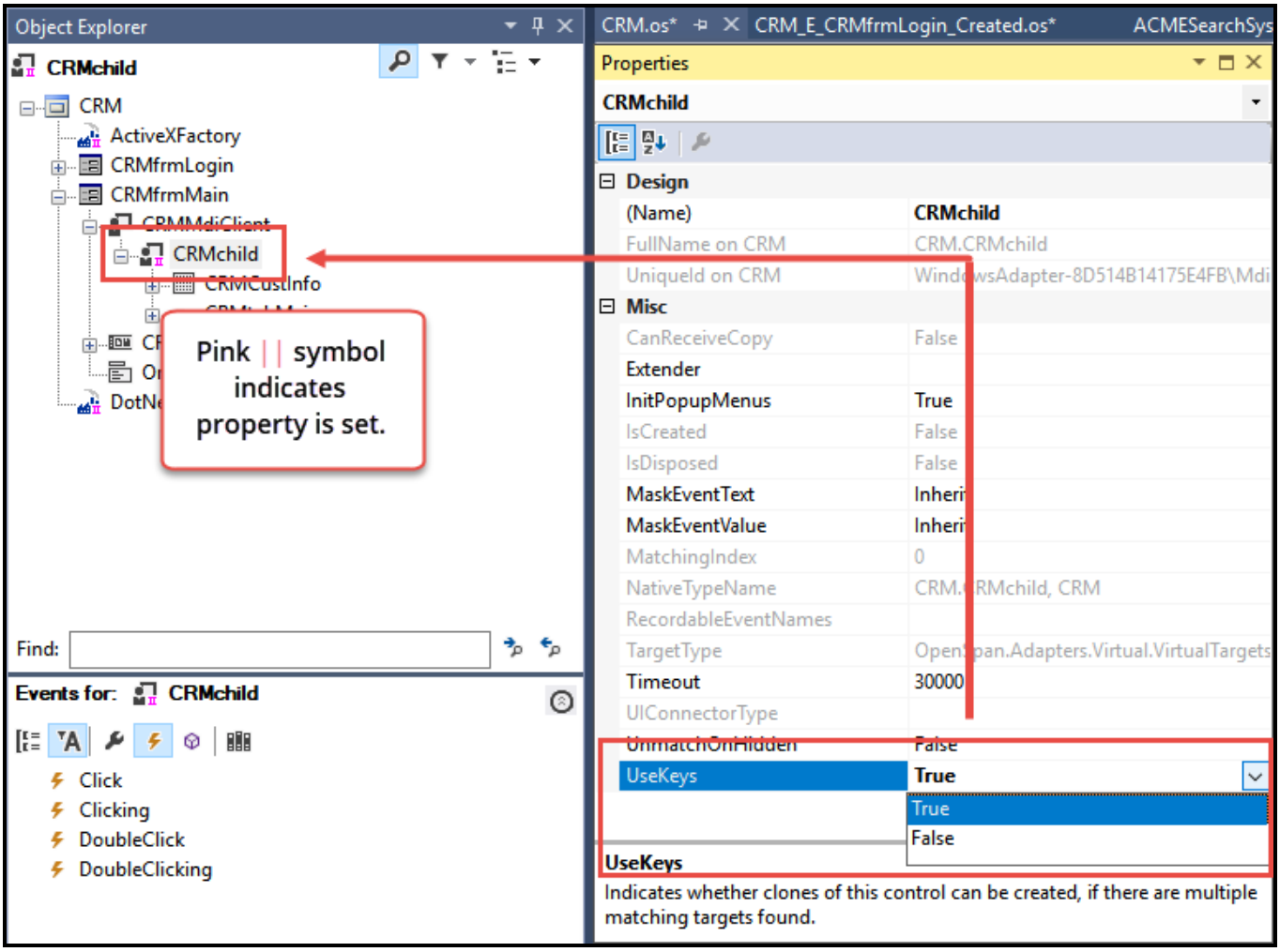
Detailed steps

Follow these steps to set the UseKey property.

1. In the Solution Explorer, double click **CRM.os** to open it in a designer window.
2. On the CRM.os designer tab, click **Start Interrogation**. The CRM application launches and the Login window appears.
3. On the Login window, click **Login**. The CRM application displays.
4. On the CRM window, click **New Call**. A new CRMChild window appears and the controls for the window match, as indicated by the green check mark (√) in the Object Explorer.



5. In Object Explorer, highlight the **CRMChild** control. You may need to expand the CRMMDIClient control in the Object Hierarchy.
6. In the Properties window, double click the **UseKeys** property to change it from False to True. Notice that the CRMChild item in Object Explorer now has a pink pipe (II) symbol, indicating the enabled Key property for that control.



7. Stop the interrogation process. The Interrogation Form and CRM application close.
8. Select **File > Save** to save the UseKeys property change.

DEBUGGING AND DIAGNOSTICS

This lesson group includes the following lessons:

- Debugging
- Diagnostics
- Error handling

Debugging and Diagnostics

Exercise: Adding a diagnostic log to an automation

Scenario

An automation requires a diagnostic logging component to capture a custom message each time users access the application.

Your Assignment

For this exercise, modify the CRM_E_CRMfrmLogin_Created automation and add a diagnostic log component between the CRMfrmLogin.Created event and the CRMbtnLogin.PerformClick method. Ensure the Runtime Diagnostics File Publisher is set to On.

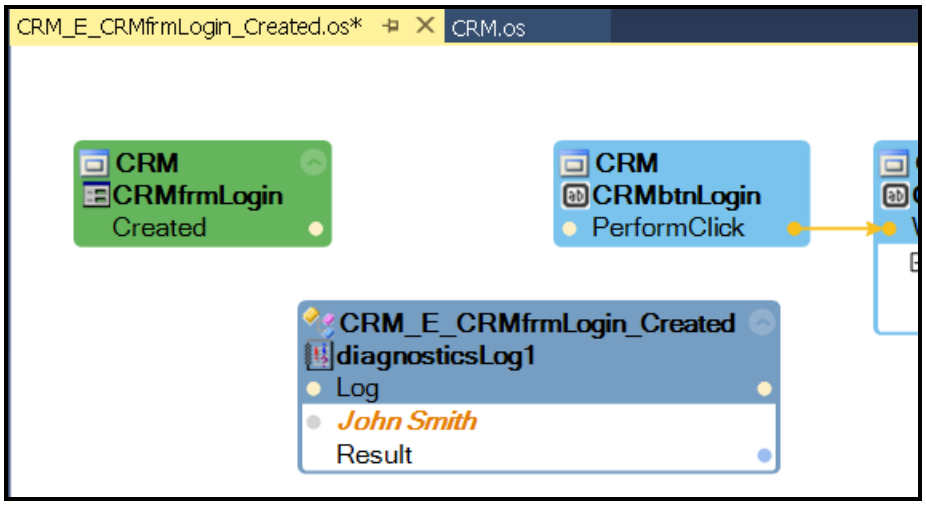
Detailed Steps

Adding and configuring the diagnostic logging component

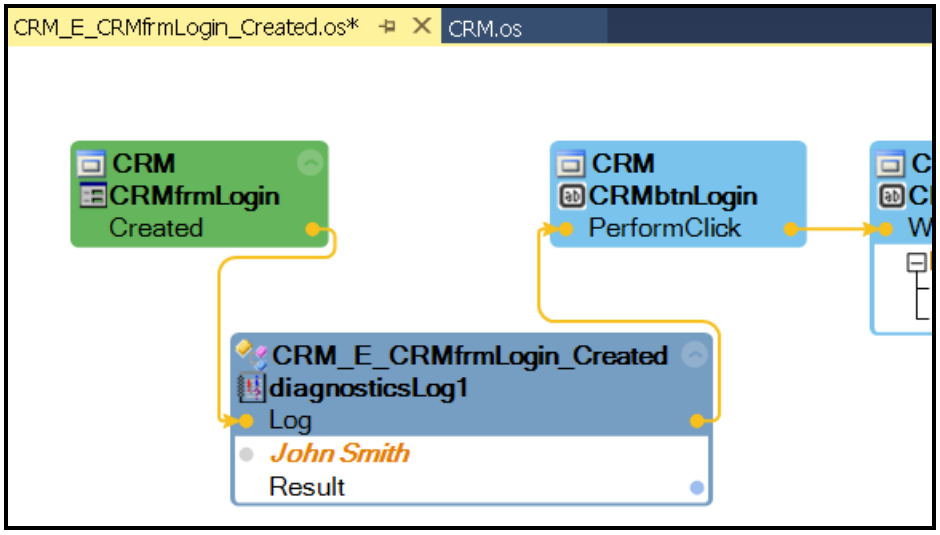
Follow these steps to add and configure the diagnostic log component.

1. From the Solution Explorer, double click the **CRM_E_CRMfrmLogin_Created** automation to open the automation in a design window.
2. Delete the automation link between the CRMfrmLogin.Created event design block and the CRMbtnLogin.PerformClick method.
3. From the Advanced section in the Toolbox, click and drag a **DiagnosticsLog** component to the automation.
4. In the diagnosticsLog1 design block, click on the **Message**.

5. In the **Message** field, enter your name.

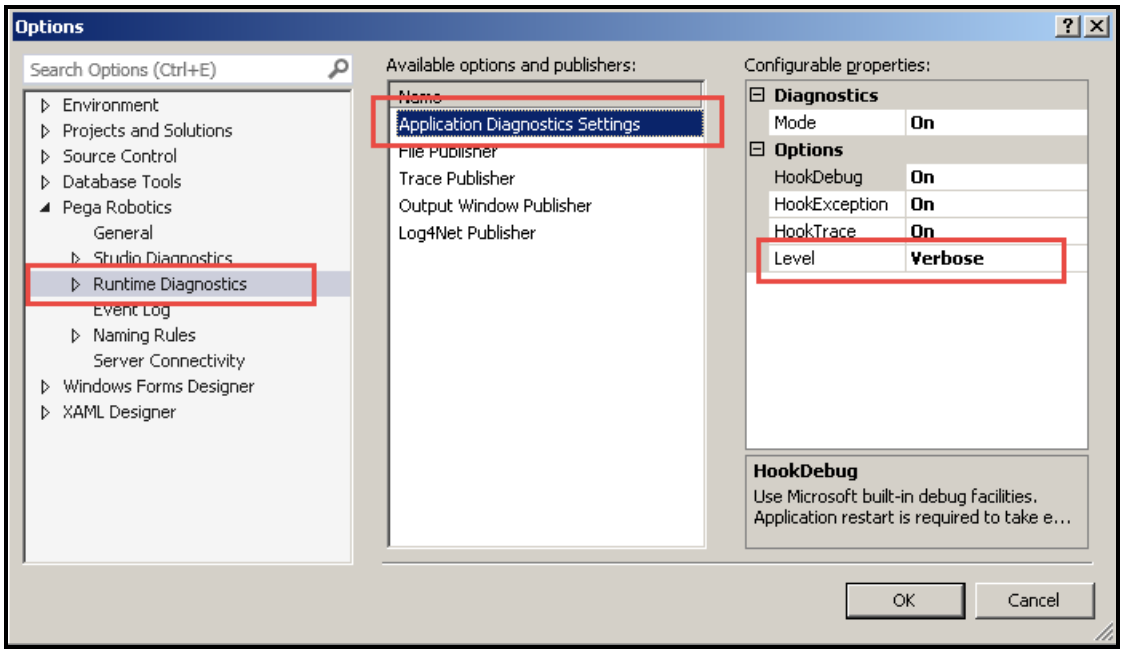


6. Connect the automation links to include the diagnosticsLog1 component.

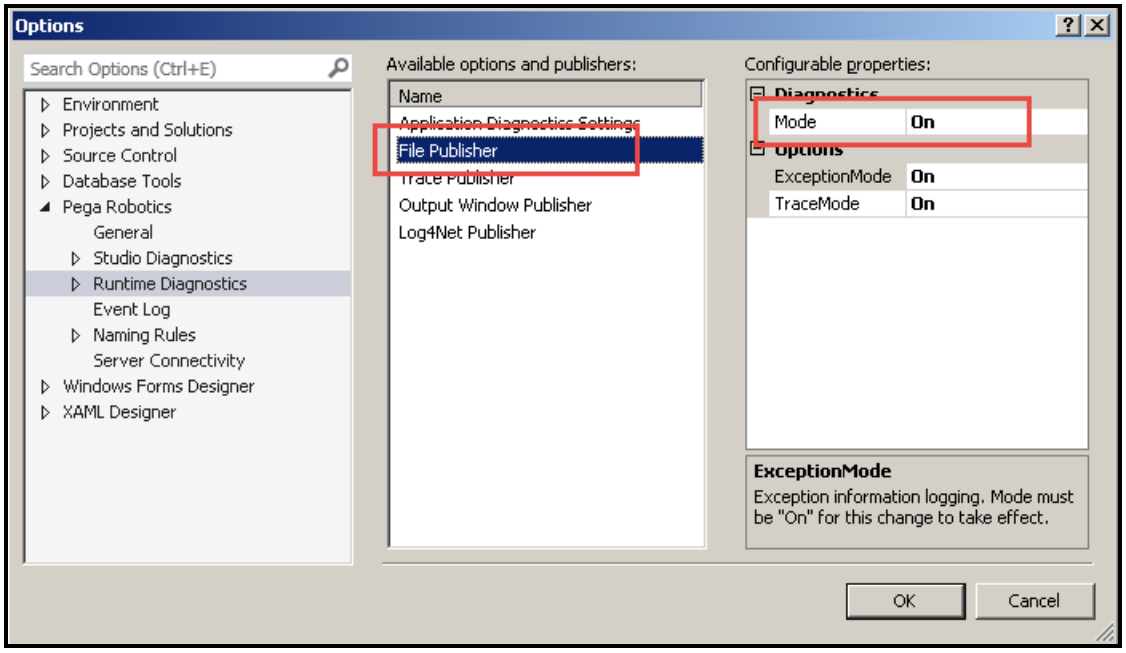


7. From the menu, select **File > Save All** to save the automation changes.
8. From the menu, select **Tools > Options**. The Options window displays.
9. Expand the Pega Robotic section.
10. Click **Runtime Diagnostics**.
11. In the center frame, select **Application Diagnostics Settings**.

12. In the Configurable Properties frame, select **Verbose** on the Level property.



13. In the center frame, select **File Publisher** to highlight it and the Configurable Properties frame updates.
14. In the Configurable Properties frame, select **On** for the Mode property.

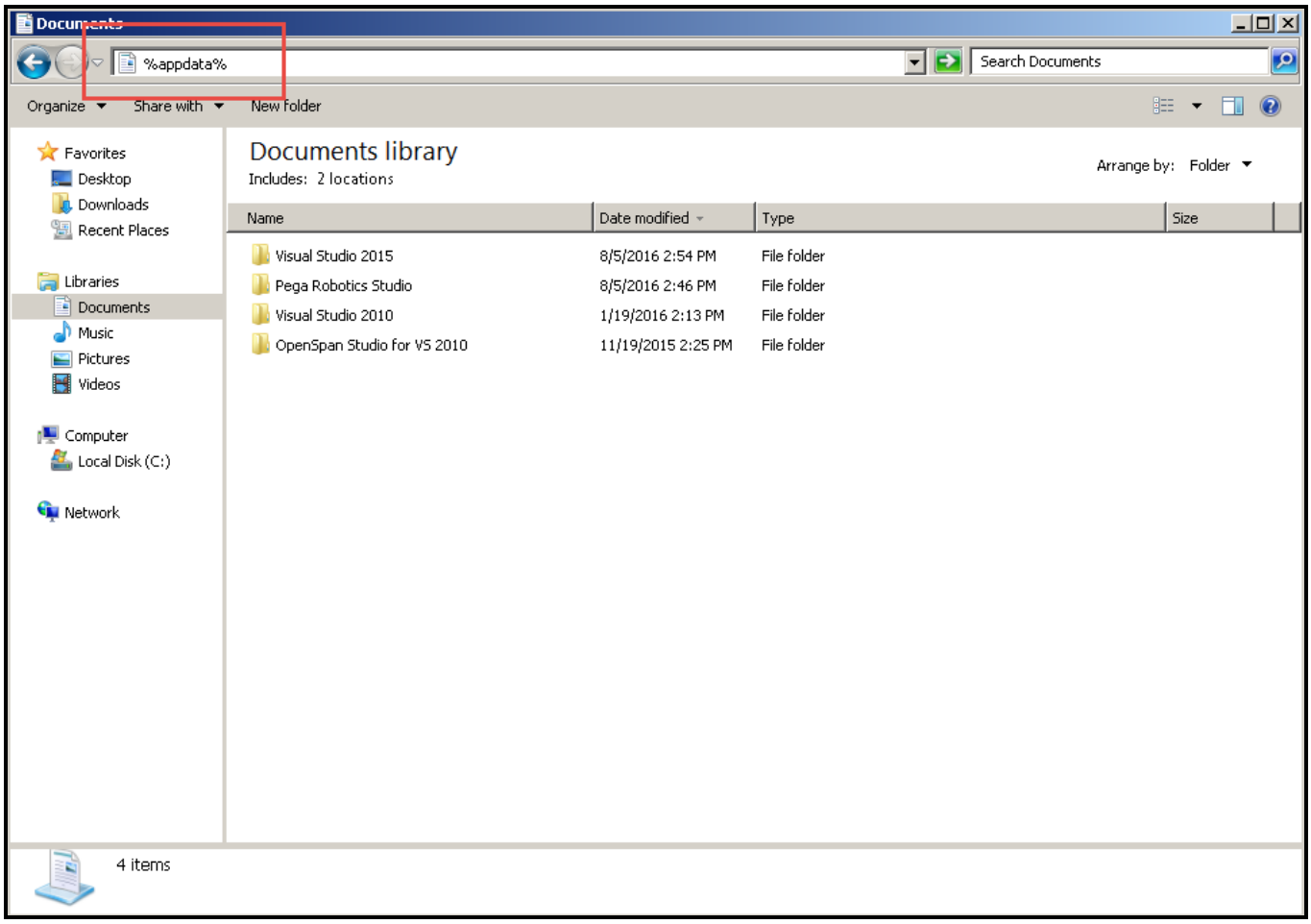


15. Click **OK** to save the work. The changes save, and the Option window closes.

Validate the log entry

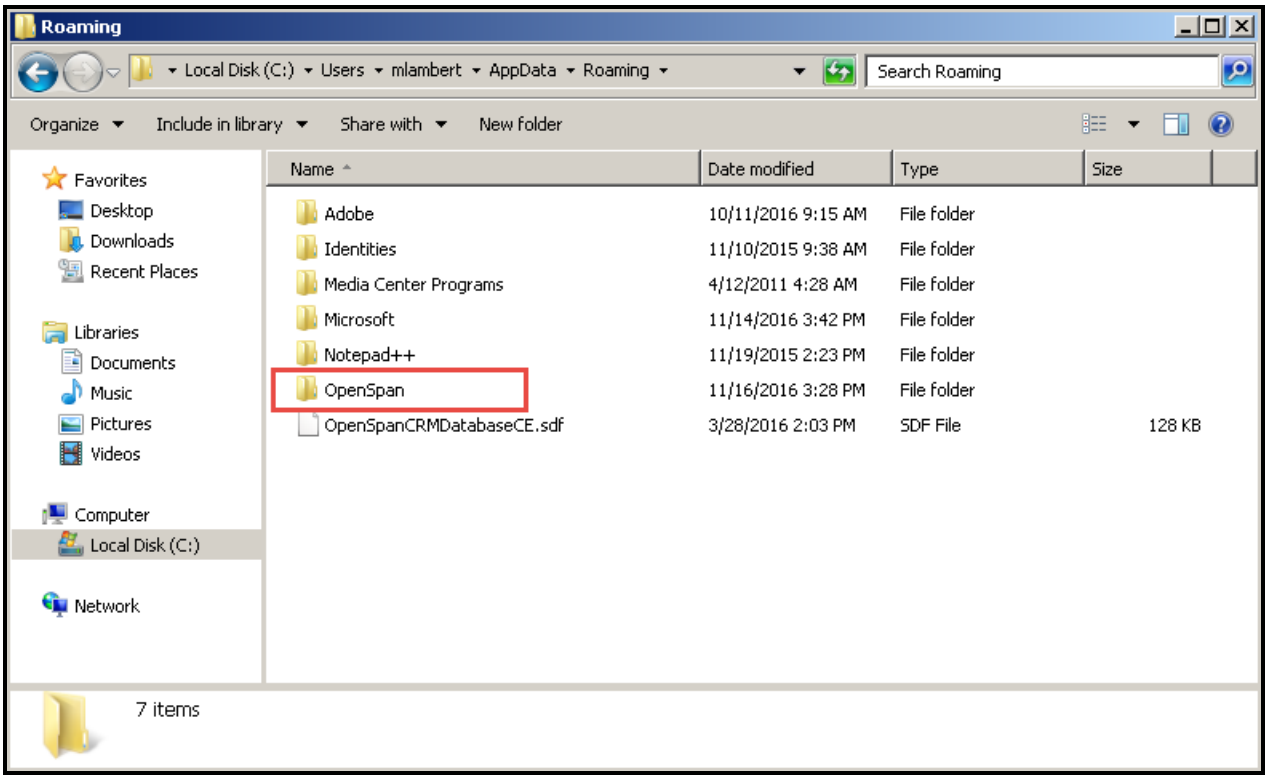
1. Debug the solution. The Customer Login window displays and the application performs an automatic login.
2. Open a Windows Explorer window.

3. In the Address bar, enter **%appdata%**.



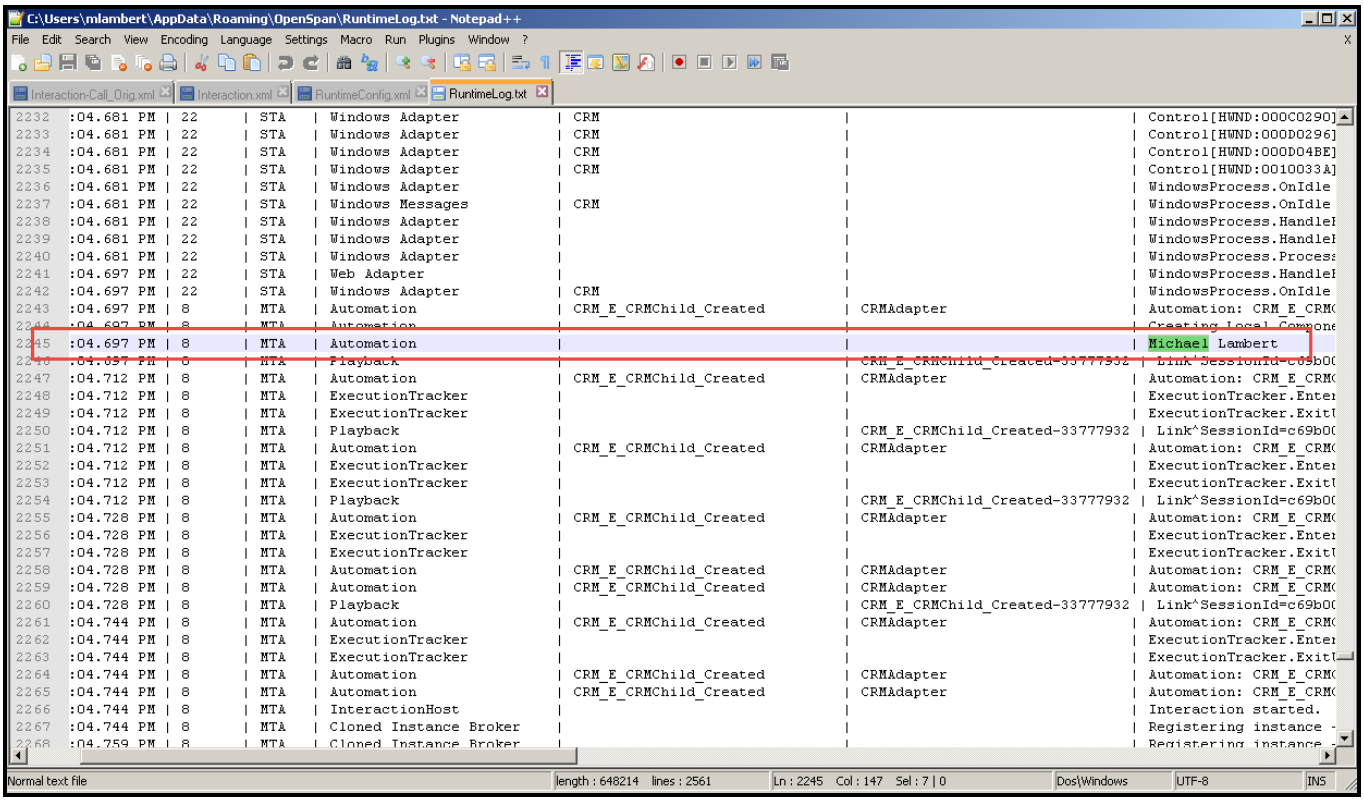
4. Press **Enter** on the keyboard. The Explorer windows opens to your Roaming application data directory.

5. Double click the **OpenSpan** folder.



6. In the OpenSpan folder, locate the RuntimeLog.txt file.
7. Double click the **RuntimeLog.txt** file to open it. The file opens in Notepad or another application.

8. If available in the open application, search for your name. Your name displays in the log.



9. Close the log file application.
10. Stop the debugging process. All application windows close and Studio returns to development mode.

WEB INTEGRATION

This lesson group includes the following lessons:

- Working with web adapters
- Working with match rules
- Navigating through a web application

- Working with Interaction Framework
- Integration with Pega 7 Platform

Working with web adapters

Exercise: Adding a web adapter

Scenario

In the first part of the business case, the solution retrieves the account information from the CRM application. In the second part, the users search for the nearest store based on the customer's zip code using the web application, ACME Search System. A new application requires a new project and adapter. The web application starts only when users click a button on the user interface to start the nearest store search.

Your assignment

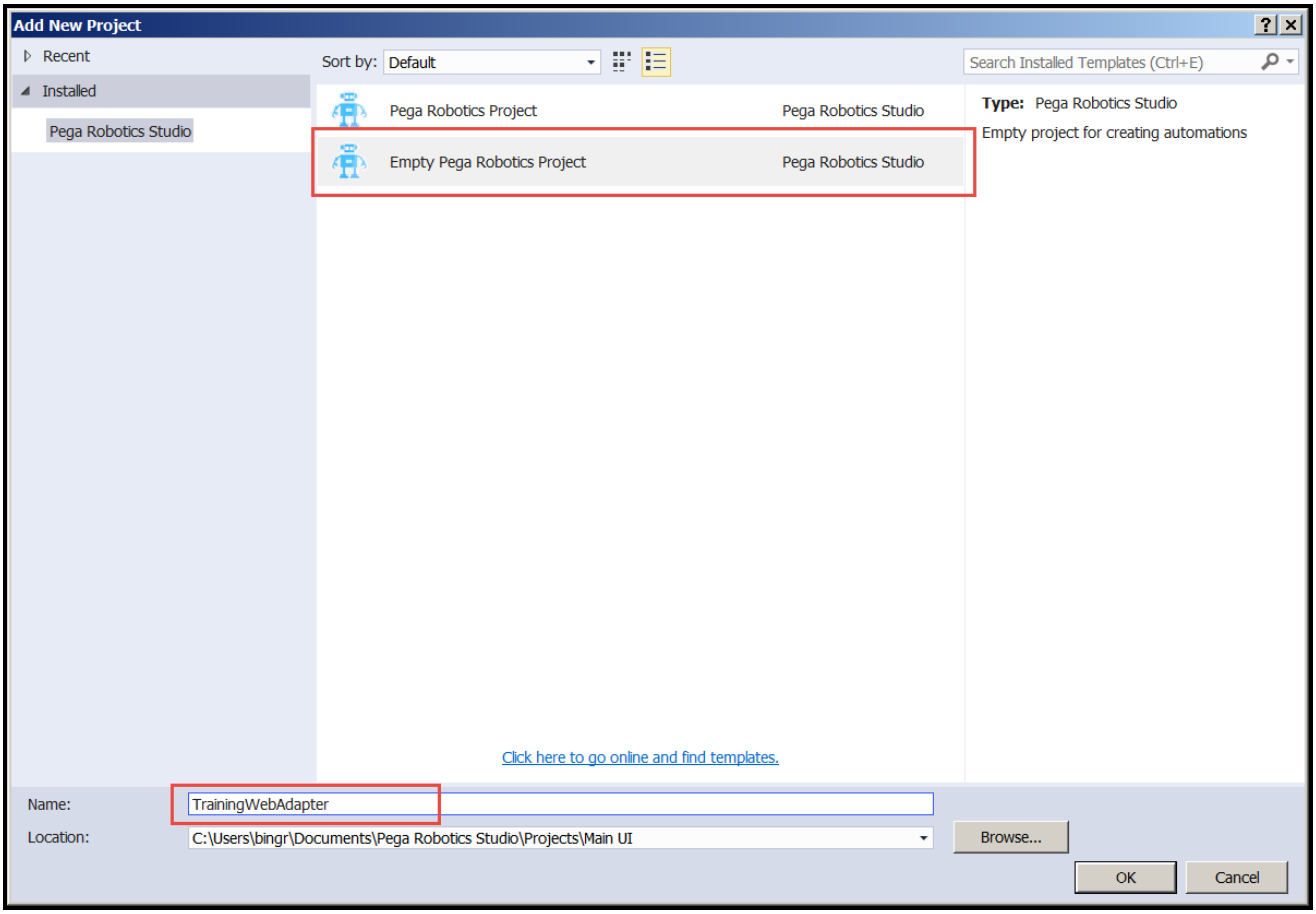
In this part of the solution, complete the following tasks:

- Add a new project to the solution, named TrainingWebAdapter.
- Add and configure a web adapter, named ACMESearchSystem, in the new project and configure the adapter properties using the following table.

Property	Value
StartPage	http://training.openspan.com
IgnoreMainBrowser	False
StartOnProjectStart	False

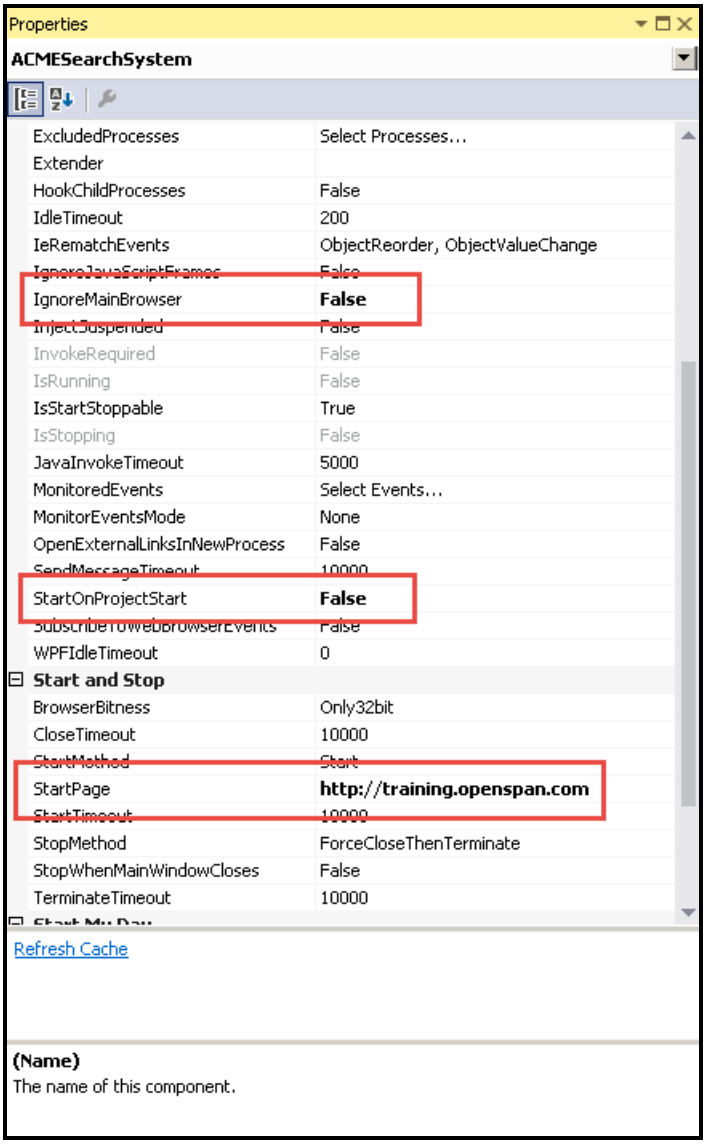
Detailed steps

1. Add a new project to the TrainingCertification solution.
   a. The project is an Empty Pega Robotic Project.
   b. Name the project **TrainingWebAdapter**.



2. In the Solution Explorer, right-click TrainingWebAdapter and select **New Web Application...**. The Add New Item window is displayed.
3. In the **Name** field, enter **ACMESearchSystem**.
4. In the Properties window for ACMESearchSystem, change the properties in the following table.

Property	Value
StartPage	http://training.openspan.com
IgnoreMainBrowser	False
StartOnProjectStart	False



5. Select **File > Save All** to save the changes.

Exercise: Interrogating a web application

Scenario

In the business case, it states that users access the ACME Search System application with an automatic login to search for the nearest store by the customer's zip code. To complete this process, you must interrogate the web application for the needed controls and rename them for use in the solution. The ACME Search System login requires four characters for both the username and password fields. A hidden control on the sign-in page requires interrogation.

Your assignment

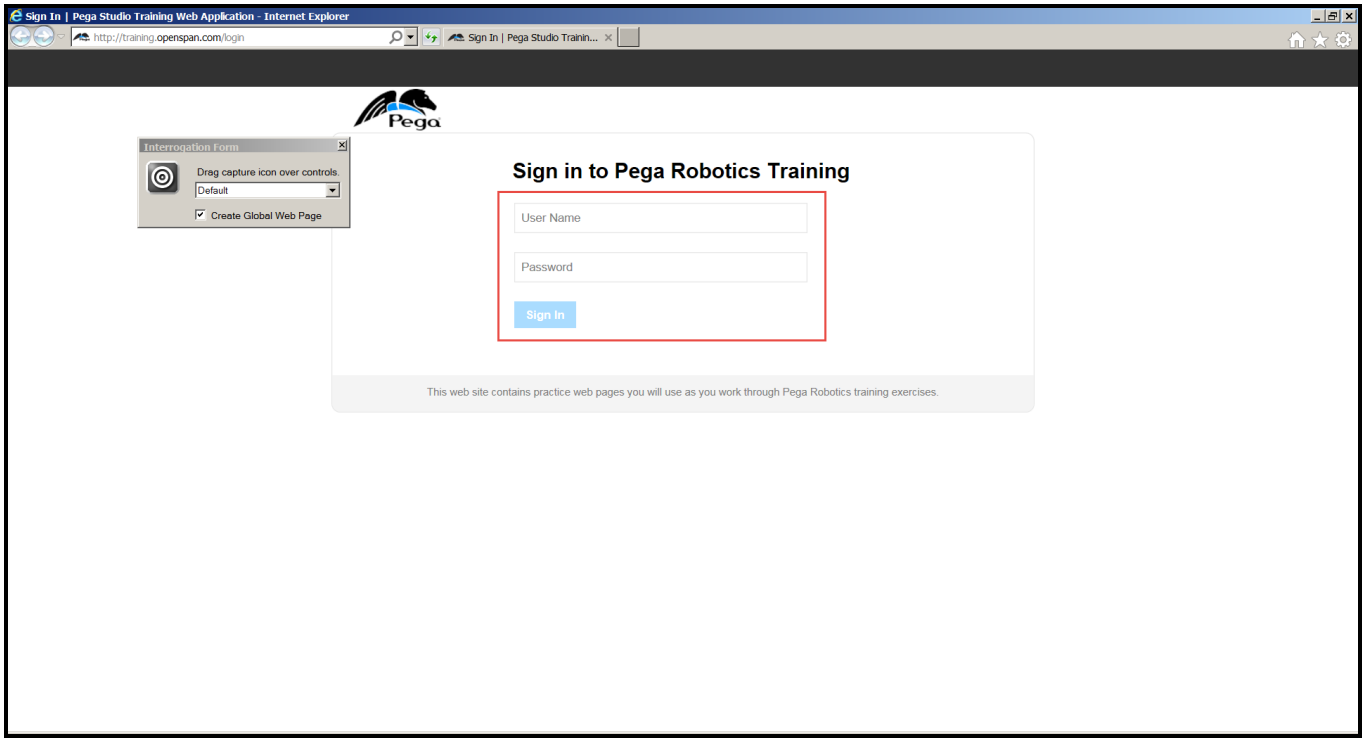
In this part of the solution, complete the following:

- Interrogate the web application
- Interrogate using Select Element
- Interrogate using Web Controls

Detailed steps

Interrogate the web application

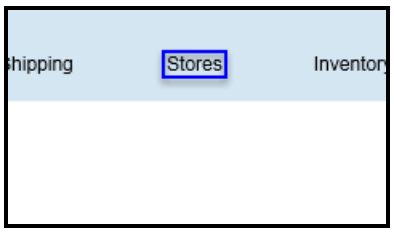
1. Open the **ACMESearchSystem.os** project item in a design window.
2. Click **Start Interrogation**. The Interrogation Form and browser open.
3. With the Interrogation Form dialog option set as Default and Create Global Web Page checked, click and drag the **Target** icon over the User Name, Password, and SignIn targets.



4. Using the Object Hierarchy and Properties window, highlight each object and rename the Design Name as follows:

Object Name	Design Name
Sign_In_Pega_Training_Web_Appl	ACMEwpSignIn
Please_enter_a_user_name_of_at_least	ACMEfrmCredentials
Login_button	ACMEbtnSignin
textBox1 (user_password)	ACMEtxtPassword
User_name	ACMEtxtUserName

5. In the **User Name** and **Password** fields, enter **1234**. SignIn becomes enabled.
6. Click **SignIn.** The web site navigates to the Home Page.
7. Click and drag the target icon over the **Stores** link. Make sure the smallest highlight box displays for the correct target.



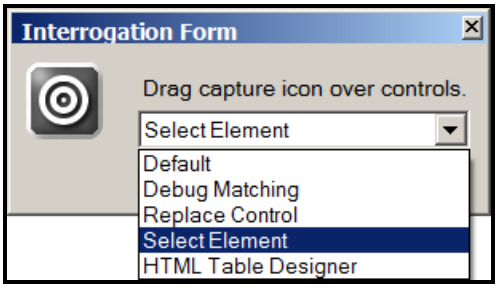
8. Click the **Stores** link. The website navigates to the Stores search page.
9. Interrogate the **Zip Code** field and **Find Store** button.
10. Using the Object Hierarchy and Properties window, highlight each object and rename the Design Name as follows:

Object Name	Design Name
Home_Pega_Training_Web_Applica	ACMEwpHome
Stores	ACMElnkStores
Stores_Pega_Studio_Training_Web_Ap	ACMEwpStores
Search_by_ZIP_CodeSubtrationORSubt	ACMEfrmSearch
Find_Store	ACMEbtnFindStore
txtZip	ACMEtxtZip

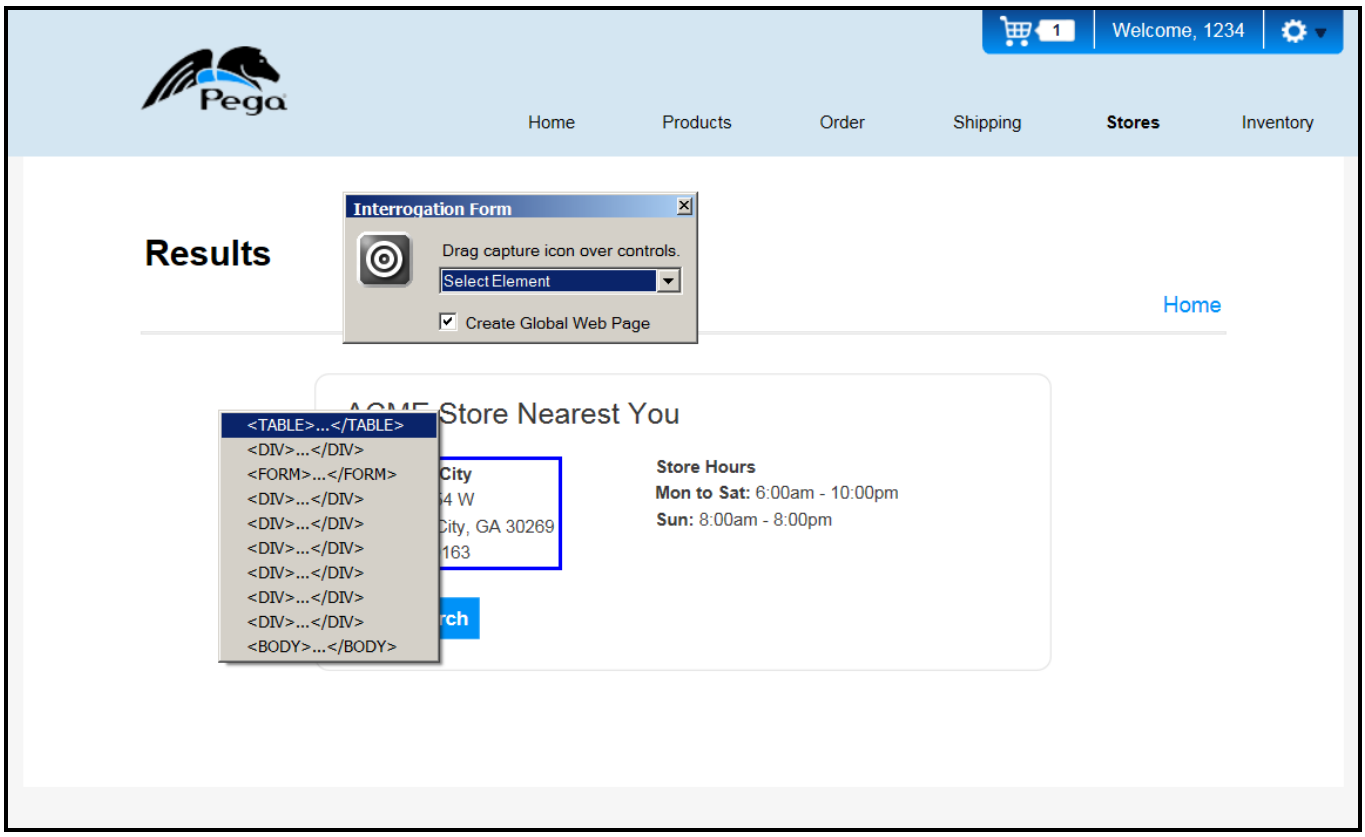
11. On the Search page, enter **30022** in the Zip Code.
12. Click **Find Store** button. The Results page displays.

Interrogate using Select Element

1. In the Interrogation Form, select **Select Element** from the combo box.



2. Interrogate the store address.
3. Release the mouse button. A context menu displays. Depending on which object you released the mouse button, the context menu may appear differently.

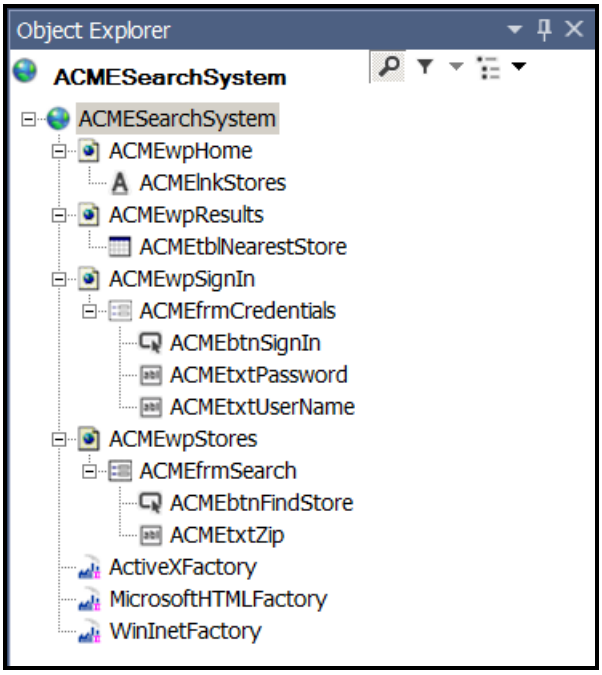


4. On the context menu, select **<TABLE>...</TABLE>**. This object collects all table rows as one unit to use for the address instead of each individual address item.
5. Using the Object Hierarchy and Properties window, highlight each object and rename the Design Name as follows:

Object Name	Design Name
Store_Locator_Pega_Studio_Training	ACMEwpResults
Peachtree_City2715_HWY_54_WPeachtre	ACMEtblNearestStore

6. Stop the interrogation process.
7. Select **File > Save All** to save the changes.

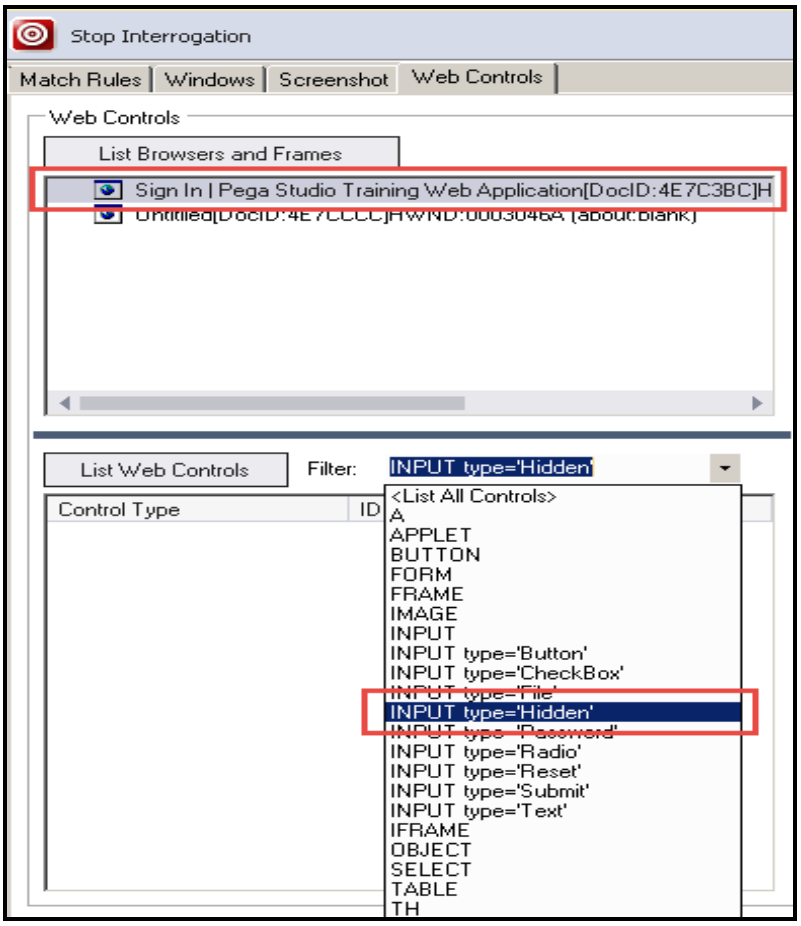
When completed, the Object Hierarchy should look as the following image:



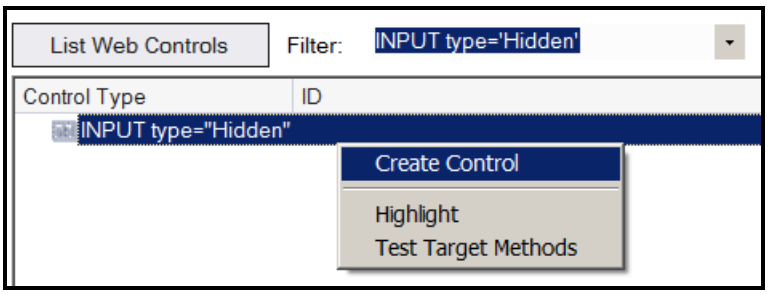
Interrogate using Web Controls

1. Click **Start Interrogation**. The Interrogation Form and browser open.
2. In the design window, click the **Web Controls** tab.
3. Click the **List Browsers and Frames** button. The open browser displays in the list.
4. Select the **Sign In** browser in the list.

5. In the Filter list, select **INPUT type = 'Hidden'**.



6. Click the **List Web Controls** button. A hidden text box control appears.
7. Right-click the INPUT type='Hidden' control and select **Create Control**. The control displays in the Object Hierarchy as **pegaystem_version**.



8. In the Properties window, rename the new control to **ACMElblVersion**.
9. Stop the interrogation.
10. On the menu, click **File > Save All** to save your edits.

Working with match rules

Exercise: Using Attribute Value match rule

Scenario

A control, although matching, needs changing to use a specific match rule. An attribute called name contains a value to use with the match rule.

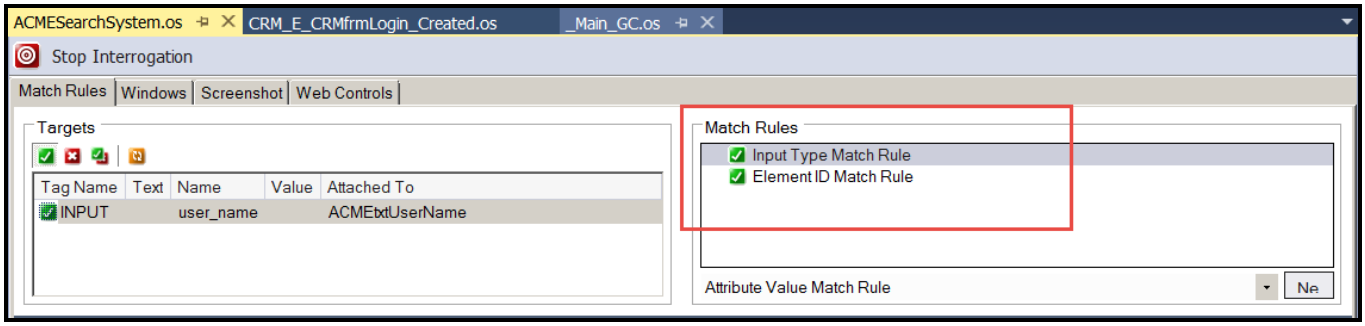
Your assignment

For this exercise, use the Attribute Value match rule to modify the match rule set on the ACMEtxtPassword control.

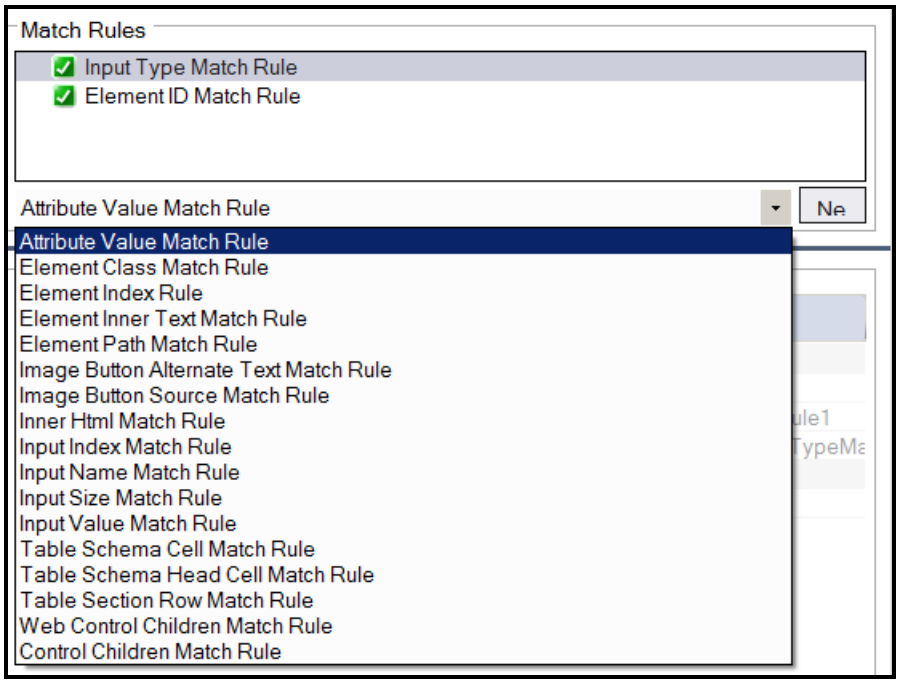
Detailed steps

Follow these steps to add the Attribute Value match rule.

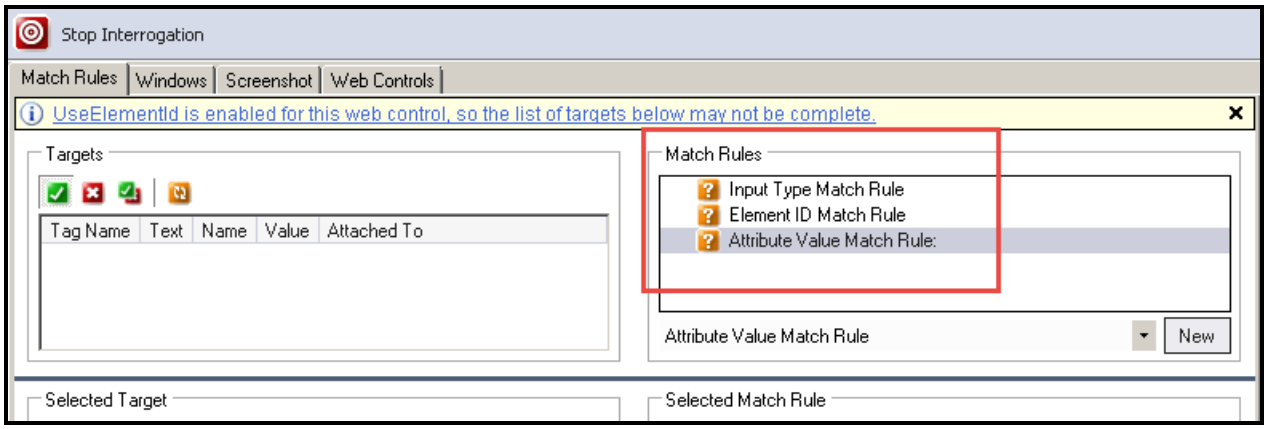
1. In the Solution Explorer, double click **ACMESearchSystem.os** to open it in a designer window.
2. On the designer tab, click **Start Interrogation**. The web application launches and the Login page appears.
3. In Object Explorer, select **ACMEtxtUserName**. As seen in the Match Rules frame on the designer window, the control uses the Input Type and Element ID match rules.



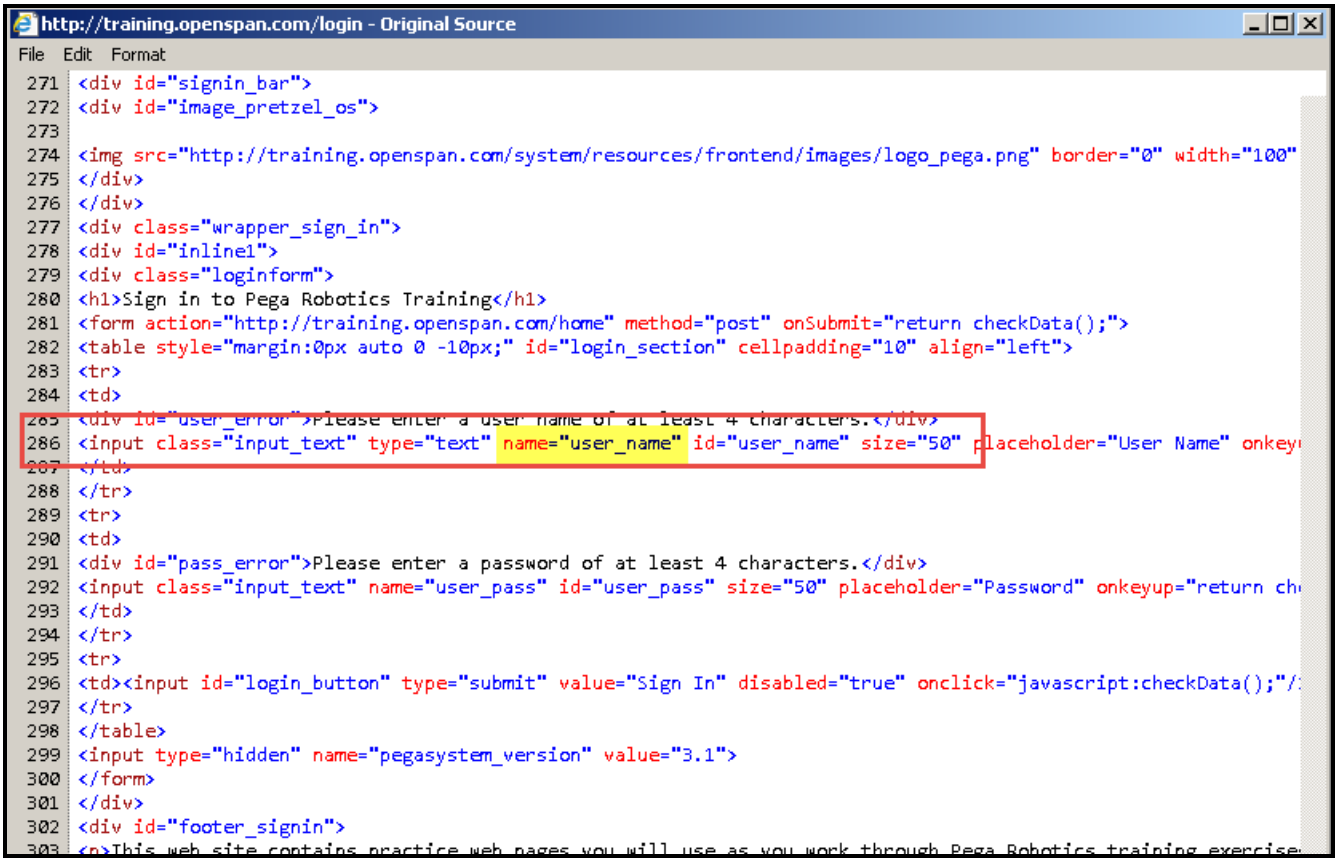
4. From the combo box below the Match Rule frame, select **Attribute Value Match Rule**.



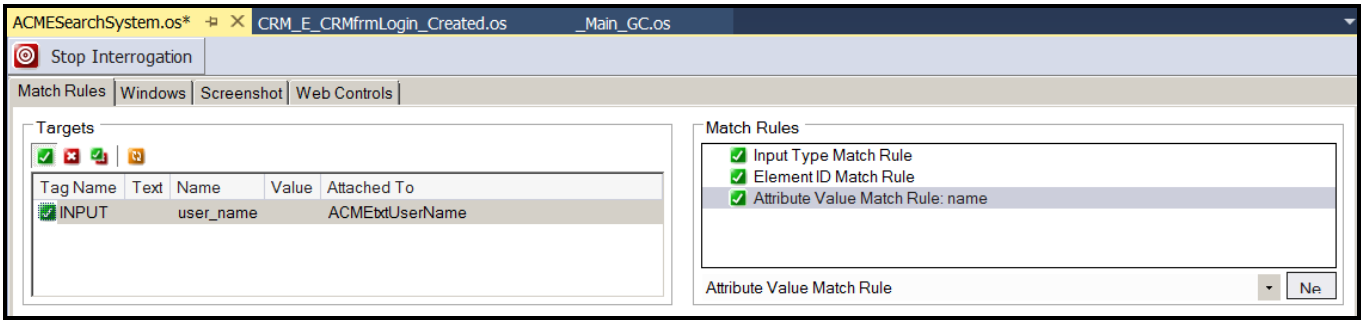
5. Click **New**. The match rule displays in the frame and all match rule green check marks are now orange question marks.



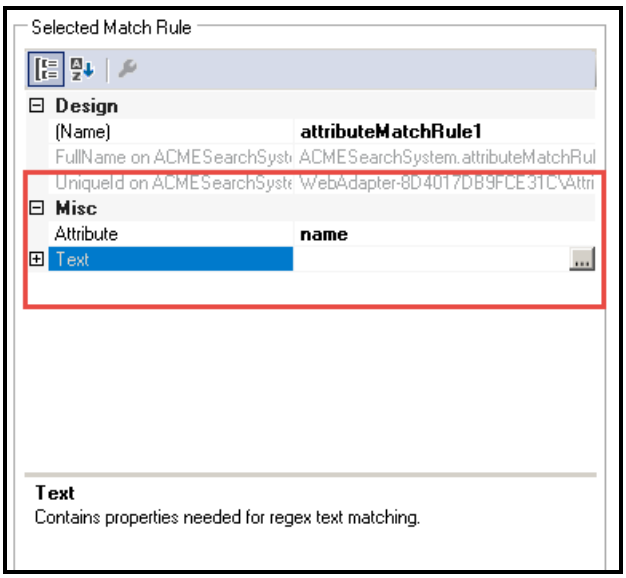
6. Bring the browser back to the active window using the Taskbar. In the browser, right-click on the web page and select **View Source**. A separate window displays with the color-coded HTML.
7. Scroll through the code until line 286. (It may be a different line number depending on your browser settings. An input tag displays indicating the user_name. This is the user name field on the web page.



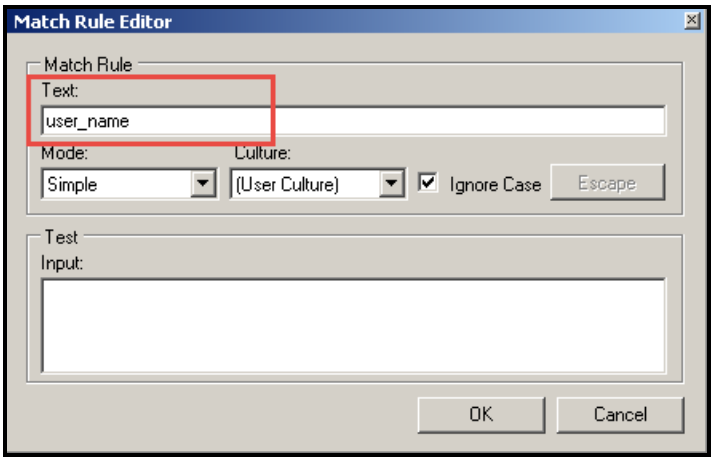
8. In Object Explorer, select the **ACMEtxtUserName** web control to highlight again.
9. From the Match Rule drop-down box, select the **Attribute Value Match** Rule.



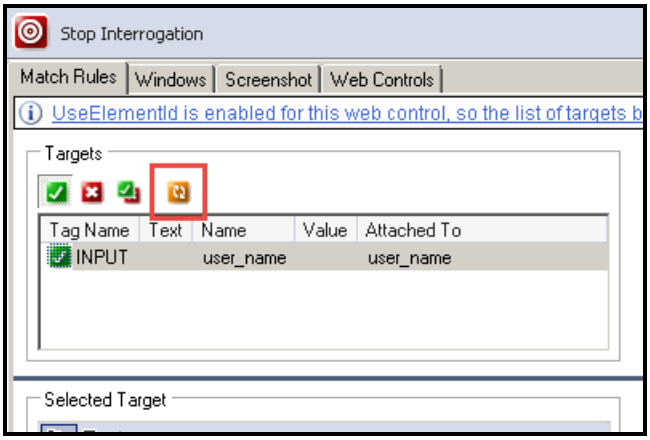
10. In the Design window, in the Selected Match Rule frame , click the **Attribute** property and enter **name**. This is the attribute in the HTML code
11. In the Design window, in the Selected Match Rule frame, click the **Text** property. An ellipses button displays.



12. Click the **ellipses** button. The Match Rule Editor window displays.
13. In the Match Rule editor window, in the Text field, enter **user_name**. This is the value of the name attribute.



14. Click **OK** to close the window.
15. In the Design window, click the **Refresh matching** button. The matching green check mark appears.



16. In Design window, in the Match Rules from, select **Element ID Match Rule** to highlight it.
17. Right-click on the **Element ID Match** Rule and select **Delete**.
18. Stop the interrogation.
19. Select **File > Save All** to save your edits.

Exercise: Using Element Index match rule

Scenario

You interrogate a text object on the home page of the ACMESearchSystem for use in the solution. You receive a message regarding Studio's inability to match the target.

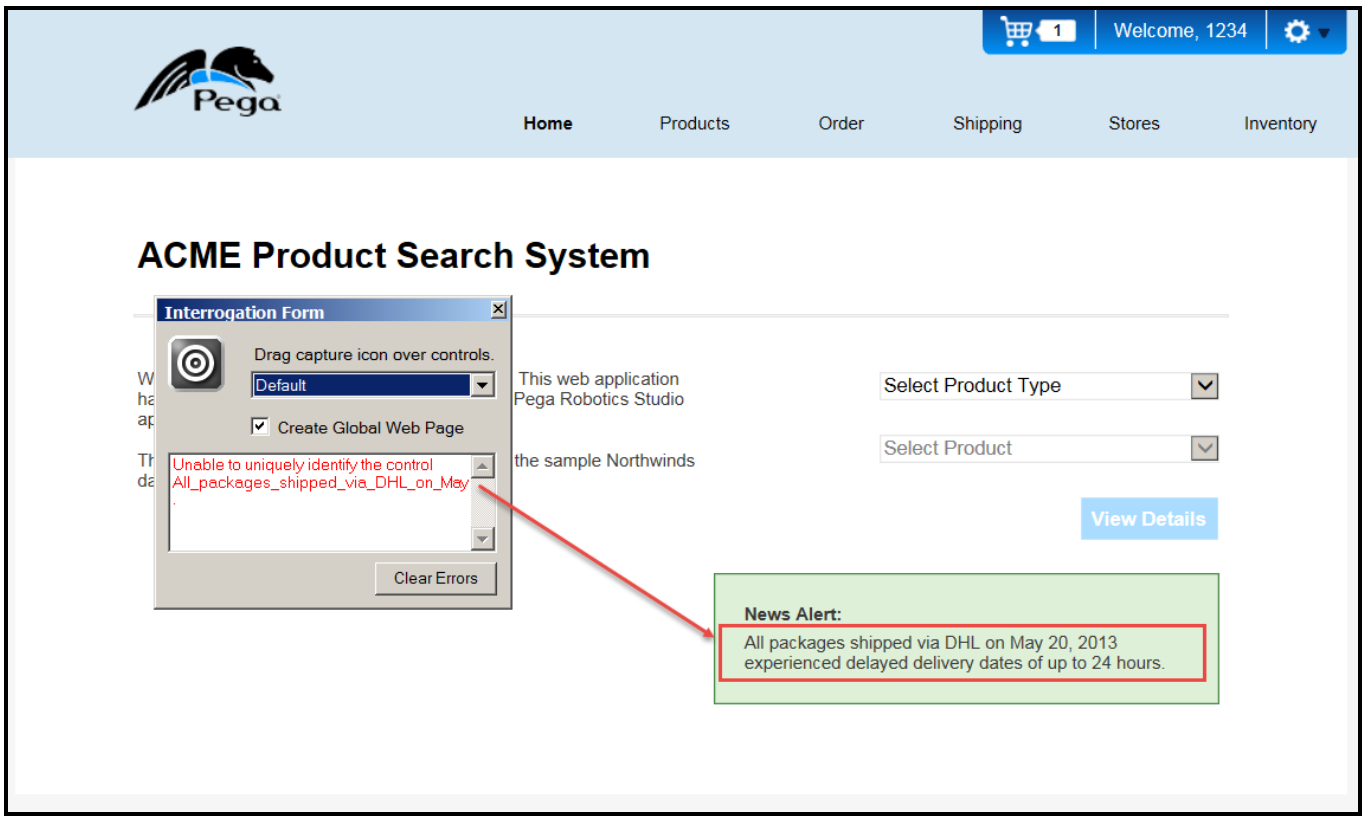
Your assignment

In this exercise, interrogate the text object, rename the text object, and correct the ambiguous matching using the Element Index match rule.

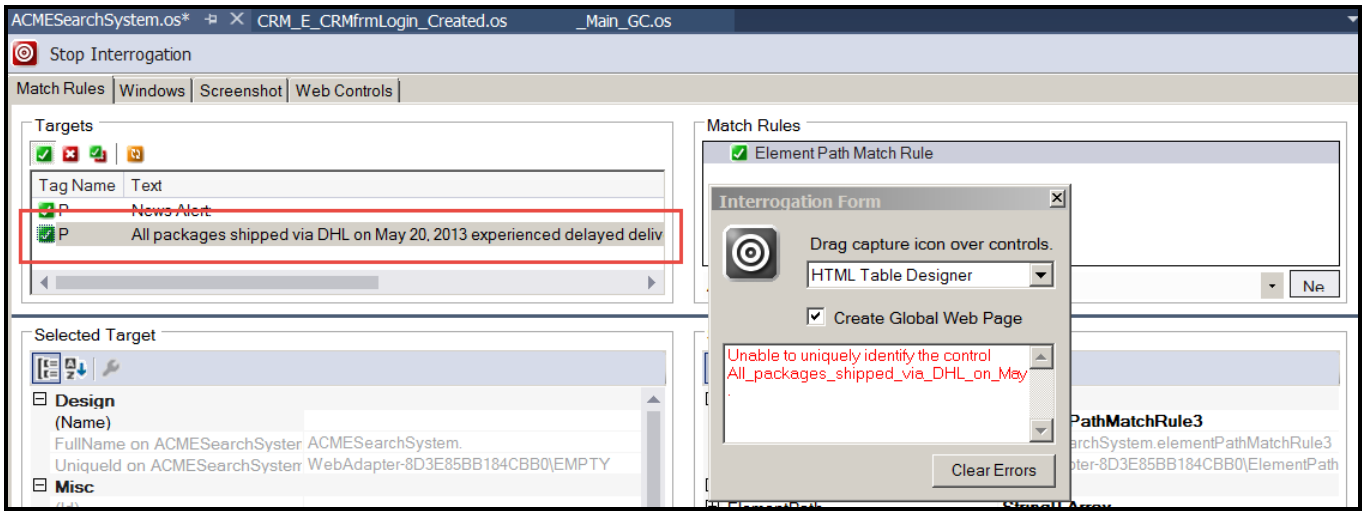
Detailed steps

Follow these steps to add the Element Index match rule.

1. In the Solution Explorer, double click **ACMESearchSystem.os** to open it in a designer window.
2. On the designer tab, click **Start Interrogation**. The web application launches and the Login page appears.
3. On the Interrogation Form, use Default and Create Global Web Page.
4. Enter **1234** in both the user name and password fields. Click **Sign In**. The home page loads.
5. Click and drag the interrogation target icon and interrogate the text: "All packages shipped via DHL on May 20, 2013 experienced delayed delivery dates of up to 24 hours." The Interrogation Form displays a message.



6. Inspect the Target frame on the Match Rules tab and note that multiple Paragraph type elements match.
7. In the properties window, rename the control to **ACMENewsAlert**.
8. In the Target frame on the Match Rules tab, highlight the Tag Name (P) and Text (All packages shipped via DHL….) line.



9. In the Match Rules frame, click the drop-down arrow to display a list of matching rules.
10. Select the **Element Index Rule** and click **New** to add to the Match Rules frame.
11. In the Match Rules frame, right-click the **Element Path Match Rule** and select **Delete.**
12. Stop the interrogation.

13. From the menu, select **File > Save All** to save your edits.

Navigating through a web application

Exercise: Creating a Windows form for testing

Scenario

The solution design requires a button click on a user interface to perform a search in the ACME Search System and return the nearest store's address based on the customer's zip code. The solution does not have the user interface developed. So, an interim solution is to develop a testing Windows form that allows the developer to complete and test the automations in the TrainingWebAdapter project.

Your assignment

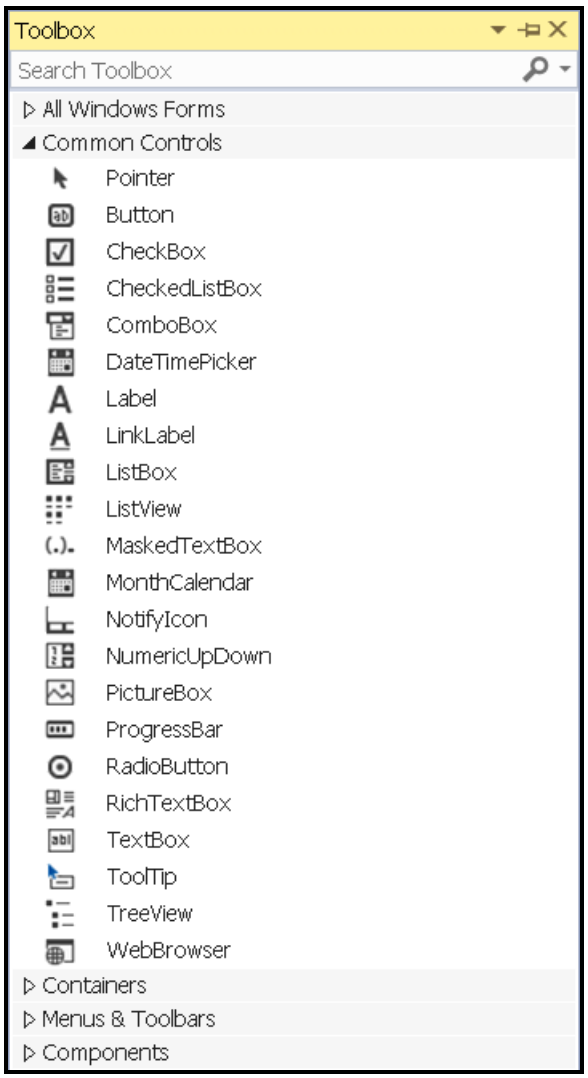
Create a Windows form to test the automations in the TrainingWebAdapter. The form contains a label, a text box, and a button. Use the following table to configure the form and controls.

Object Name	Properties to update	Property Value
label1	Design Name	lblZipCode
	Text	Zip Code
textbox1	Design Name	txtZipCode
button1	Design Name	btnTest
	Text	Test

Detailed steps

Follow these steps to add the testing Windows form.

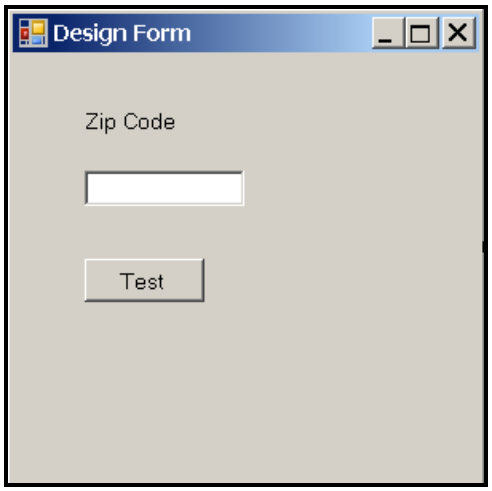
1. From the Solution Explorer, right-click the **TrainingWebAdapter** project and select **Add > New Windows Form...**. The Add New Item window is displayed with the Windows Form item highlighted.
2. In the **Name** field, enter **frmTest**.
3. Click **Add**. The frmTest.os window is displayed in the Studio Designer window.
4. In Object Explorer, highlight **frmTest** to display the full list of available properties in the Properties window.
5. In the Properties window, set the TopMost property to **True**.
6. Select **File > Save**.
7. The controls you need for the Customer Information window are in the Common Controls section of the Toolbox. Expand the **Common Controls** section of the Toolbox and locate the Label, Textbox, and Button controls.



8. Drag and drop a label, a text box, and a button onto the **frmTest** window.
9. Select **File > Save**.
10. Using the Properties window, change each control's Properties and Design (Name) property as displayed in the following table.

Object Name	Properties to update	Property Value
label1	Design Name	lblZipCode
	Text	Zip Code
textbox1	Design Name	txtZipCode
button1	Design Name	btnTest
	Text	Test

11. Select **File > Save** to save the Windows form.



Exercise: Updating project structure

Scenario

The business requirements state that the user should begin the search process and all other processes from the ACME Search System home page. The business case requires the solution to check the running of the web adapter, sign the user in automatically to the ACME Search System, and ensure the user is on the home page.

Your assignment

At this point in the solution, complete the following tasks:

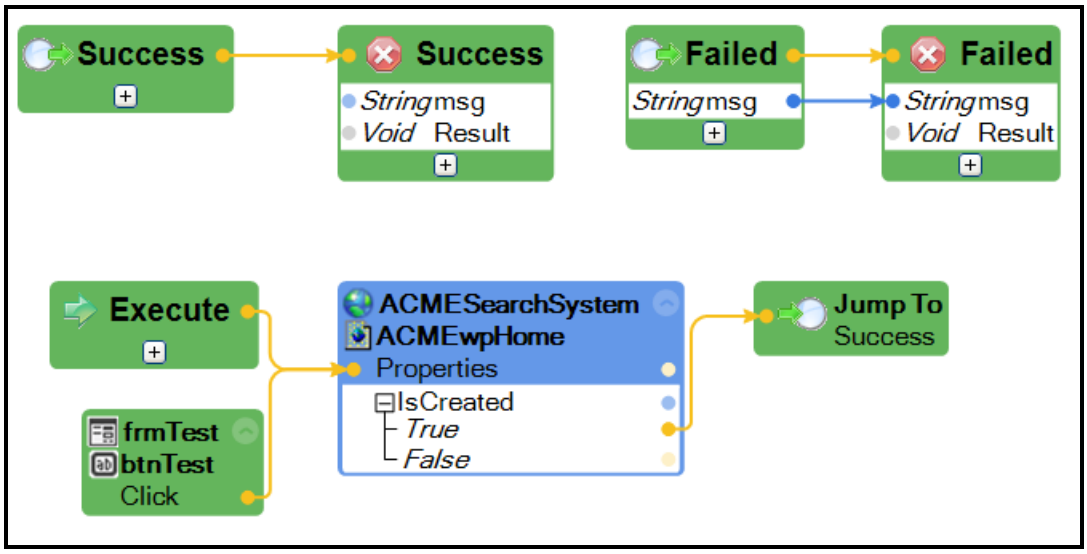
- Create a Procedures folder in TrainingWebAdapter to store the procedure automations.
- Create an automation, ACME_P_NavToHome, to check the location of the user in the web application, navigate the user to the home page, confirm the adapter is running, and call a sign-in automation.
- Create a procedure automation, ACME_P_SignIn, to sign in to the web application.

Detailed steps

1. In the Solution Explorer, add a folder to the TrainingWebAdapter and name the folder **Procedures**.
2. In the new Procedures folder, add a new automation.
3. Name the automation **ACME_P_NavToHome**. This automation validates the web application and navigates users to the home page to begin the search in the same spot each time. It also calls another automation to sign in to the web application.
4. Add the design blocks shown in the following table to the automation. Remember to use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
Right-click in the automation	Add Entry Point	
frmTest	btnTest.Click	This is a temporary design block to test the automations. It will be removed.
ACMESearchSystem	ACMEwpHome.IsCreated property	Expand the result. If the home page exists, the user has logged in to the web application. The automation exists if the result is True.
Right-click in the automation	Add two labels	• Rename one label **Success** and one **Failed**. • On the Failed label, click **+** to add one parameter. • Change param1 to **String** and rename it **msg**.
Right-click in the automation	Add two exit points	• Rename one **Success** and one **Failed**. • On the Failed Exit point, click **+** to add one parameter. • Change param1 to **String** and rename it **msg**.
Right-click in the automation	Select **Jump To > Success**.	

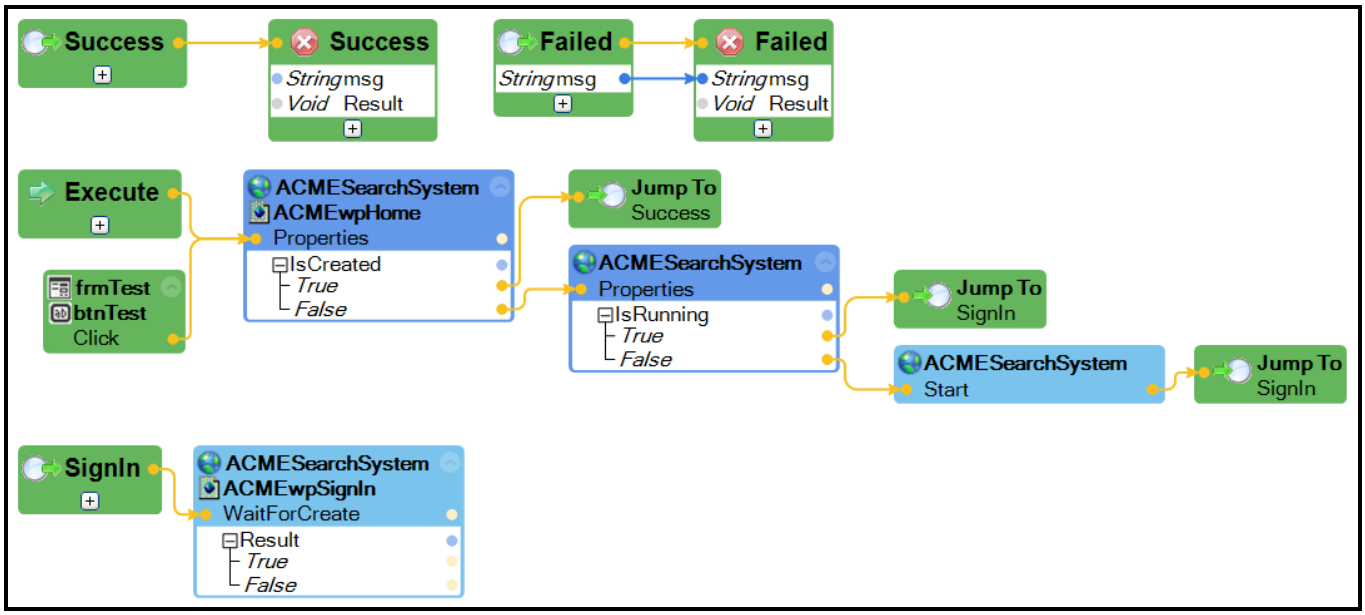
5. Connect the automation links as shown in the following image.



6. Save the automation.
7. Add the design blocks shown in the following table to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
ACMESearchSystem	ACMESearchSystem.IsRunning property	If the home page does not exist, check to see if the adapter is running. Click to expand the result.
ACMESearchSystem	ACMESearchSystem.Start method	This starts the adapter and launches the StartPage property.
ACMESearchSystem	ACMEwpSignIn.WaitForCreate method	If the automation starts the adapter, wait for the sign-in page to create before signing in.
Right-click in automation	Add label	Rename Label1 to **SignIn**.
Right-click in automation	Select **Jump To > SignIn** twice.	Copy and paste **Jump To Sign In**.

8. Connect the automation as shown in the following image.



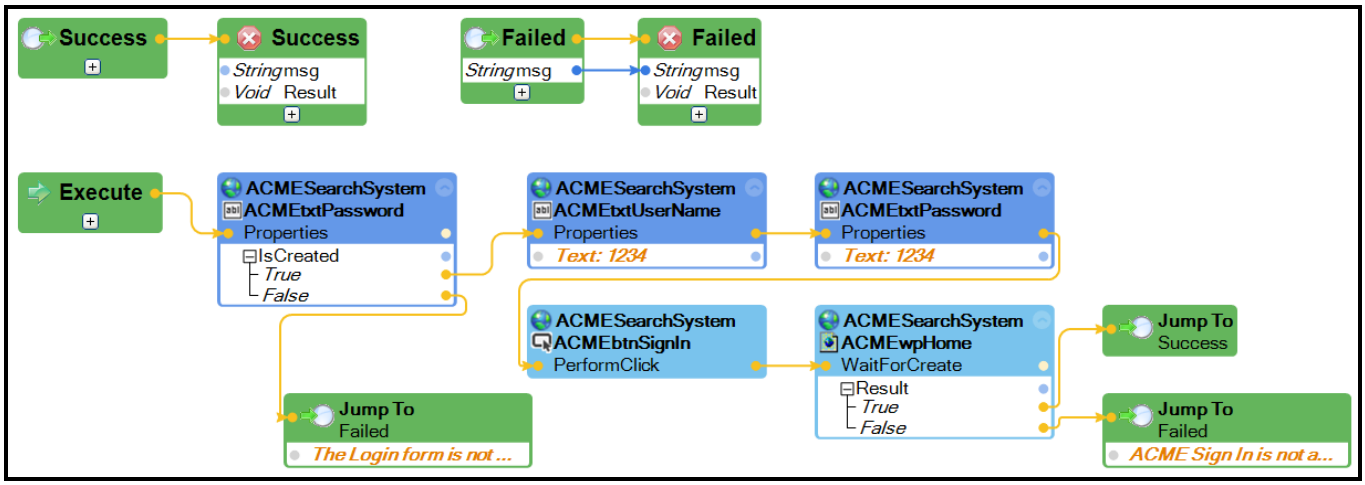
9. Save the automation.
10. In the Procedures folder, add a new automation.
11. Name the automation **ACME_P_SignIn**.
12. Add the design blocks shown in the following table to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
Right-click in the automation	Add Entry Point	
ACMESearchSystem	ACMEtxtPassword.IsCreated property	Expand the result. If the password field exists, continue with the sign-in process.
Right-click in the automation	Add 2 Labels	• Rename one label **Success** and one **Failed**. • On the Failed label, click **+** to add one parameter. • Change param1 to **String** and rename it to **msg**.
Right-click in the automation	Add two exit points	• Rename one **Success** and one **Failed**. • On the Failed exit point, click **+** to add one parameter. • Change param1 to **String** and rename to **msg**.

13. Add the design blocks shown in the following table to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
ACMESearchSystem	ACMEtxtUserName.Text property	Click **Text** and enter **1234**. Use four characters for the user name and password fields.
ACMESearchSystem	ACMEtxtPassword.Text property	Click **Text** and enter **1234**.
ACMESearchSystem	ACMEwpSignIn.PerformClick method	After entering the user name and password, click **Sign In** to access the application.
ACMESearchSystem	ACMEwpHome.WaitForCreate method	When the automation logs in the user, wait for the home page to create before exiting the function to find the nearest store.
Right-click in the automation	Select **Jump To > Success**.	
Right-click in the automation	Select **Jump To > Failed**.	Add one message to string parameter for each Jump To.Failed • **The Login form is not available.** • **ACME Sign in is not available.**

14. Connect the automation links as shown in the following image.

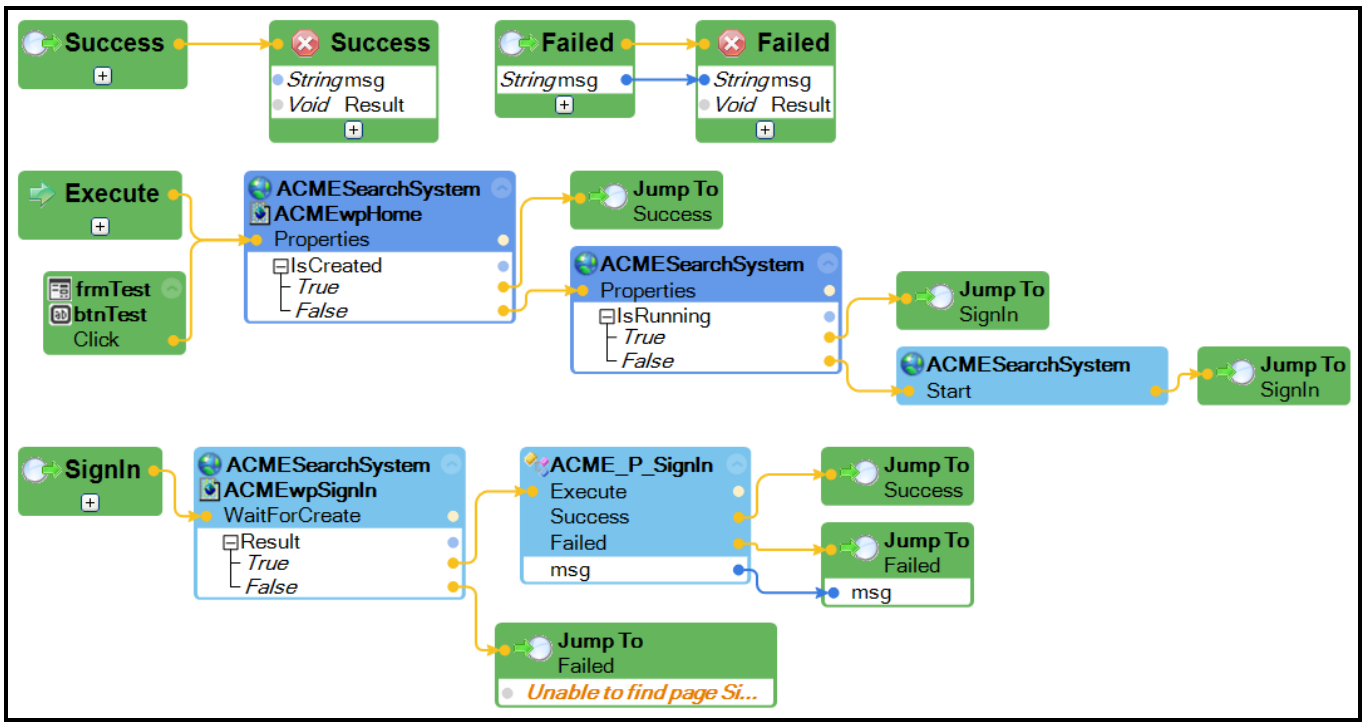


15. Save the automation.
16. Return to the ACME_P_NavToHome automation.
17. From the Object Explorer, add the **ACME_P_SignIn.Execute** method.

18. Add the design blocks shown in the following table to the automation. Remember to use the Configure Type icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
Right-click in the automation	Select **Jump To > Success**.	
Right-click in the automation	Select **Jump To > Failed** twice.	Add a message to the string parameter for one Jump To.Failed • **Unable to find page Sign In.**

19. Connect the automation links as shown in the following image.



20. Save the automation.

Verify your work

1. In the Solution Explorer, right-click **TrainingWebAdapter** and select **Set as Startup Project**.
2. Start the debugger.
3. On the test Windows form, click **Test**. Notice the process works until the Sign In page, where the button is not clicked.

Exercise: Incorporating the RaiseEvent method

Scenario

The ACME_P_SignIn automation did not successfully log in the user. Refactor the automation to have the user log in successfully to complete the search for the nearest store.

Your assignment

In this exercise, investigate the source code on the Sign In page to determine how to modify the ACME_P_SignIn automation. Edit the automation to account for the event needed to log in.

Detailed steps

Follow these steps to add a RaiseEvent method to an automation.

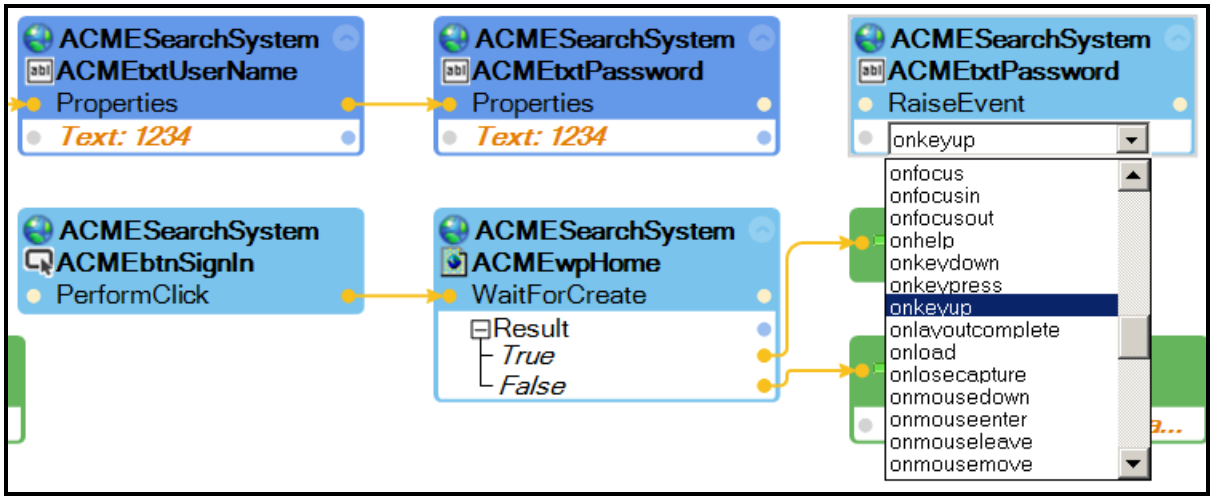
1. In the Solution Explorer, double click **ACMESearchSystem.os** to open it in a Designer window.
2. On the Designer tab, click **Start Interrogation**. The web application launches and the Login page appears.
3. Right-click on the Sign In page and select **View Source**. A viewer displays the page source code.
4. Search for checkLengthP() (roughly around line 32). The code calls a function to check the length of the password entry before enabling the Sign In button.

```
30 </script>
31 <script type="text/javascript">
32 function checkLengthP()
33 {
34     var login = document.getElementById("user_name").value;
35     var passw = document.getElementById("user_pass").value;
36
37     if(passw.length>=4 && passw!="Password")
38     {
39         if(login.length>=4 && login!="User Name")
40         {
41             document.getElementById("user_error").style.display = 'none';
42             document.getElementById("pass_error").style.display = 'none';
43             document.getElementById("login_button").disabled = false;
44             return true;
45         }else{
46             document.getElementById("user_error").style.display = 'block';
47             document.getElementById("login_button").disabled = true;
48             return true;
49         }
50     }else{
51         document.getElementById("login_button").disabled = true;
52         return false;
53     }
54 }
55 </script>
56 <script type="text/javascript">
```

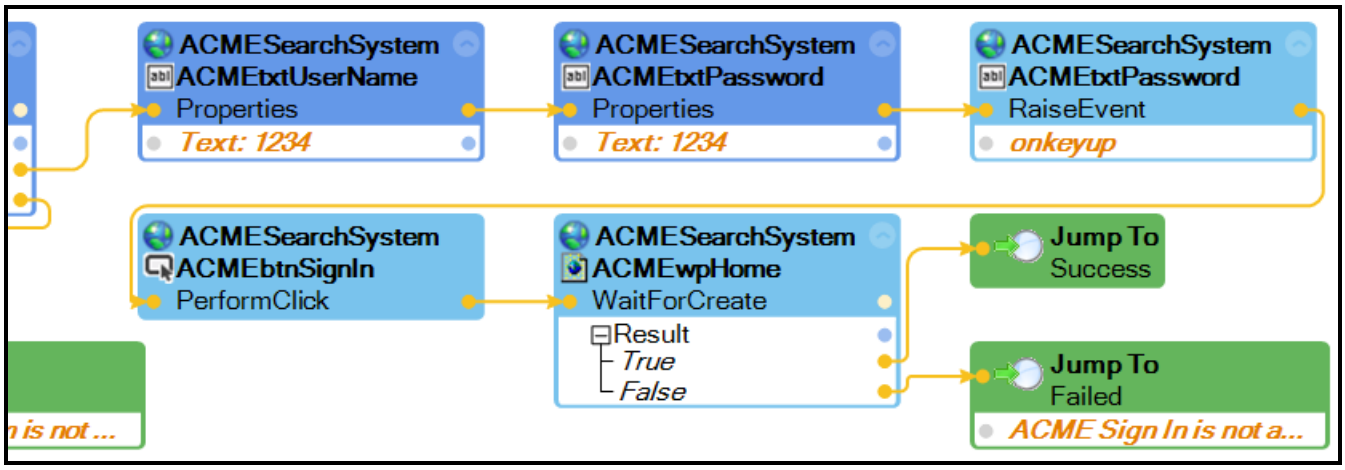
5. Search for checkLengthP() (roughly around line 292). A method, onkeyup, calls the checklengthP() function.

```
placeholder="User Name" onkeyup="return checkLengthL();"/>
287 </td>
288 </tr>
289 <tr>
290 <td>
291 <div id="pass_error">Please enter a password of at least 4 characters.</div>
292 <input class="input_text" name="user_pass" id="user_pass" size="50" placeholder="Password"
    onkeyup="return checkLengthP();" type="password"/>
293 </td>
294 </tr>
295 <tr>
296 <td><input id="login_button" type="submit" value="Sign In" disabled="true"
    onclick="javascript:checkData();"/></td>
297 </tr>
298 </table>
```

6. Stop the interrogation.
7. Return the ACME_P_SignIn automation to a Design window.
8. In the automation, delete the automation link between the ACMEtxtPassword.Text and ACMEbtnSignin.PerformClick design blocks.
9. From the Object Explorer, add the ACMEtxtPassword.RaiseEvent method to the automation. Remember to use the Configure Type button to search for the method.
10. On the RaiseEvent design block, click the **evt**property and select **onkeyup**.



11. Connect the components as shown in the following image.



12. From the menu, select **File > Save All** to save your edits.
13. Debug the solution. You successfully log in to the ACME Search System.

Exercise: Completing navigation automation

Scenario

The expected result of the business case is to locate the nearest store from the customer's zip code. With the home navigation and sign-in automations completed, the remaining automations to locate the nearest store in the ACME Search System are needed. A temporary automation that handles the button click from the Windows test form is required to debug the solution.

Your assignment

Complete the following tasks:

- Create the ACME_P_SearchByZip automation to search for the nearest store passing in a zip code from the user interface and return the store address in the automation. Use two exit points, Success and Failed. With a successful search, return the store address; if the search fails, return a message. Also use Label/Jump to organize the automation. Consider using Object Explorer to extract a proxy for the input parameter.
- Create the ACME_P_GetNearestStore automation by passing in string parameter for the zip code and by calling the ACME_P_NavToHome and the ACME_P_SearchByZip automations. Use two exit points, Success and Failed. With a successful search, return the store address; if the search fails, return a message. Consider using Object Explorer to extract a proxy for the input parameter.
- Create the ACME_E_btnTestClicked automation that calls the ACME_P_GetNearestStore automation and returns the store address or a failed message to a MessageDialog.

Detailed steps

Create the ACME_P_SearchByZip automation

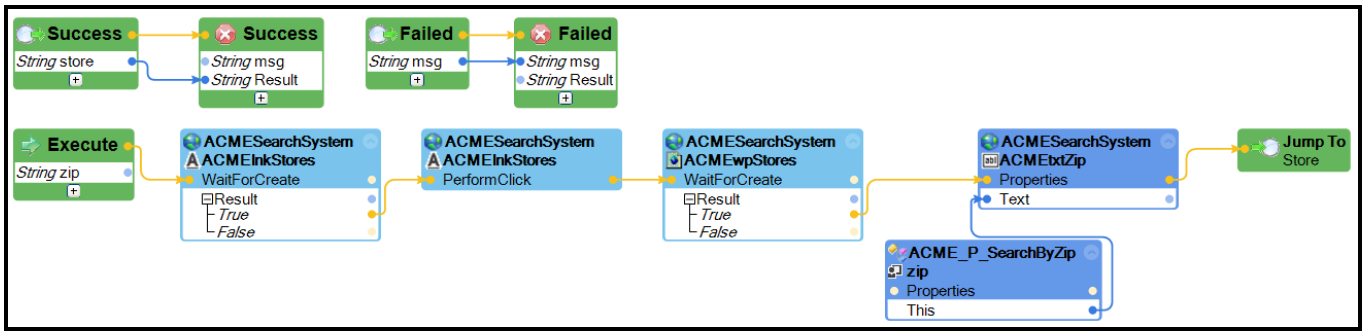
1. In the ACME_P_NavToHome automation, delete the frmTest.btnTest.Click design block.
2. In the Solution Explorer, in the TrainingWebAdapter project, right-click the **Procedures** folder and select **Add Automation**. The Add New Item window is displayed.
3. In the **Name** field, enter **ACME_P_SearchByZip** and click **Add**. The Add New Item window closes and the automation opens in a new designer tab.
4. Add the following design blocks to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
Right-click in the automation	Select **Add Entry Point**	• Click **+** to add a parameter • Rename the param1 **zip**.
ACMESearchSystem	ACMElnkStores.WaitForCreate method	• Expand the result. • If the link exists, click the

		link.
Right-click in the automation	Add two labels	• Rename one label **Success** and one **Failed**. • On the Success label, click **+** to add one parameter. • Rename the param1 **store**. • On the Failed label, click **+** to add one parameter. • Rename param1 **msg**.
Right-click in the automation	Add two exit points	• Rename one **Success** and one **Failed**. • On Failed Exit point, click **+** to add one parameter. • Change param1 data type to **String** and rename **msg**. • Change Void parameter to **String** data type.
ACMESearchSystem	ACMElinkStore.PerformClick method	
ACMESearchSystem	ACMEwpStores.WaitForCreate method	
ACMESearchSystem	ACMEtxtZip.Text property	
Right-click in automation	Select **Add Label**	Rename Label1 to **Store**.
Right-click in the automation	Select **Jump To > Store**	

5. In Object Explorer, select the ACME_P_SearchByZip automation and click **+** to expand the automation and the Execute node. The zip input parameter is displayed.
6. Select the **zip** parameter.
7. In the Object Inspector, click **Configure Type**. The Configuration window is displayed.
8. In the Configuration window, click **+** in front of Properties to expand the options.
9. Check the **This** property and click **OK** to add the property to the Object Inspector and close the window.
10. In the Object Explorer, click **Show Properties Only**.
11. Click and drag the **This** property to the automation. The design block is a proxy of the input parameter.

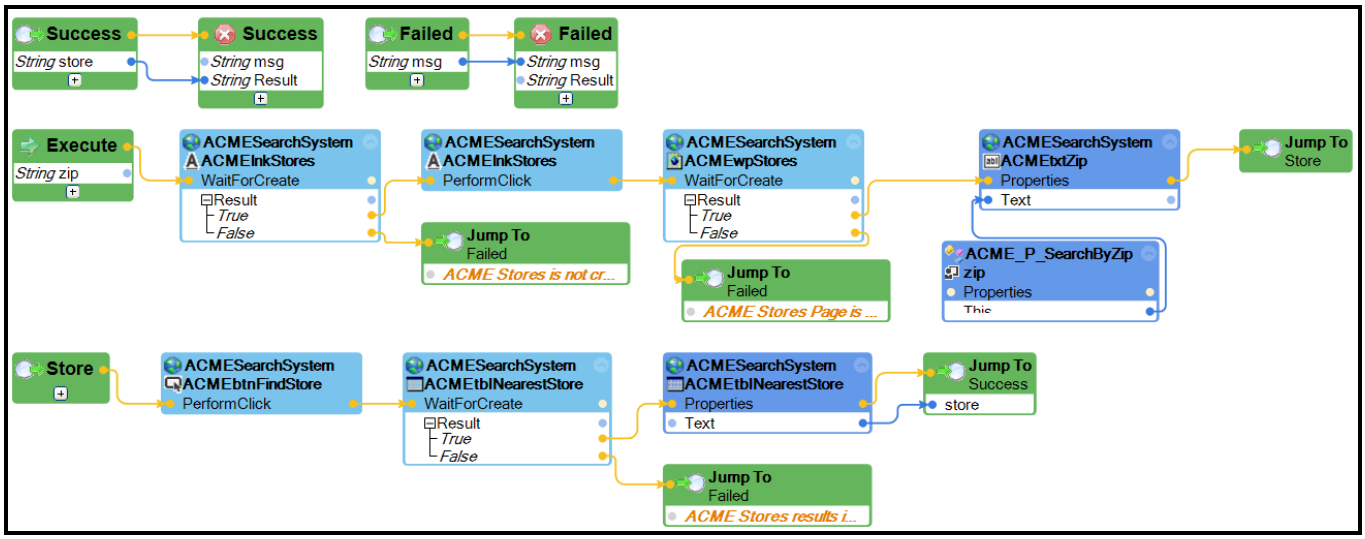
12. Connect the design blocks as shown in the following image.



13. Add the following design blocks to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
ACMESearchSystem	ACMEbtnFindStore.PerformClick method	
ACMESearchSystem	ACMEtblNearestStore.WaitForCreate method	• Wait for the Results page to load before storing the store address. • Expand the Results parameter.
ACMESearchSystem	ACMEtblNearestStore.Text property	• This property contains the store address.
Right-click in the automation	Select **Jump To > Success**	
Right-click in the automation	Select **Jump To > Failed** three times	• Enter one message into each Failed design block: • **ACME Store is not created.** • **ACME Stores Page is not available.** • **ACME Stores results is not available.**

14. Connect the components as shown in the following image.



15. From the menu, select **File > Save All** to save your edits.

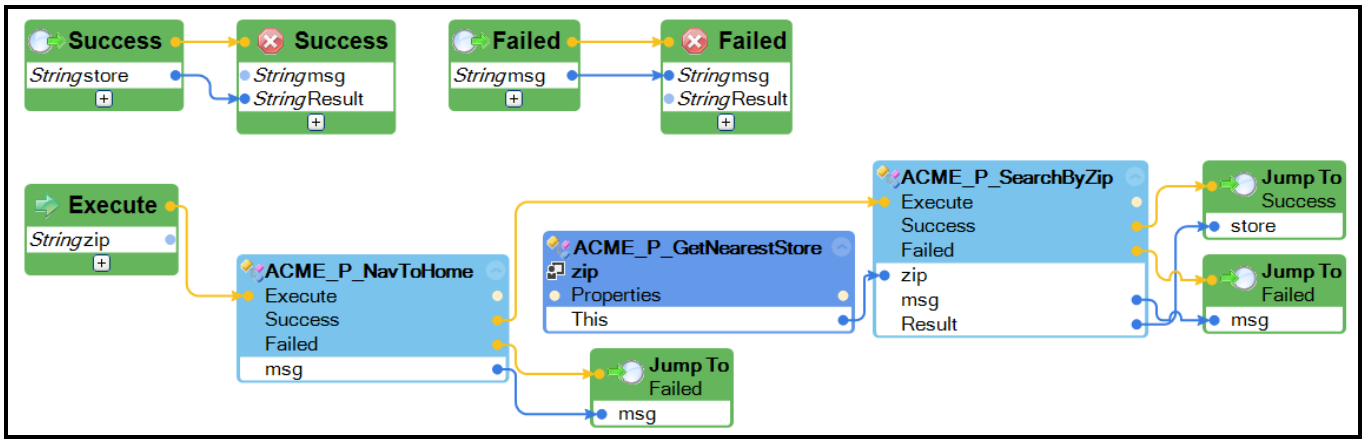
Create the ACME_P_GetNearestStore automation

1. In the Solution Explorer, in the TrainingWebAdapter project, right-click the **Procedures** folder and select **Add Automation**. The Add New Item window is displayed.
2. In the **Name** field, enter **ACME_P_GetNearestStore** and click **Add**. The Add New Item window closes and the automation opens in a new designer tab.
3. Add the following design blocks to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
Right-click in the automation	Select **Add Entry Point**	• Click **+** to add a parameter. • Rename the param1 to **zip**.
ACME_P_NavToHome	**Execute**method	• Call the automation to confirm the adapter and sign in to the web application.
Right-click in the automation	Add two labels	• Rename one label **Success** and one **Failed**. • On the Success label, click **+** to add one parameter. • Rename the param1 to **store**. • On the Failed label, click **+** to add one parameter. • Rename param1 to **msg**.
Right-click in the automation	Add two exit points	• Rename one label **Success** and one **Failed**. • On the Failed label, click **+** to add one parameter. • Rename param1 to **msg**. • Change Void parameter to **String** data type.

ACME_P_SearchByZip	**Execute** method	Call the automation to perform the search on a zip code.
ACME_P_GetNearestStore	zip.This property	Create a proxy for the zip input parameter using the This property.
Right-click in automation	Select **Jump To > Success**	
Right-click in the automation	Select **Jump To > Failed**	Create two Jump To design blocks.

4. Connect the automation links as shown in the following image.



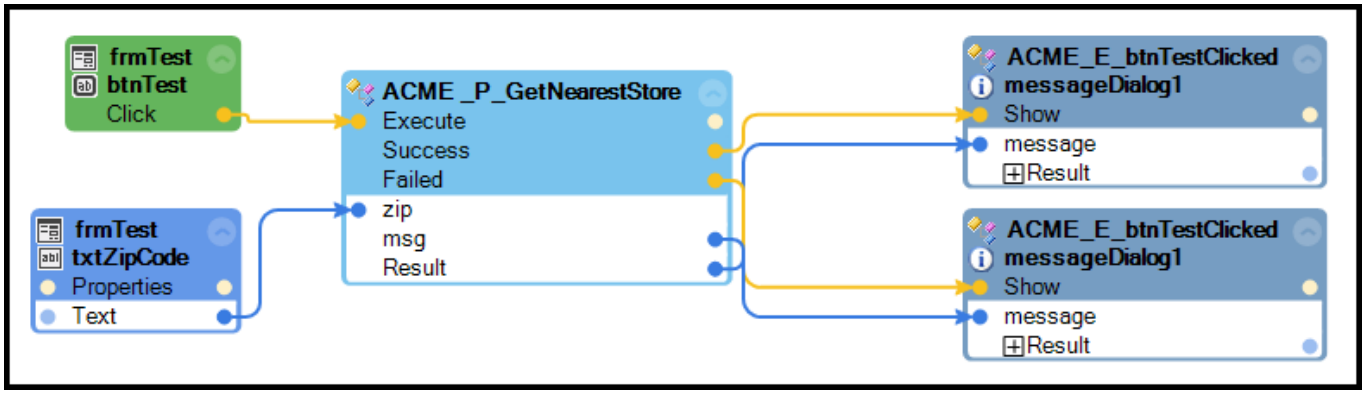
Create the ACME_E_btnTestClicked automation

1. In the Solution Explorer, add a folder to the TrainingWebAdapter and name the folder **Events**.
2. In the Solution Explorer, add a folder to the new Events folder and name the folder **Controls**.
3. In the Solution Explorer, in the TrainingWebAdapter project, right-click the **Controls** folder and select **Add Automation**. The Add New Item window is displayed.
4. In the **Name** field, enter **ACME_E_btnTestClicked** and click **Add**. The Add New Item window closes and the automation opens in a new designer tab.
5. Add the following design blocks to the automation. Use the **Configure Type** icon in the Object Inspector to locate properties, events, and methods.

Source project Items	Design Name	Description
frmTest	btnTest.Clicked event	Perform this automation when the user clicks the button.
ACME_P_GetNearestStore	Execute method	Call the automation to confirm the adapter, sign in to the web application, search for the store address, and return the address.
frmTest	txtTest.Text property	Use to pass in the zip code from the frmTest text field.

Toolbox	Add two MessageDialog	On the Choose Method Overflow window, select the second radio button to display a message.

6. Connect the automation links as shown in the following image.



Verify your work

1. Start the debugger.
2. In the test window, enter a five-digit zip code.
3. Click **Test**. The web adapter starts, signs in to the applications, and performs the search on the zip code. A message box displays with the store address.
4. Optionally, you can test the error messages within the automations.

INTERACTION FRAMEWORK

This lesson group includes the following lessons:

- Interaction Framework structure
- Working with multiple-project solutions
- Interacting between applications and projects

Interaction Framework structure

Exercise: Adding a user interface

Scenario

The first two application projects are completed. The solution needs the final project to allow the user to interact with the CRM application and ACME Search system from the user interface which displays the customer information from CRM and returns the closest store based on the customer's zip code.

Your Assignment

Reviewing the solutions structure, you need a user interface project, named Main-UI, which contains a Windows form to display the customer information and display the nearest store address.

In this part of the solution, complete the following:

- Add a project, named **Main-UI**.
- Add a Windows form to Main-UI, named **CustomerInformation**.
- Add the following .Net controls from the Toolbox and update the their properties from the following table:

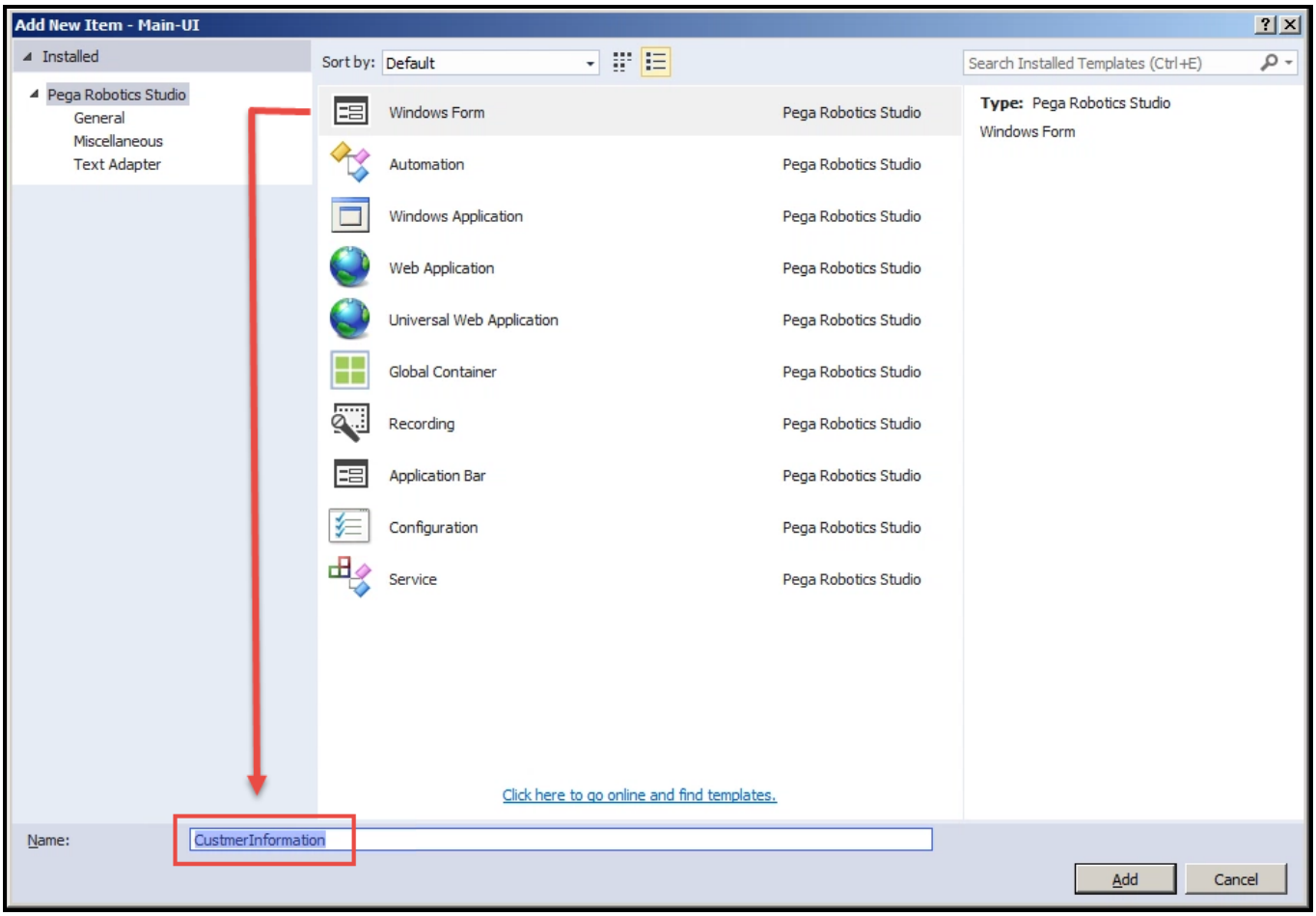
Object Name	Properties to update	Property Value
Customer Information	Text TopMost	Customer Information True
label1	Design Name Text Font > Bold Font > ForeColor -	MainlblCurrentAcct Current Account True Highlight
label2	Design Name Text Text Font – Bold Font > ForeColor	MainlblGetAcct Get Account True Highlight
label3	Design Name Text Text Font – Bold Font > ForeColor	MainlblName Customer Name (True) Highlight
label4	Design Name Text Text Font – Bold	MainlblAddress Address True
label5	Design Name Text Text Font – Bold	MainlblCity City True

label6	Design Name Text Text Font – Bold	MainlblState State True
label7	Design Name Text Text Font – Bold	MainlblZip ZIP Code True
combobox1	Design Name MaxDropDownItems Sorted	MaincmbGetAcct 100 True
textbox1	Design Name Text	MaintxtCurrentAcct -------- (eight dashes)
textbox2	Design Name	MaintxtName
textbox3	Design Name	MaintxtAddress
textbox4	Design Name	MaintxtCity
textbox5	Design Name	MaintxtState
textbox6	Design Name	MaintxtZip
Groupbox1	Design Name Text	MaingboxNearestStore ACME Nearest Store
Button1	Design Name Text	MainbtnNearestStor Find Store
Label 8	Design Name Text	MainlblNearestStore ----- (five dashes)

Detailed Steps

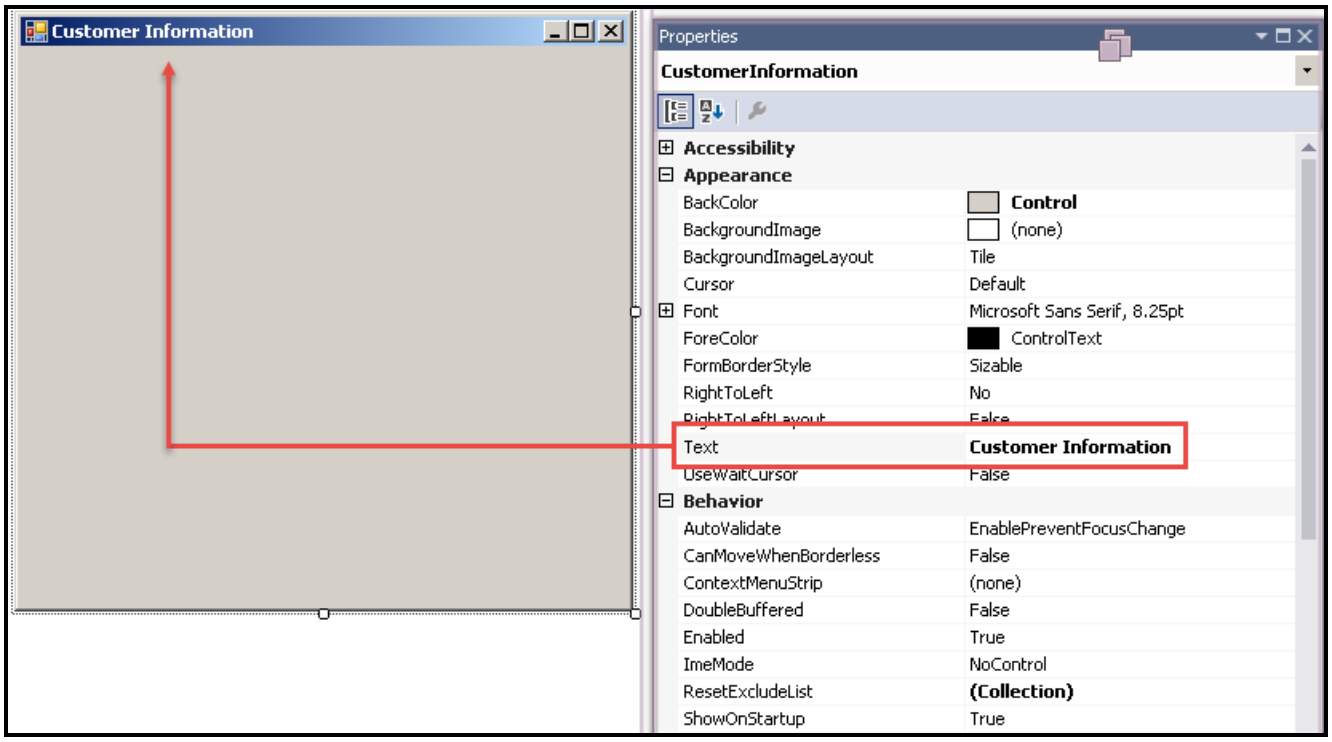
1. From the Solution Explorer, right-click the **TrainingCertification** soluiton and select **Add > New Project**.
2. From the Add New Project window, confirm **Empty Pega Robotics Project** is highlighted.
3. In the Name field, enter **Main-UI** and click **OK**. The window closes and the Solution Explorer displays the new project.
4. From the Solution Explorer, right-click the **Main-UI** project and select **Add > New Windows Form...**. The Add New Item window displays with the Windows Form item highlighted.
5. In the **Name** field, enter **CustomerInformation**.

6. Click **Add**. The CustomerInformation.os window appears in the Studio Designer window.



7. In Object Explorer, highlight **CustomerInformation** to display the full list of available properties in the Properties window.

8. Locate the Text property and change its name to **Customer Information**.



9. In the Properties window, set the TopMost property to **True**.
10. Select **File > Save**.

    The account number is a unique identifier for the solution to use to differentiate and recognize the different customers. The business case stated that the agent might work with numerous customers simultaneously. We use a combo box to collect the account numbers, so the agent can switch customers when selecting a different account number in the combo box.

11. The controls you need for the Customer Information window are in the Common Controls section of the Toolbox. Expand the **Common Controls** and **Containers** section of the Toolbox and locatecontrols.

12. Drag and drop the following controls onto the **Customer Information** window and arrange the controls as shown below:
    - 8 Label controls
    - 6 Textbox controls
    - 1 ComboBox control
    - 1 GroupBox control
    - 1 Button control

13. Select **File > Save**.

    For this course, you will add the adapter name in front of the object type prefix.

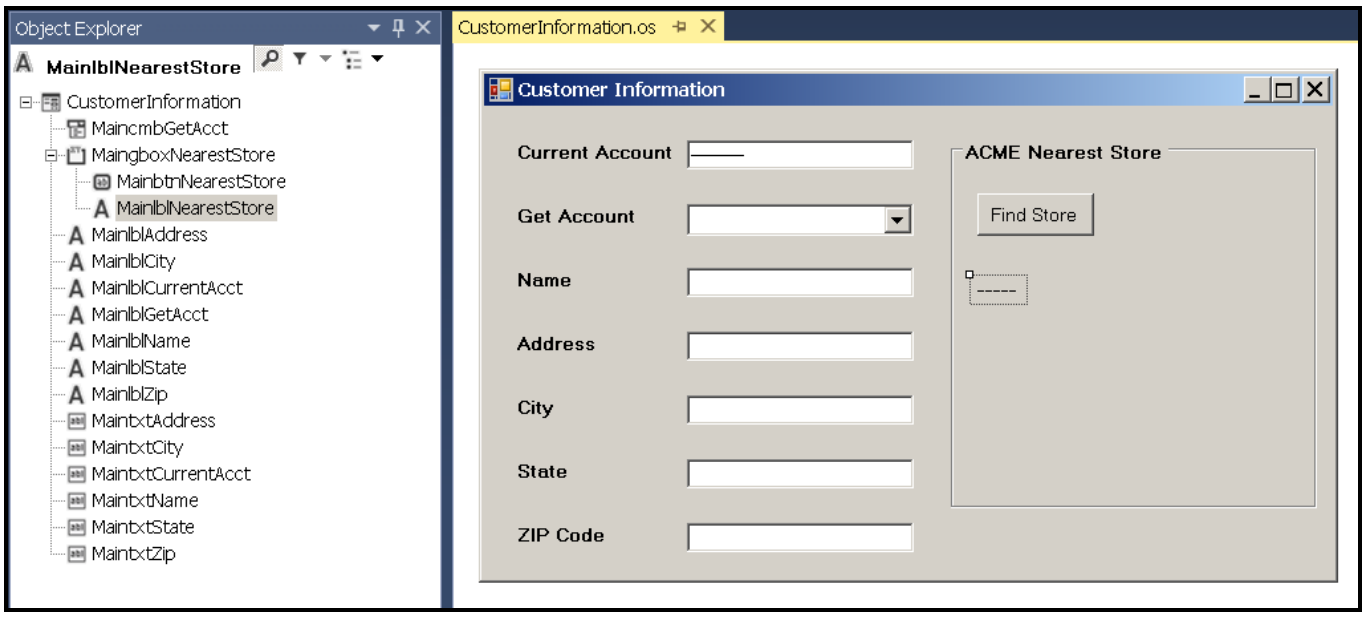
14. In Object Explorer, select the **CustomerInformation** control.
15. Using the Properties window, change each control's Properties.

Object Name	Properties to update	Property Value

Customer Information	BackColor	Window
label1	Design Name Text Font > Bold Font > ForeColor -	MainlblCurrentAcct Current Account True Highlight
label2	Design Name Text Text Font – Bold Font > ForeColor	MainlblGetAcct Get Account True Highlight
label3	Design Name Text Text Font – Bold Font > ForeColor	MainlblName Customer Name True Highlight
label4	Design Name Text Text Font – Bold	MainlblAddress Address True
label5	Design Name Text Text Font – Bold	MainlblCity City True
label6	Design Name Text Text Font – Bold	MainlblState State True
label7	Design Name Text Text Font – Bold	MainlblZip ZIP Code True
combobox1	Design Name MaxDropDownItems Sorted	MaincmbGetAcct 100 True
textbox1	Design Name Text	MaintxtCurrentAcct -------- (eight dashes)
textbox2	Design Name	MaintxtName
textbox3	Design Name	MaintxtAddress
textbox4	Design Name	MaintxtCity
textbox5	Design Name	MaintxtState
textbox6	Design Name	MaintxtZip
Groupbox1	Design Name Text	MaingboxNearestStore ACME Nearest Store
Button1	Design Name Text	MainbtnNearestStore Find Store

Label 8	Design Name Text	MainlblNearestStore ----- (five dashes)

The Object Explorer window should look like the following example:



16. Select **File > Save** to save the edits to the Customer Information window.

Exercise: Adding the Interaction Framework components

Scenario

From the business case, you want to store the customer account information in the CRM application to be displayed in the Customer Information window. To communicate and pass data efficiently from one project to another, use the Interaction Framework for the solution.

Your assignment

In this part of the solution, complete the following tasks:

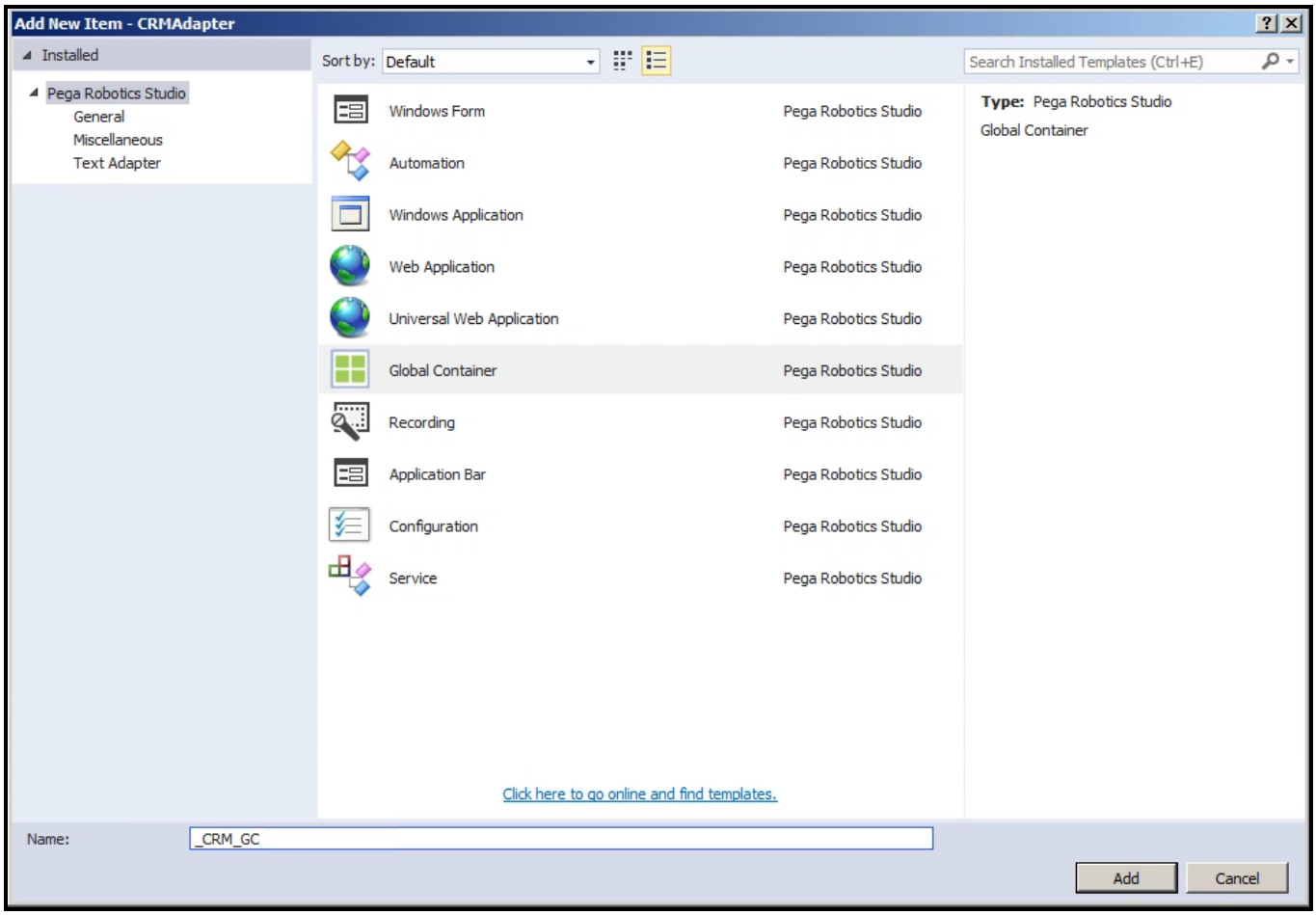
- Add a global container to the CRMAdapter, TrainingWebAdapter, and Main-UI projects, named _**CRM_GC**, **_ACME_GC**, and **_Main_GC**, respectively.
- Add the interaction-call.xml to the Main-UI folder in the TrainingCertification solution directory.
- Add and configure the Interaction Manager component to the global containers in the CRMAdapter TrainingWebAdapter and Main-UI projects. Rename each Interaction Manager component to **IntMgr**.

Use the supplied interaction-call.xml in the Pega Robotics Training Exercise zip file located in the Related Documents section of this course.

Detailed steps

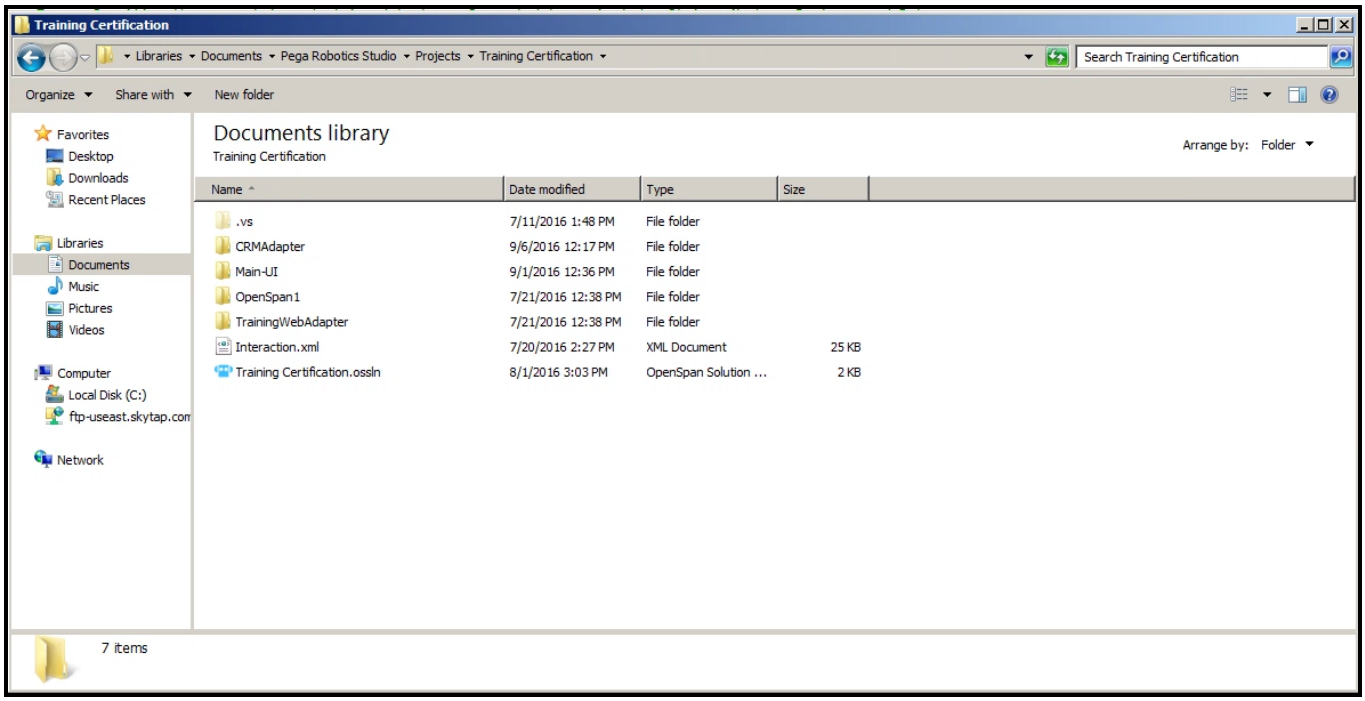
Follow these steps to set the add Interaction Framework components.

1. In Solution Explorer, right-click the **CRMAdapter** project.
2. Select **Add > New Item**. The Add New Item – CRMAdapter window opens.
3. In the list, click **Global Container**.
4. In the **Name** field, enter **_CRM_GC**.

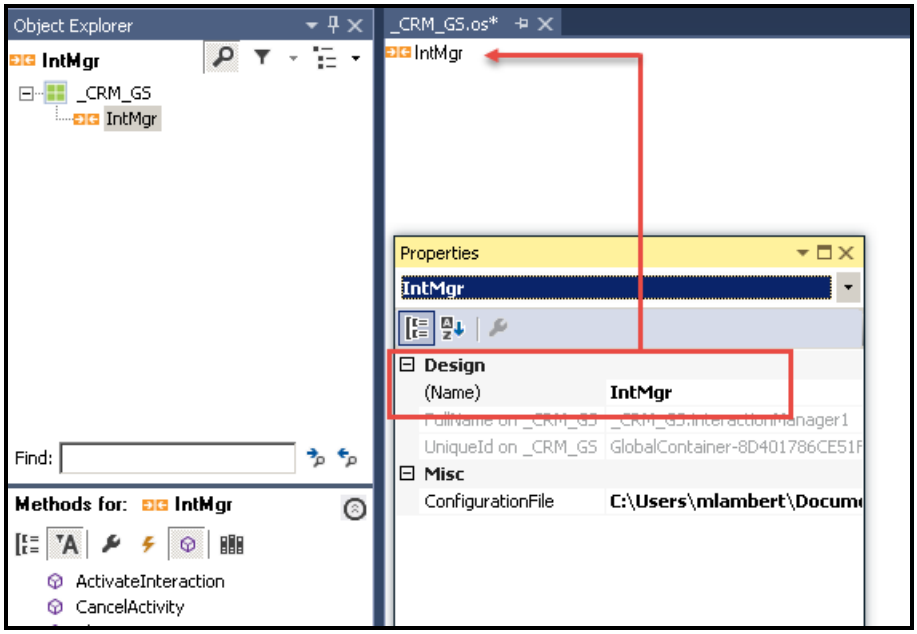


5. Click **Add**. The window closes.
6. Using steps 1-5:
   a. Create a global container for the Main-UI project and name it **_Main_GC**.
   b. Create a global container for the TrainingWebAdapter and name it **_ACME_GC**.
7. From the **Related Documents** of this course, open the Pega Robotics Training Exercise file.
8. Locate the interaction-call.xml file.

9. In the solution directory located in your My Documents, copy and paste the interaction-call.xml to the Main-UI project folder **C:\Users\{userid}\Documents\Pega Robotics Studio\Projects\TrainingCertification\Main-UI**.



10. In Solution Explorer, double-click **_CRM_GC.os** to open the global container in a designer window.
11. Open the Toolbox window.
12. Click the **Interaction Management** section.
13. Click the **InteractionManager** component and drag it onto the **_CRM_GC.os** window.
14. In the **Properties** window, in the **(Name)** field, rename **interactionManager1** to **IntMgr**.



15. In the **Properties** window, click the **Configuration File** property. An ellipses button is displayed.

16. Click the **Ellipses** button to display the Open File window.
17. In the **Open File** window, in the solution directory, navigate to the **Main-UI** project folder.
18. Select the **interaction-call.xml** file.
19. Click **Open**. The Open File window closes.
20. Using steps 10-19, add an Interaction Manager component to the _Main_GC and the _ACME_GC.
21. From the menu, select **File > Save All** to save the new global containers to the projects and the Interaction Manager components.

Exercise: Modifying the interaction.xml

Scenario

The business case requires that the CRM customer account information and the nearest store's address be displayed on the Customer Information window.

Your assignment

Verify that the interaction-call.xml document contains the seven account information items, their data types, and default values, and edit the XML document for any items missing.

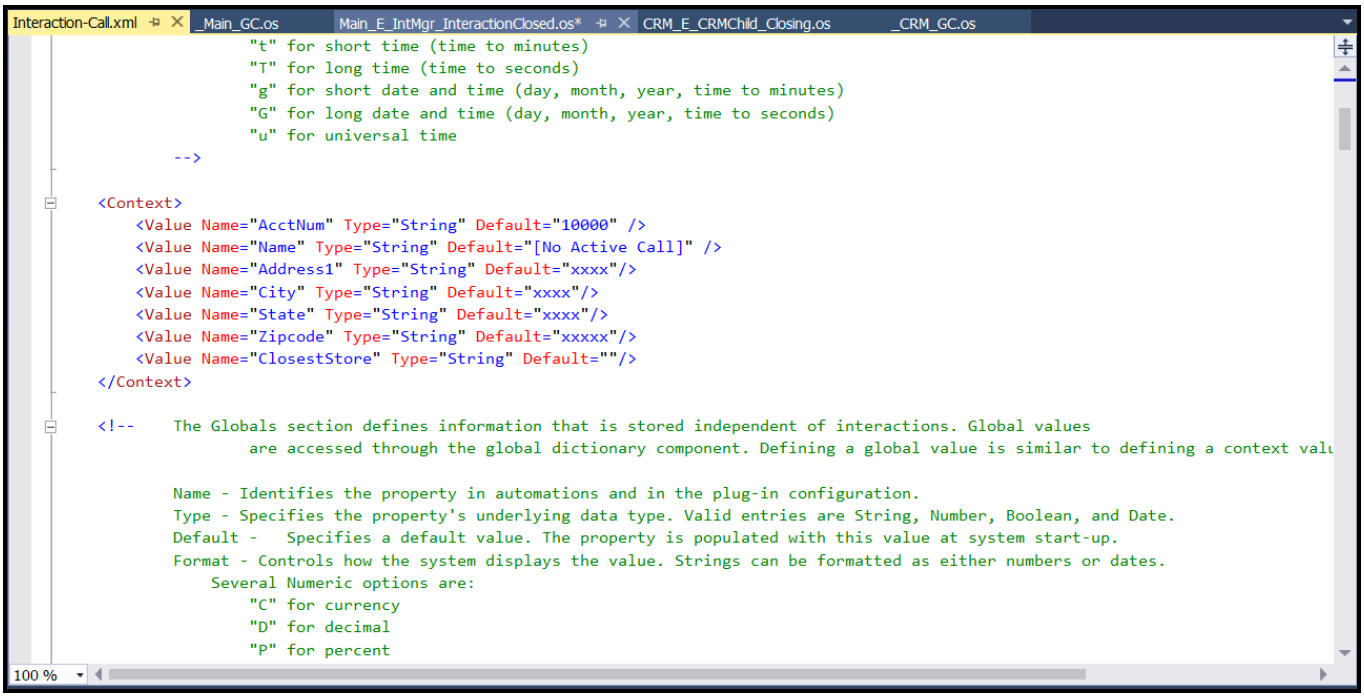
- AcctNum, String, 10000
- Name, String, [No Active Call]
- Address, String, xxxx
- City, String, xxxx
- State, String, xxxx
- ZipCode, String, xxxx
- NearestStore, String, xxxx

Detailed steps

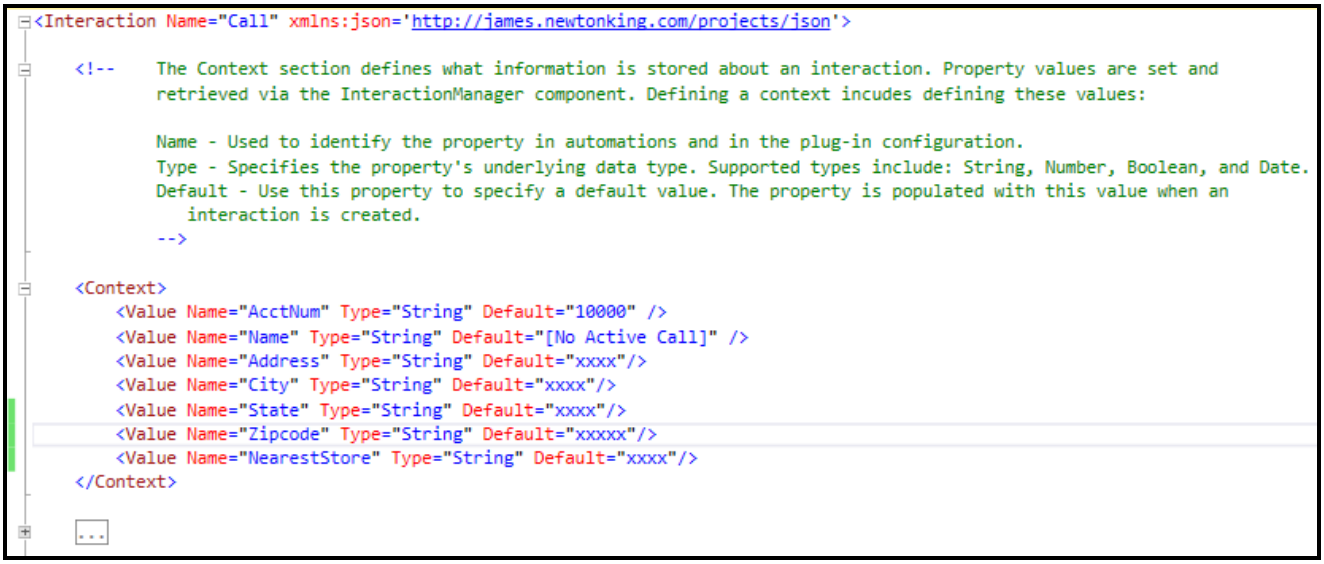
Follow these steps to modify the interaction-call.xml.

1. In Solution Explorer, double-click **_Main_GC.os** to open it in a designer window.
2. In the _Main_GC window, right-click **IntMgr**.

3. Select **Modify Configuration**. The Interaction_Call.xml file opens in the Designer window.



4. Locate the Context section in the Interaction-Call.xml file.
5. Add the following new context values after the last context value in the Context section. For each edited line in the XML file, a yellow marker is displayed in the left margin.
   - <Value Name=”State” Type=”String” Default=”xxxx”/>
   - <Value Name=”ZipCode” Type=”String” Default=”xxxx”/>
   - <Value Name=”NearestStore” Type=”String” Default=”xxxx”/>



6. Select **File > Save All** to save the edits to the XML file. The yellow markers in the file display green.
7. From the Build menu, select **Build > Clean Solution**.

8. When finished, select **Build > Rebuild Solution**. This ensures that the new XML entry is available to use in the solution and removes the old XML stored in the memory cache.

Working with multiple-project solutions

Exercise: Creating a project-to-project reference

Scenario

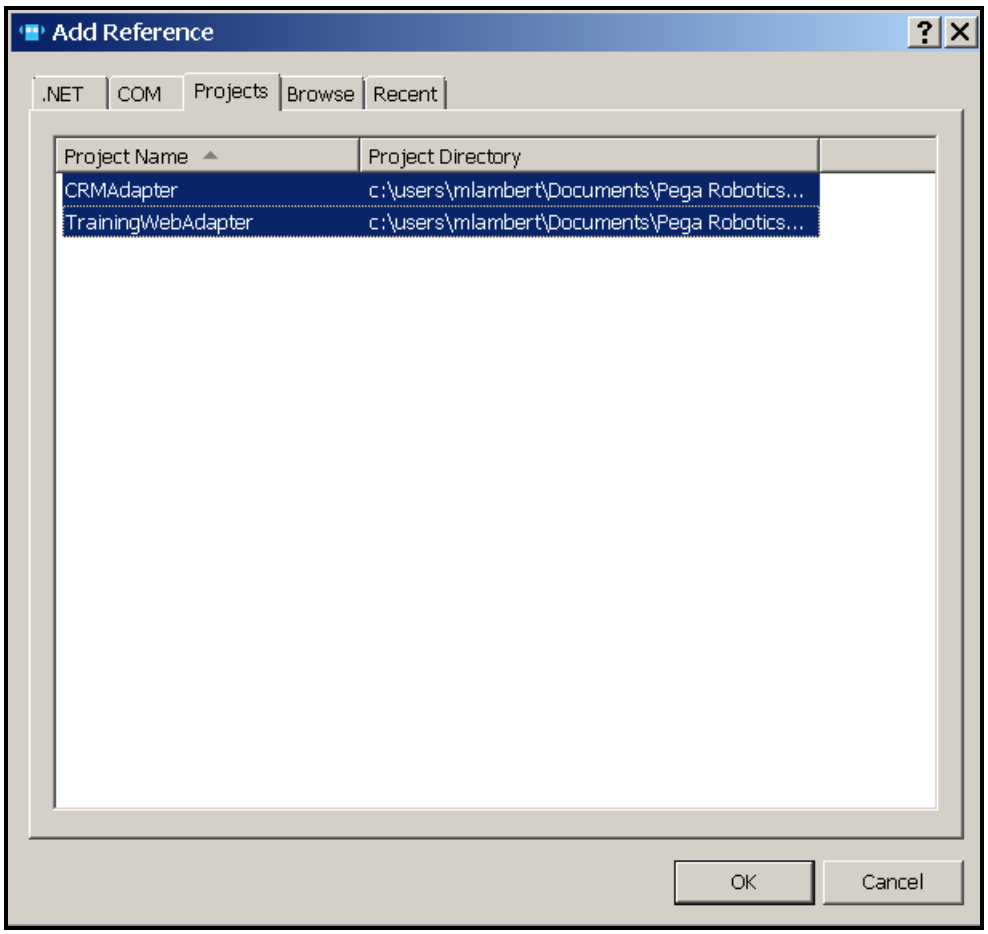
The business case states that the final solution should act, and be deployed, as one solution. The Main-UI project contains the user interface that is the primary conduit to the other applications. The Main-UI project references the other projects to meet the business need as well as be set as the start-up project for the solution.

Your assignment

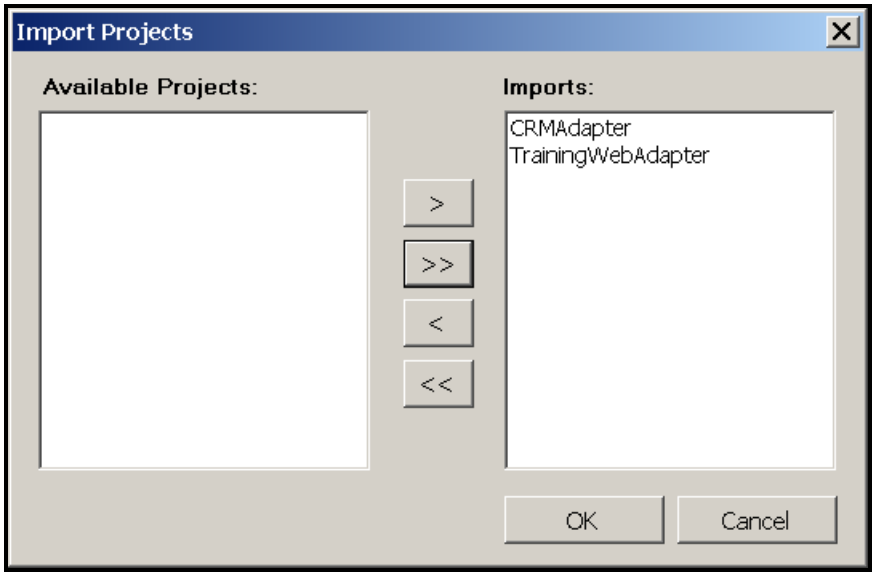
Using the Main-UI, make a project-to-project reference to the CRMAdapter and TrainingWebAdapter projects. Configure the Main-UI project as the start-up project. Set the TrainingWebAdapter test Window form to not display in the completed solution.

Detailed steps

1. In the Solution Explorer, right-click the **Main-UI** project and choose **Add Reference**. The Add Reference window opens.
2. On the Projects tab, select **CRMAdapter** and **TrainingWebAdapter** projects while holding down **Shift**.
3. Click **OK**. The Add Reference window closes, and the project is displayed in the Reference folder under the Main-UI project in the Solution Explorer.



4. In the Solution Explorer, right-click the **Main-UI** project and choose **Manage Imported Projects**. The Import Projects window opens.
5. From the Available Projects list, click **>>**. This moves both projects into the Imports list.



6. Click **OK**. This closes the window and saves the imported project to the Main-UI project.

7. In the Solution Explorer, right-click the **Main-UI** project and select **Set as StartUp Project**. The Main-UI text turns bold.
8. In the Solution Explorer, double-click **frmTest**. The Windows form is displayed in the a designer tab.
9. In the Properties window, change the **ShowOnStartUp** property to **False**.

Exercise: Adding and removing items from the combo box

Scenario

In reviewing the case, the account numbers should display in the combo box on the Customer Information window to allow users to select and move between opened accounts easily. CRM_E_CRMfrmLogin_Created logs in to the CRM application and presses the User 1 button to access the first customer. Based on the case, when a customer account opens, the solution should update all needed information and start the Interaction Framework to begin to store and share data. The business case states that the CRM account number should be used as the interaction key for the framework.

Your assignment

In this part of the solution, complete the following:

- Create an automation to start an interaction from the CRMChild window
- Create an automation to add the customer's account number to the combo box after the interaction starts.
- Create an automation to close the interaction when the CRMChild window closes.
- Create an automation that removes the account number from the combo box when the CRMChild window closes.

Detailed steps

Follow these steps to start and close an interaction and to add and remove an item from the combo box.

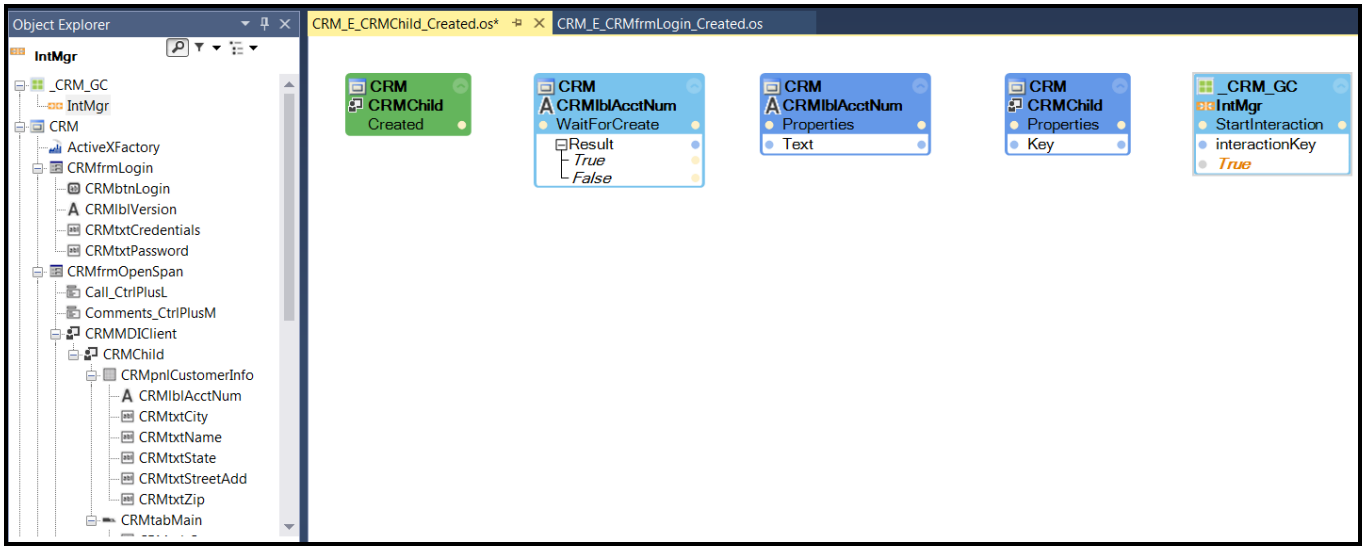
Create an automation to start an interaction from the CRMChild window

1. In Solution Explorer and within the CRMAdapter project, expand the **Events** folder.
2. Right-click the **Controls** folder and select **Add > New Automation.**
3. In the **Add New Item** window, enter **CRM_E_CRMChild_Created** in the **Name** field.
4. Click **Add**.
5. Add the following design blocks to the CRM_E_CRMChild_Created automation. If a property, event, or method does not exist for a control in the Object Inspector, click the **Configure Type** icon to select

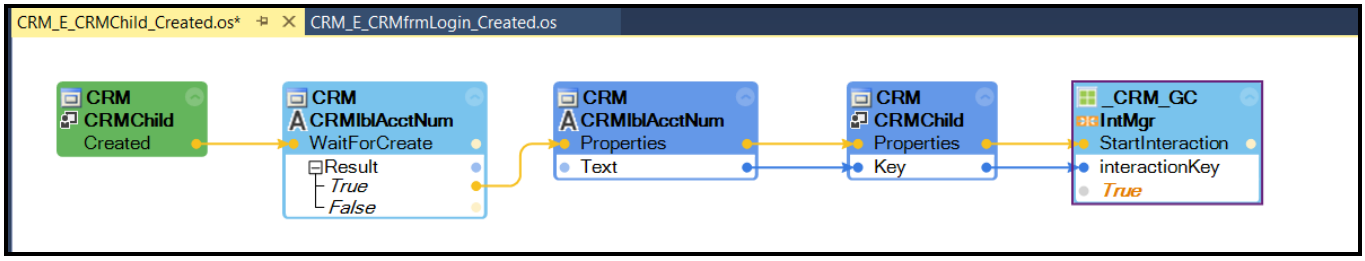
more options to display.

Project Item Source	Control	Description/Configuration
CRM	CRMChild.Created event	For this solution, any time a New Call window creates (opens), the automation logic runs.
CRM	CRMlblAcctNum.WaitForCreate method	This method will wait for the Account Number field to create before assigning its value to the Key property of the CRMChild.
CRM	CRMlblAcctNum.Text property	The text of the account number is the key.
CRM	CRMChild.Key property	Assign the Key property of the CRMChild as the account number text.
_CRM_GC	CRMIntMgr.StartInteraction (2 parameters) method	On the design block, click Activate and set the property to True.

The automation should be similar to the following image .



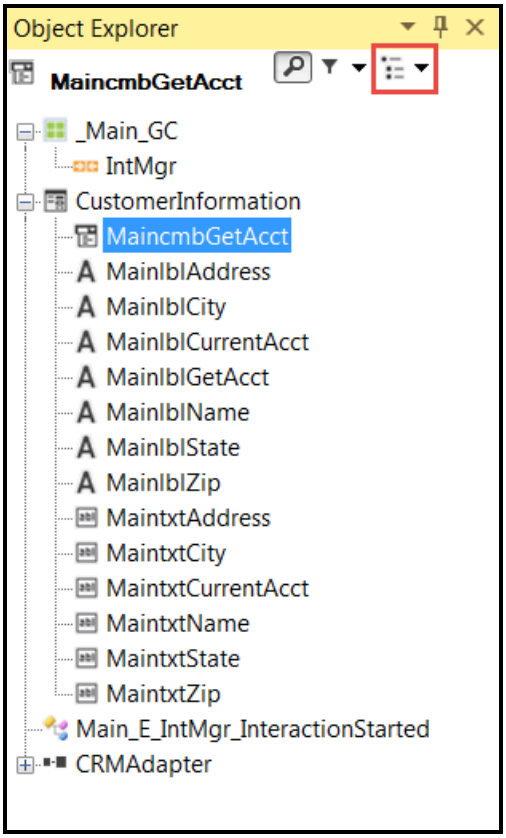
6. In the automation, connect the design blocks so they look similar to the following image .



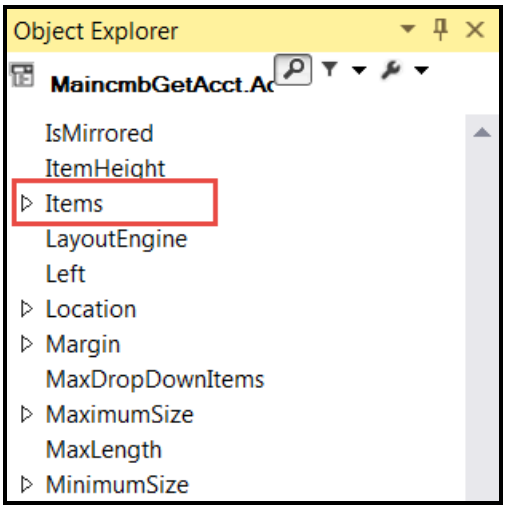
7. From the menu, select **File > Save All.**

Create an automation to add the account number to the combo box after the interaction starts

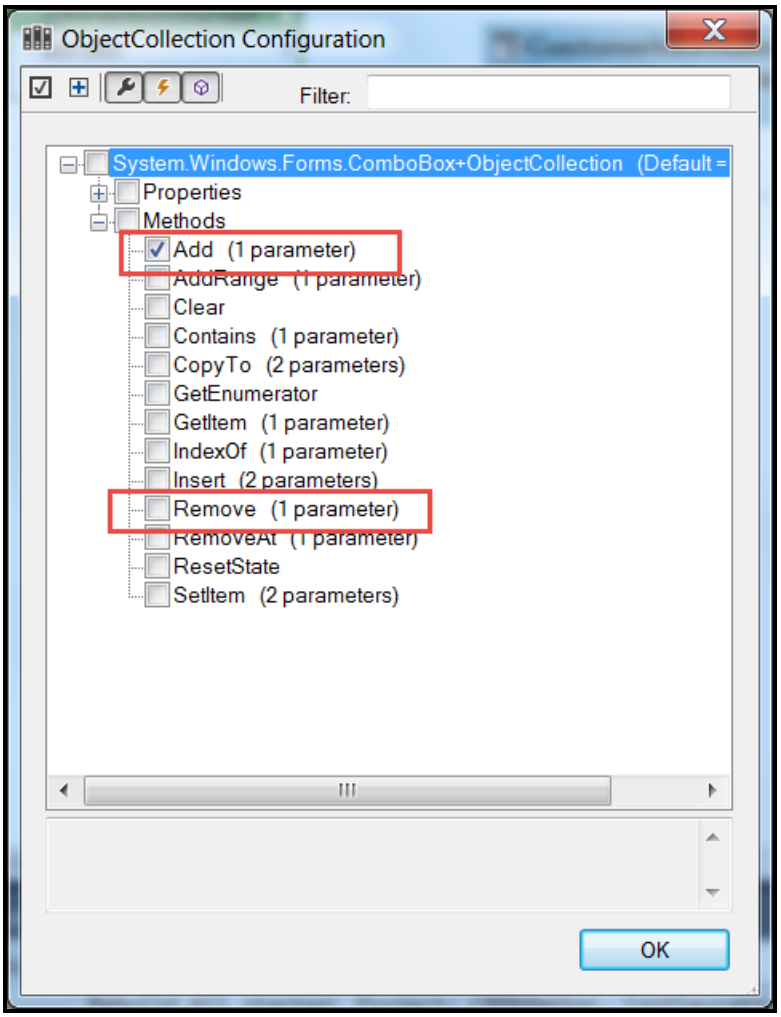
1. In Solution Explorer, right-click the **Main-UI** project and select **Add > New Folder.**
2. In the Solution Explorer, enter **Events** to rename the folder.
3. In Solution Explorer, right-click the **Events** folder and select **Add > New Folder**.
4. In the Solution Explorer, enter **Interactions** to rename the folder. This folder contains all automations starting with Interaction Manager events.
5. In the Solution Explorer, right-click the **Interactions** folder and select **Add > New Automation.**
6. In the **Name** field on the Add New Item window, enter **Main_E_IntMgr_InteractionStarted**.
7. Click **Add**. The automation opens in a designer window.
8. From the Object Hierarchy, select the **IntMgr from _Main_GC** to highlight it.
9. In the Object Inspector, click **Show Events Only** (lightning bolt) button.
10. From the list of events, click and drag the InteractionStarted event to the automation. The design block displays.
11. In the Object Hierarchy, select the **MaincmbGetAcct** control to highlight it.
12. In the upper right corner of Object Explorer, select **Explore Component Properties**.



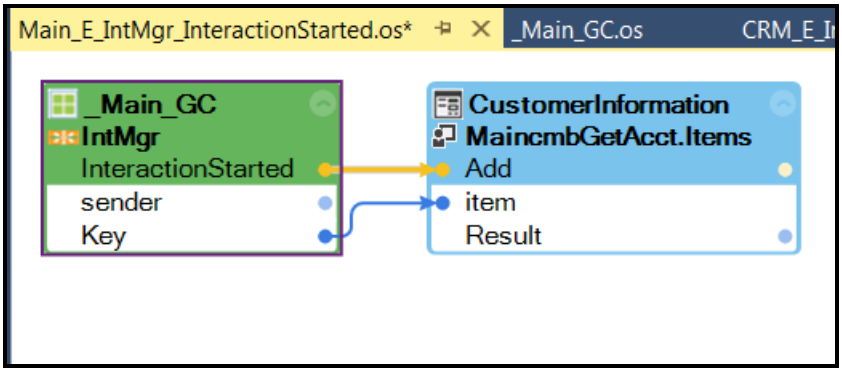
13. Scroll down the list of components and select **Items** to highlight it.



14. In Object Inspector, click **Configure Type** (books) button. The Object Configuration window displays.
15. In the Object Configuration window, expand the **Methods** category.
16. From the Object Configuration window, select the **Add (1 parameter)** method and the **Remove (1 parameter)** method.



17. Click **OK**. The Object Configuration window closes.
18. In the Object Inspector, click **Show Methods Only** button.
19. Click and drag the **Add** method to the automation.
20. In the automation, connect the design blocks as showin in the following image .



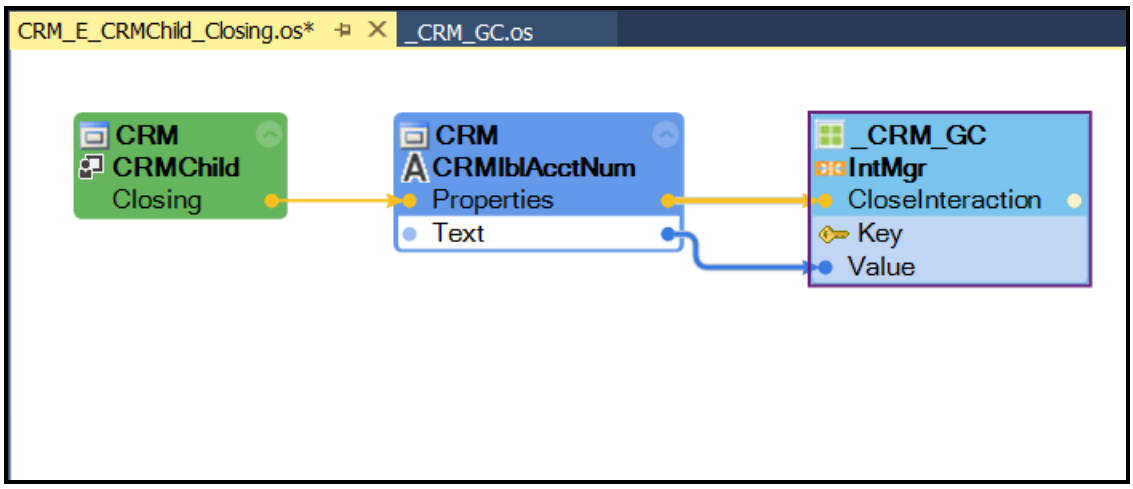
21. From the menu, select **File > Save All**.
22. Press **F5** to test the automations. The combo box has the account number 23453.
23. Click **User 2** in the CRM application. Account number 23454 populates the combo box.
24. Press **Shift + F5** to stop debugging.

Create an automation to close the interaction when the CRMChild window closes

1. In the Solution Explorer, add an automation, named **CRM_E_CRMChild_Closing**, in the Controls folder of the CRMAdapter project. The automation opens in a design window.
2. Add the following design blocks to the CRM_E_CRMChild_Closing automation.

Project Item Source	Control	Description/Configuration
CRM	CRMChild.Closing event	For this solution, any time a CRMChild window closes, the automation logic runs.
CRM	CRMlblAcctNum.Text property	The text property provides the value of the interaction key in the framework
CRM	IntMgr.CloseInteraction method	The Close Interaction method removes the any stored context values in memory based on the interaction key.

3. In the automation, connect the design blocks. The CRM_E_CRMChild_Closing automation should be similar to the following image .



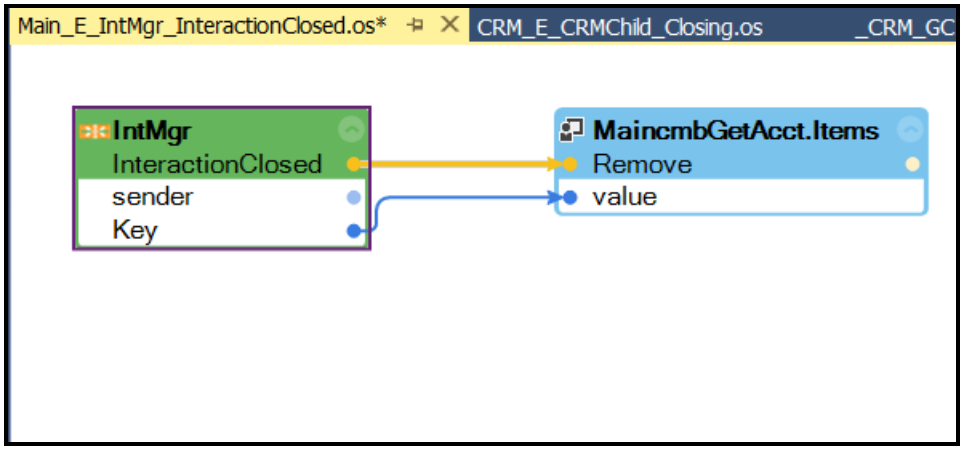
4. From the menu, select **File > Save All** to save your automation.

Create an automation to remove the account number from the combo box when the CRMChild window closes

1. In the Solution Explorer, add an automation, named **Main_E_IntMgr_InteractionClosed**, in the **Interactions** folder of the **Main-UI** project. The automation opens in a design window.
2. Add the following design blocks to the Main_E_IntMgr_InteractionClosed automation.

Project Item Source	Control	Description/Configuration
_Main_GC	IntMgr.InteractionClosed event	Click the down arrow in the design block to expose the interaction key parameter.
CustomerInformation	MaincmbGetAcct.Items.Remove method	Use the Explore Component properties button on the Object Explorer to access the Remove method.

3. In the automation, connect the design blocks. The following images shows the completed Main_E_IntMgr_InteractionClosed.



4. From the menu, click **File > Save All.**

5. Start the debugging.
   a. Click **User 2** on the CRM application. Both account numbers populate the combo box.
   b. In the CRM Application, close a customer window. The account number associated with the window disappears from the combo box.

Exercise: Moving context values from the Framework

Scenario

The business case states that the customer account information should be displayed on the Customer Information window after the account is accessed. You created automations to add and remove the account number from the combo box. Account information can be updated numerous times throughout the interaction.

Your assignment

For the solution, create a procedure automation, Main_P_LoadCustInfoValues, that moves customer account context values from the Framework to the user interface only if the interaction exists within the Framework. When updating the values, make the account number/interaction key the selected item in the combo box.

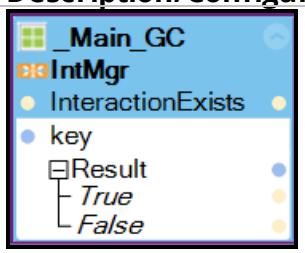
Detailed steps

Follow these steps to create a function automation to populate the Customer Information window. The function automation requires an entry parameter of the interaction key to check for the interaction. If it exists, populate the window.

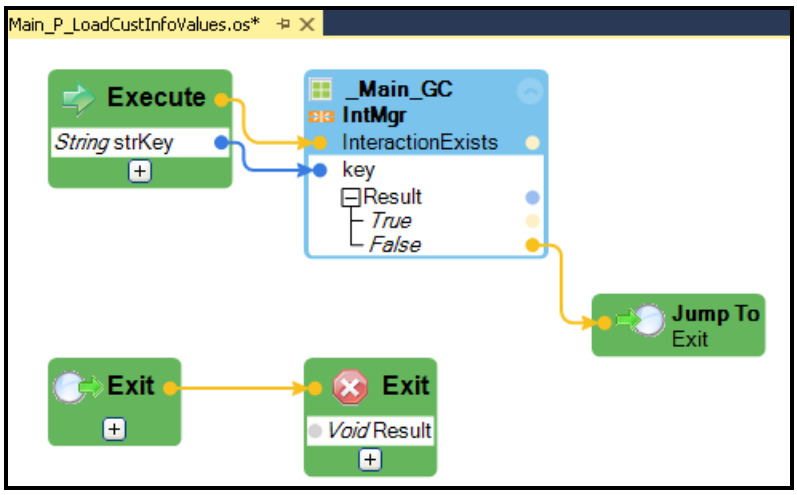
1. In the Solution Explorer, right-click the **Main-UI** project.
2. Select **Add > New Folder** and name it **Procedures**.
3. Right-click the **Procedures** folder and select **Add > New Automation**.
4. In the **Name** field, enter **Main_P_LoadCustInfoValues** and click **Add**. The automation opens in a design window.
5. In the automation, right-click and select **Add Entry Point**.
6. On the Execute design block, click the **+** icon to add a parameter.
7. Set the data type to **String** and the param1 name to **strKey** to edit the parameter.
8. In the automation, right-click and select **Add Exit Point**.
9. In the **Exit1** point, change the name to **Exit**.
10. Right-click an open area inside the Main_P_LoadCustInfoValues automation and select **Add Label**.
11. Rename the label **Exit**.

12. Right-click in the automation and select **Jump To > Exit**.

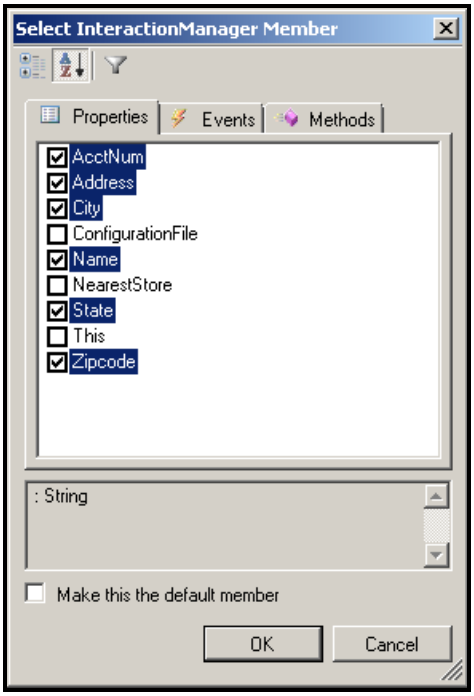
13. Add the design block to the Main_P_LoadCustInfoValues automation.

Project Item Source	Control	Description/Configuration
_Main_GC	IntMgr.InteractionExists method	

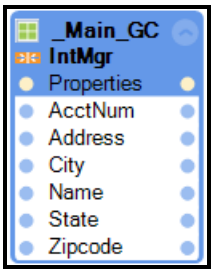
14. Organize and connect the design blocks as shown in the following image.



15. In the Object Hierarchy, click and drag the **IntMgr** object to the automation. The Select Member window is displayed.
16. On the **Select Member** window, select the following properties of the IntMgr:
    - AcctNum
    - Address
    - City
    - Name
    - State
    - Zipcode

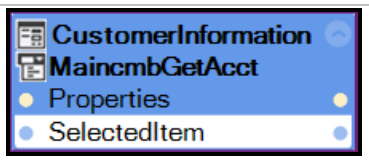


17. Click **OK**. The Select Member window closes and the design block is displayed.

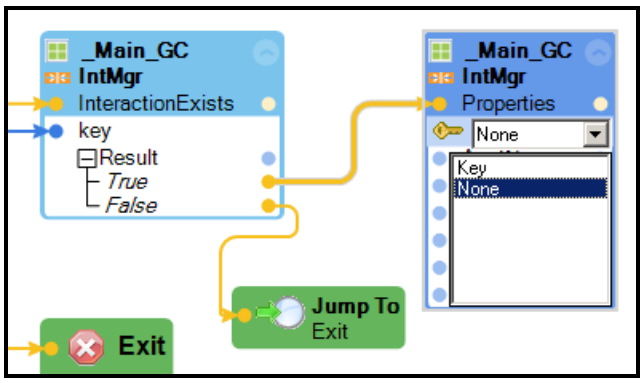


18. In the Object Hierarchy, click and drag each of the following controls to the automation. The Text property design block of each control is displayed.
    - MaintxtCurrentAcct
    - MaintxtAddress
    - MaintxtCity
    - MaintxtName
    - MaintxtState
    - MaintxtZip
19. Add the design blocks to the Main_P_LoadCustInfoValues automation.

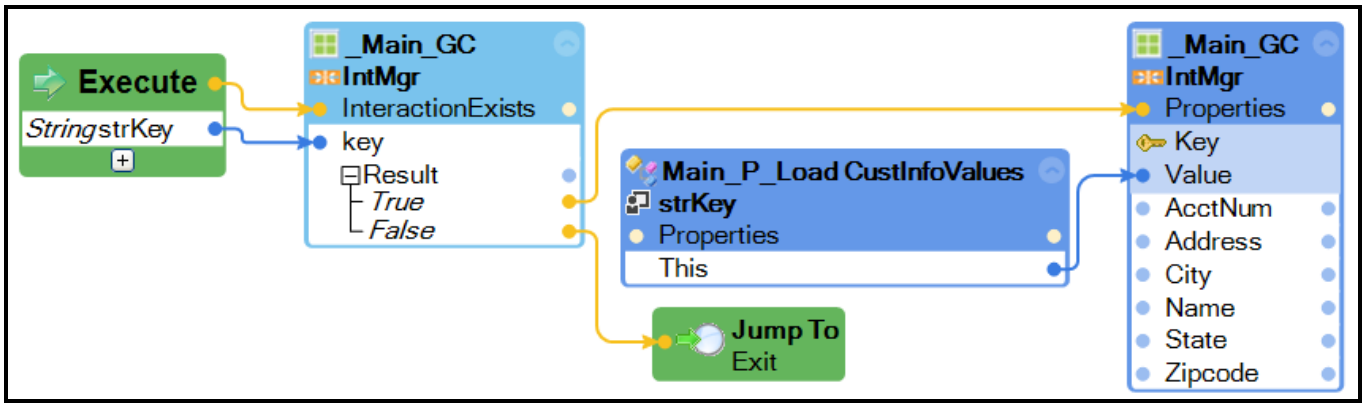
Project Item Source	Control	Description/Configuration
Customer Information	MaincmbGetAcct.SelectedItem property	In the Object Inspector, use **Configure Type** to locate the property.

		
Main_P_ LoadCustInfoValues	strKey.This property	Create a proxy for the strKey input parameter

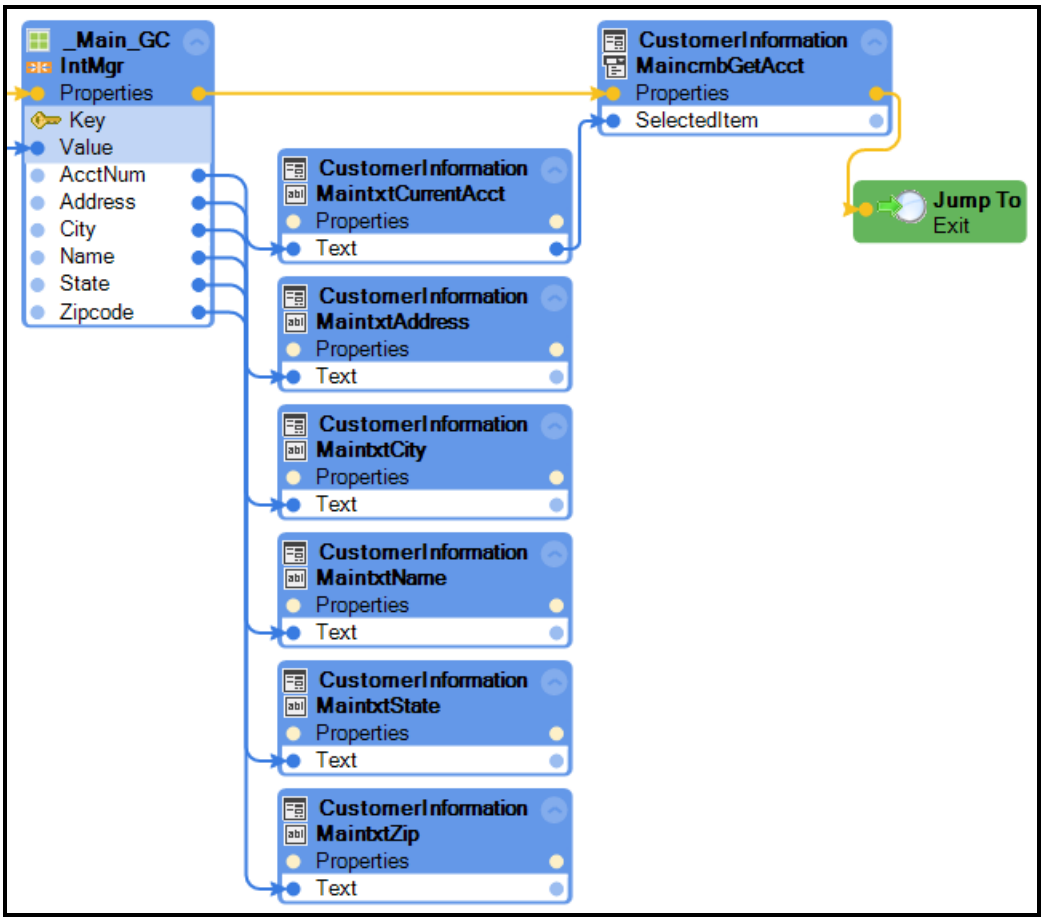
20. In the automation, right-click to add another **Jump To Exit**.
21. In the automation, connect the IntMgr.InteractionExists True output to the IntMgr.Properties input. A key icon is displayed on the properties design block.
22. On the design block, click the **Key** icon. A drop-down box is displayed.



23. Select **Key**. A value parameter is displayed on the design block.
24. From the strKey.This design block, connect to the IntMgr.Key value input.



25. Connect the links as shown in the following image.



26. From the menu, select **File > Save All**.

Interacting between applications and projects

Exercise: Adding Activities to a project

Scenario

The business case states that the customer account information should be displayed on the Customer Information window after the account window opens. You created automations to add and remove the account number from the combo box. Each time a new account window is displayed in the CRM application, the user interface updates with the new account information. Because the application is in the CRMAdapter project and the user interface (UI) is in the Main-UI projects, start activities in the CRMAdapter project to update the UI in the Main-UI project.

Your assignment

For the solution, add the activity components and add an automation to start the activities.

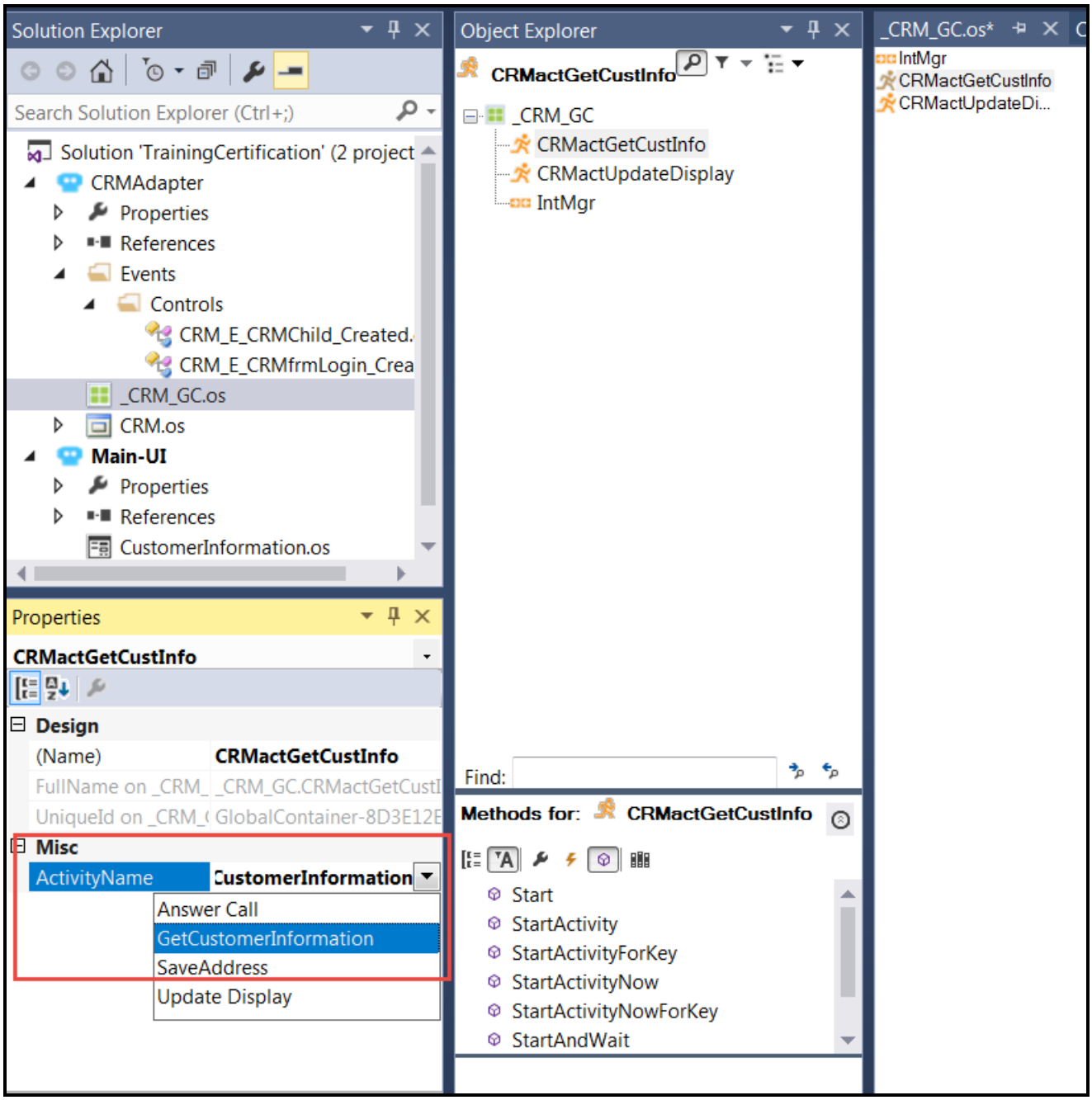
Detailed steps

Add activity components

Follow these steps to add the activity components.

1. In the Solution Explorer, double-click **_CRM_GC.os** to open it in a designer window.
2. In the Toolbox, expand the Interaction Management section.
3. Click and drag two activity components to the global container.
4. In **_CRM_GC.os**, highlight **activity1**.
5. In the Properties window, rename activity1 to **CRMactGetCustInfo**.

6. In the **Misc > Activity Name** property drop-down, select **GetCustomerInformation**.

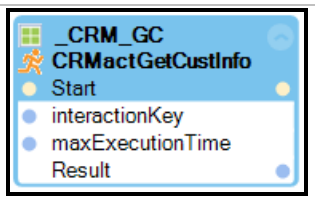
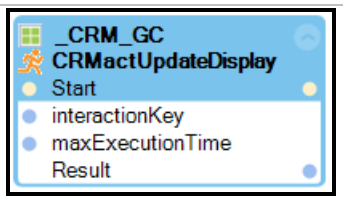


7. In the **_CRM_GC.os**, highlight **activity2**.
8. In the Properties window, rename activity2 to **CRMactUpdateDisplay**.
9. In the **Misc > Activity Name** property drop -own, select **Update Display**.
10. From the menu, select **File > Save All**.

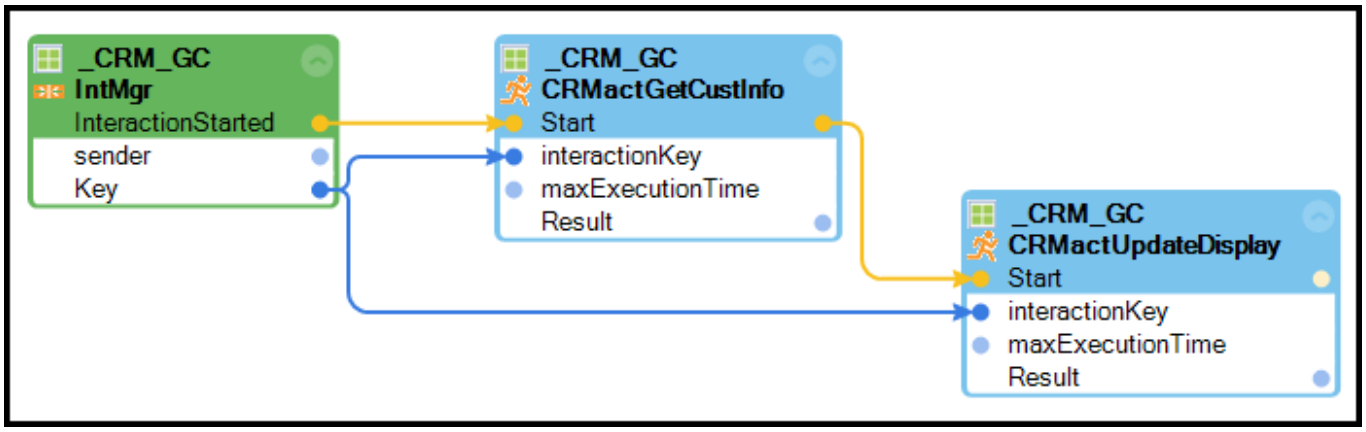
Add automation to start the activities

Follow these steps to add an automation to start the activities.

1. In the Solution Explorer and within the CRMAdapter project, highlight the Events folder.
2. Right-click and select **New Folder**.
3. In the Solution Explorer, rename the folder to **Interactions**. The new renamed folder is displayed under the Events folder in CRMAdapter project.
4. In the CRMAdapter project, in the Interactions folder, create an automation named **CRM_E_IntMgr_InteractionStarted**. The automation opens in a design window.
5. Add the following design blocks to the CRM_E_IntMgr_InterationsStarted automation.

Project Item Source	Control	Description/Configuration
_CRM_GC	IntMgr.InteractionStarted event	Click the down arrow in the design block to expose the interaction key parameter.
_CRM_GC	CRMactGetCustInfo.Start method	
_CRM_GC	CRMactUpdateDisplay.Start method	

6. Connect the design blocks as shown in the following image.



7. From the menu, select **File > Save All** to save your work.

Exercise: Storing context values in the framework

Scenario

The business case requires that the CRM customer account information and the nearest store address display on the Customer Information window by using the Interaction Framework. For data integrity, do not store data into the framework if the customer information is not available in the application.

Your assignment

For this solution, create an automation to perform the following:

- Update and organize the project with folders.
- Validate the matching of the interaction key in the framework with the customer account information in the CRM Application.
- If a match exists, store the account information in the framework; if a match does not exist, display a message to the end user.
- In the controller project, add an automation that calls Main_P_LoadCustInfoValues when the Framework has started the activity UpdateDisplay

Detailed steps

Add automation logic to validate the interaction and the account number match

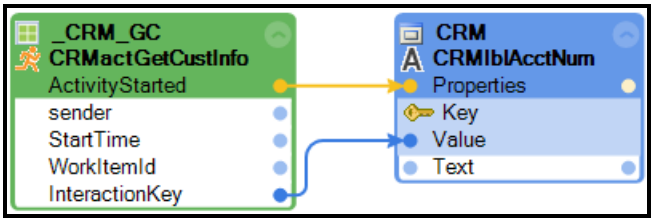
Follow these steps to modify the interaction-call.xml.

1. In Solution Explorer, right click the **Events folder** in the CRMAdapter and select **Add > New Folder**.
2. In the Solution Explorer, name the folder **Activities**.
3. Right-click the Activities folder and select **Add > New Automation**. The Add New Item window displays.
4. In the Add New Item window, name the automation **CRM_E_GetCustInfo_ActivityStarted**.
5. Click **Add**. The automation opens in a design window.
6. Add these design blocks to the automation.

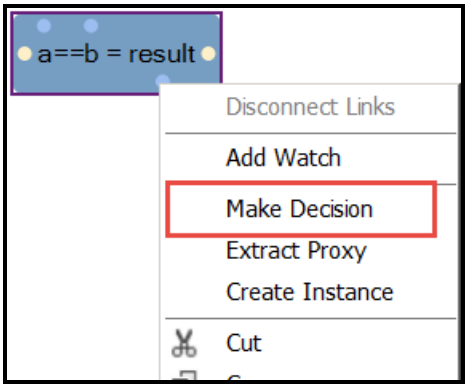
Project Item Source	Control	Description/Configuration
_CRM_GC	CRMactGetCustInfo.ActivityStarted	Click the down arrow in the design

	event	block to expose the interaction key parameter.
CRM	CRMlblAcctNum.Text property	Use the account number to compare to the key value.

7. Connect the automation links as shown in the following image. After connecting to the CRMlblAcctNum, the Key icon displays to identify the change from None to Key.

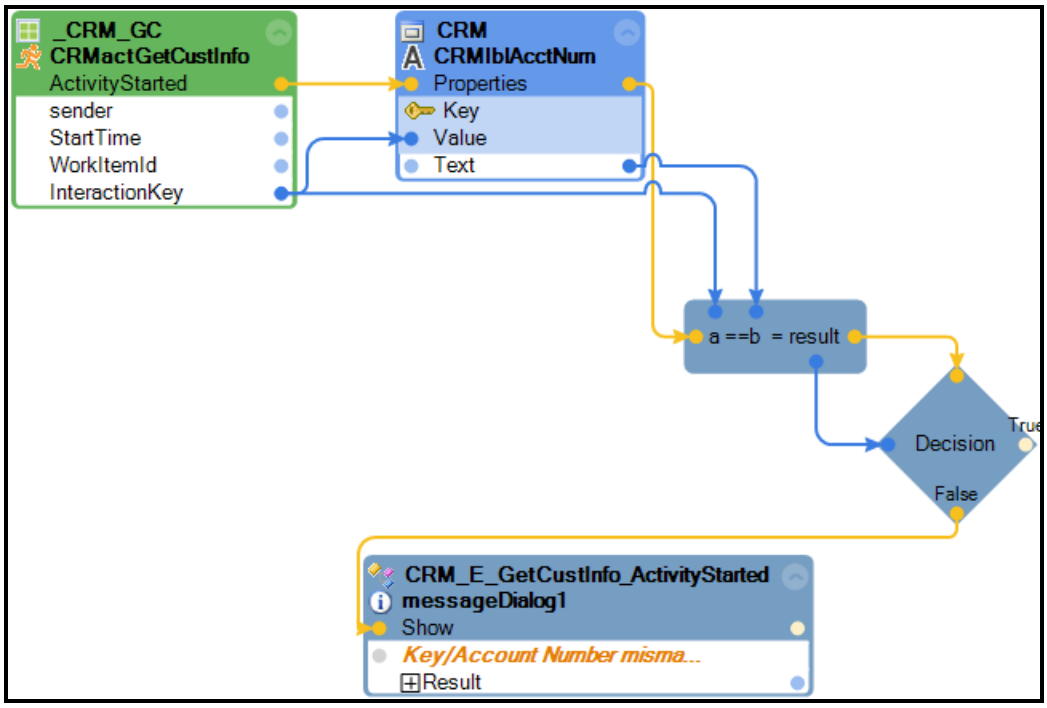


8. From the Toolbox, expand the Comparisons and Expression section.
9. Click and drag a Boolean Expression component to the automation. The design block displays and the CRM_component appears on the Local tab at the bottom of the automation.
10. In the Properties window, configure the Boolean Expression as follows:
    a. In Expression property, enter **a = = b**.
    b. In the Identifiers property, click **ellipses button** and change both datatypes from Double to String.
11. On the Boolean Expression in the automation, right-click on the **result blue data port** and select **Make Decision**.



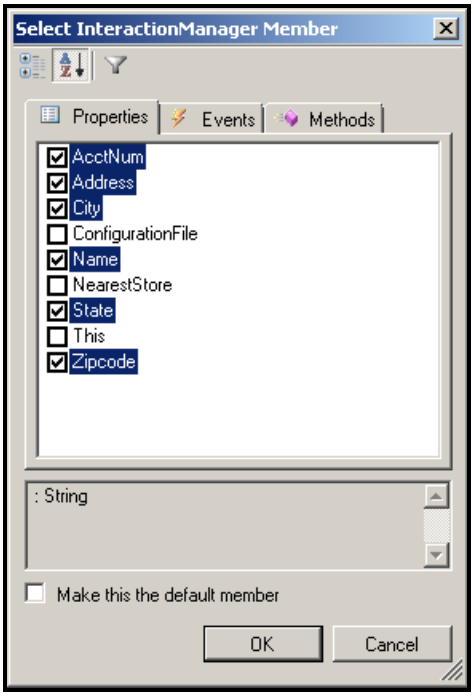
12. From the Toolbox, expand the Advanced section.
13. Click and drag a MessageDialog component to the automation. A Choose Method Overload window displays to select which type of message window to use in the automation.
14. Select **(String message): Dialog Result**. The string message allows you to manually enter or pass a string to display a message to the end user.
15. Click **OK**. The MessageDialog displays in the automation and on the Local tab at the bottom.
16. On the messageDialog1 design block, click **text Message** and enter **Key/Account Number mismatch**.

17. Connect the automation links as shown in the image below.

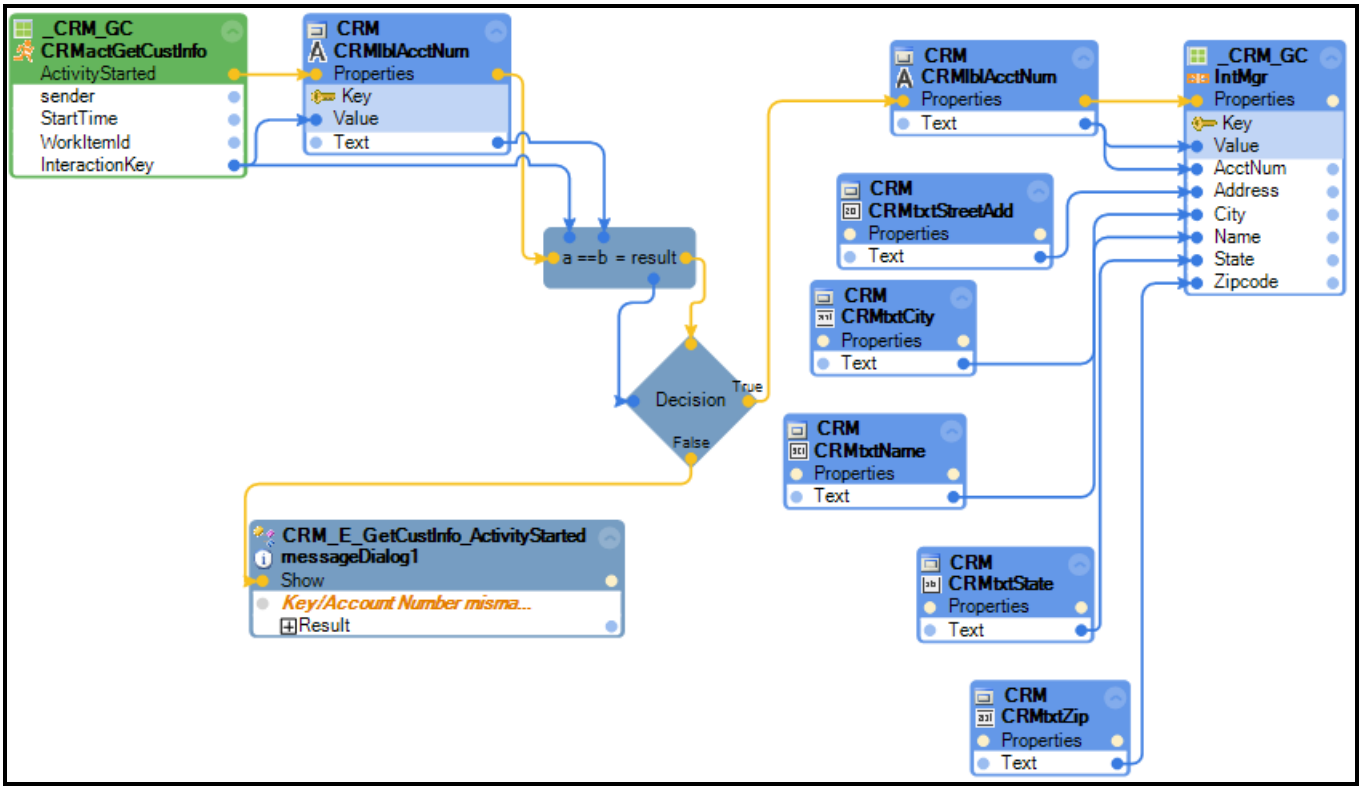


Add automation logic to store data to the framework

1. In the Object Hierarchy, click and drag each of the following controls to the automation. The Text property design block of each control displays:
   - CRMlblAcctNum
   - CRMtxtName
   - CRMtxtStreetAdd
   - CRMtxtCity
   - CRMtxtState
   - CRMtxtZip
2. In the Object Hierarchy, click and drag the **IntMgr** object to the automation. The Select Member window displays.
3. On the Select Member window, select the following properties of the IntMgr:
   - AcctNum
   - Address
   - City
   - Name
   - State
   - Zipcode



4. Click **OK**. The Select Member window closes and the design block displays.
5. Connect the automation links as shown in the following image . After connecting to the IntMgr properties design block, the Key icon displays to show the change from None to Key.

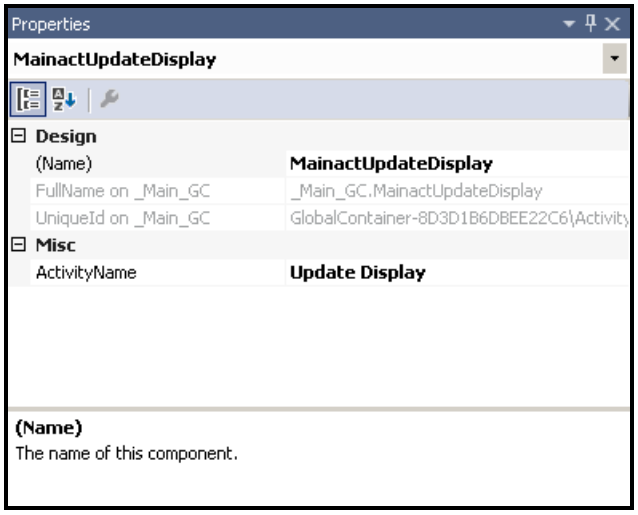


6. Select **File > Save All** to save your work.

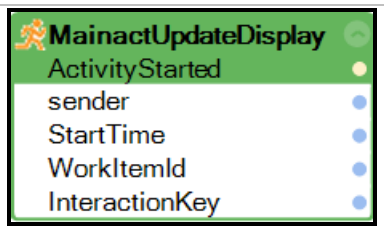
Create an automation to update the user interface

Follow the steps to create an automation to call the Main_P_LoadCustInfoValues automation and update the user interface.

1. In Solution Explorer, double click the **_Main_GC.os** to open it in a designer window.
2. In the Toolbox, expand the **Interaction Management** section.
3. Click and drag an activity component to the global container.
4. In the __Main_GC.os, highlight **activity1**.
5. In the **Properties** window, rename it to **MainactUpdateDisplay**.
6. In the **Misc > Activity Name** property drop down, select **UpdateDisplay**. Adding this activity allows the Framework to respond to the starting of the activity in the CRMAdapter project.

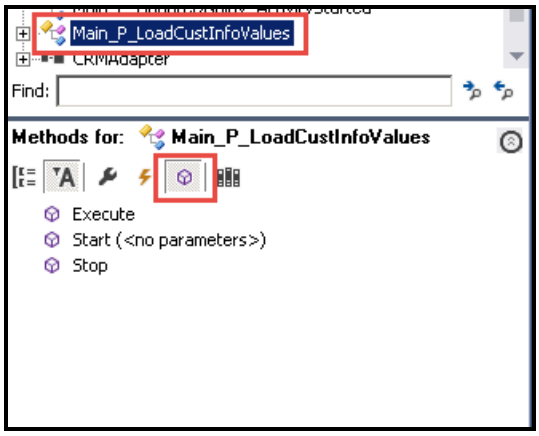


7. Select **File > Save All**.
8. In the Solution Explorer in the Main-UI project, add a new folder under the Events folder and name it **Activities**.
9. Right-click on the **Activities** folder and add a new automation.
10. Name the automation, **Main_E_UpdateDisplay_ActivityStarted**. The automation opens in a design window.
11. Add the design block to the Main_E_UpdateDisplay_ActivityStarted automation.

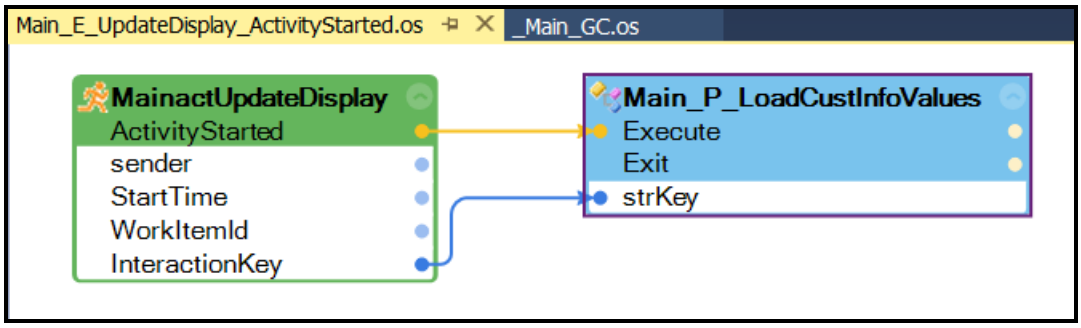
Project Item Source	Control	**Description/Configuration**
_Main_GC	MainActUpdateDisplay.ActivityStarted event	

12. From the Object Hierarchy, select the **Main_P_LoadCustInfoValues** automation to highlight it.

13. In the Object Inspector, click the **Show Methods Only** (box) button.



14. Click and drag the **Execute** method to the automation.
15. Connect the design blocks as shown in the following image. .



16. Select **File > Save Al**l to save your work.
17. Test the solution. The Customer Information window populates with the default values from the XML file because you have not created the automation to store the values from the CRM application into the Framework.

Exercise: Adding an activity to the XML

Scenario

In reviewing the case, users can interact with either the CRM application or the UI. This scenario focuses on interacting with the UI. When users select an account number in the combo box, the automation should check to see if the value is currently active. If the value is active, then nothing. If the value is not active, then activate the CRMChild window and update the UI with the selected account information. The Framework requires an activity to alert the CRMAdapter project about a change in the account number in the UI combo box. The interaction-call.xml does not contain the necessary activity to complete this requirement.

Your assignment

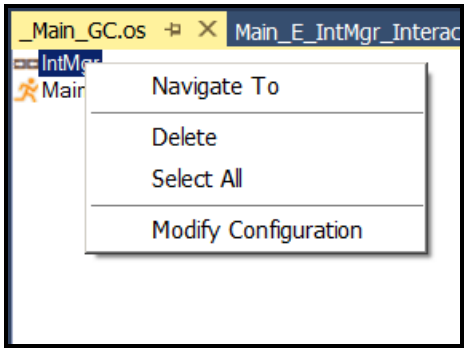
In this part of the solution, add 2 activities to the XML file.

- ActivateMDIWindow with a string value named AcctNumber
- GetNearestStore with two values: a string named strResult and a Boolean named isSuccess.

Detailed steps

Follow these steps to add an activity to the interaction-call.xml.

1. Open the _Main_GC project item to display it in a design window.
2. In the _Main_GC window, right-click the **IntMgr** and select **Modify Configuration**. The interaction-call.xml opens in a design window.



3. Locate the Activities section in the interaction-call.xml file.
4. After the last activity item in the Activities section, enter the following:
   - **<Activity Name="ActivateMDIWindow">**
   - **<Value Name="AcctNumber" Type="String" />**
   - **</Activity>**
   - **<Activity Name="GetNearestStore">**
   - **<Value Name="strResult" Type="String" />**
   - **<Value Name="isSuccess" Type="Boolean" />**
   - **</Activity>**
5. Select **File > Save All** to save the edits.
6. From the menu, select **Build > Clean Solution**. The Output window opens to display the Studio messages for the clean.
7. From the menu, select **Build > Rebuild Solution**. The Output window opens to display the Studio messages for the rebuild.

Exercise: Activating an interaction

Scenario

In reviewing the case, users can interact with either the CRM application or the UI. This scenario focuses on interacting with the CRM application. When users have more than one window open or more than one interaction within the framework, users may switch windows in the CRM application by clicking on the inactive window. When activating a CRM window manually, the UI account information should update to display the new active interaction context values.

Your assignment

In this part of the solution, complete the following:

- Create an automation, CRM_E_CRMChild_Activated, that activates an interaction when the window activates.
- Create an automation, Main_E_IntMgr_InteractionActivated, that updates the UI when an interaction activates.

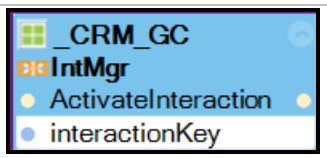
Detailed steps

Follow these steps to start an interaction and update the UI.

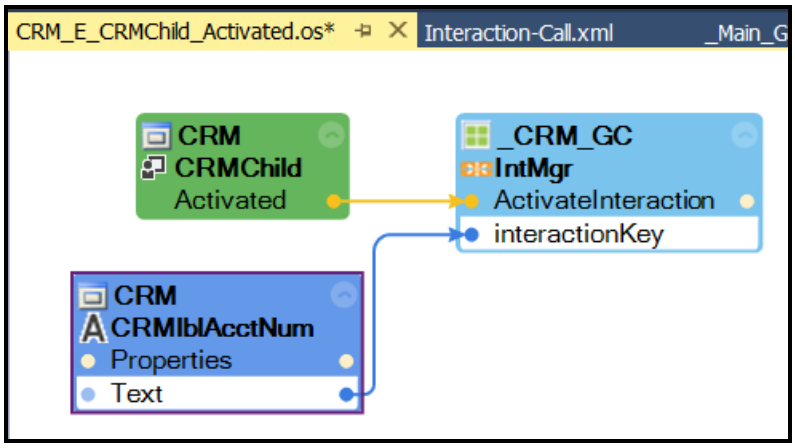
Create an automation that activates an interaction when a window activates

1. Add a new automation to the **CRMAdapter** project.
   - Add the automation in **Controls** folder.
   - Name the new automation **CRM_E_CRMChild_Activated**.
2. Add the following design blocks to the new automation. If a property, event, or method does not exist for a control in the Object Inspector, click the **Configure Type** icon to select more options to display.

Project Item Source	Control	Description/Configuration
CRM	CRMChild.Activated event	The automation begins when the window activates. 
CRM	CRMlblAcctNum.Text property	The text property of the activated window contains the interaction key needed to activate the correct interaction

		in the Framework.
_CRM_GC	IntMgr.ActivateInteraction method	

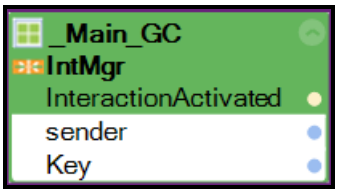
3. Connect the design blocks as shown in the following image.

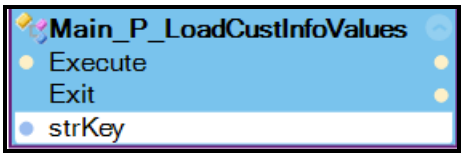


4. Select **File > Save Al**l to save the automation.

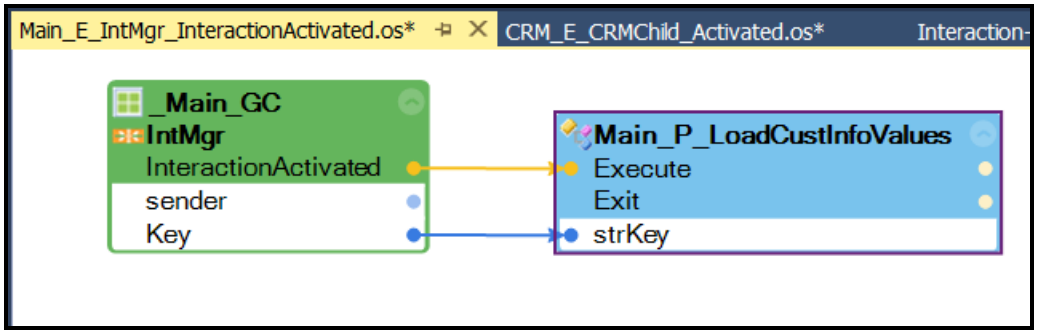
Create an automation that updates the UI when an interaction activates

1. Add a new automation to the **Main-UI** project.
   - Add the new automation to the **Interactions** folder.
   - Name the new automation **Main_E_IntMgr_InteractionActivated**.
2. Add the following design blocks to the new automation. If a property, event, or method does not exist for a control in the Object Inspector, click the **Configure Type** icon to select more options to display.

Project Item Source	Control	Description/Configuration
_Main_GC	IntMgr.InteractionActivated event	The automation begins when an interaction activates. Click the down arrow to expose the interaction key. 
Main_P_	Main_P_	Because a procedure automation already

LoadCustInfoValues automation	LoadCustInfoValues.Execute method	exists to update the UI, call that automation. 

3. Connect the design blocks as shown in the following image.



4. Select **File > Save All** to save the automation.
5. Test the solution.
   - John Smith's information displays on the Customer Information window.
   - In the CRM application, click User 2. The second account updates on the Customer Information window.
   - In the CRM application, click the window with John Smith to activate the window. The Customer Information window updates with John Smith's account.
   - In the CRM application, close a window. The Customer Information window updates with the last active window and removes the closed account number from the combo box.

Exercise: Activating an activity

Scenario

In reviewing the case, users can interact with either the CRM application or the UI. This scenario focuses on interacting with the UI. When users have more than one window open or more than one interaction within the framework, users may switch the windows by selecting an account in the combo box. When users select an account number, the automation should check to see if the value is currently active. If the value is active, then the automation does nothing. If the value is not active, then the automation activates the CRMChild using the ActivateMDIWin activity.

Your assignment

In this part of the solution, complete the following:

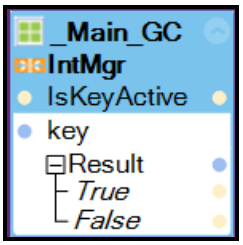
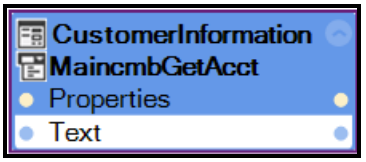
- Create an automation, Main_E_cmbGetAcct_SelectedIndexChanged, that starts an activity when the selected account number is not active.
- Create an automation, CRM_E_ActivateMDIWin_ActivityStarted, that activates the window when the activity starts.

Detailed steps

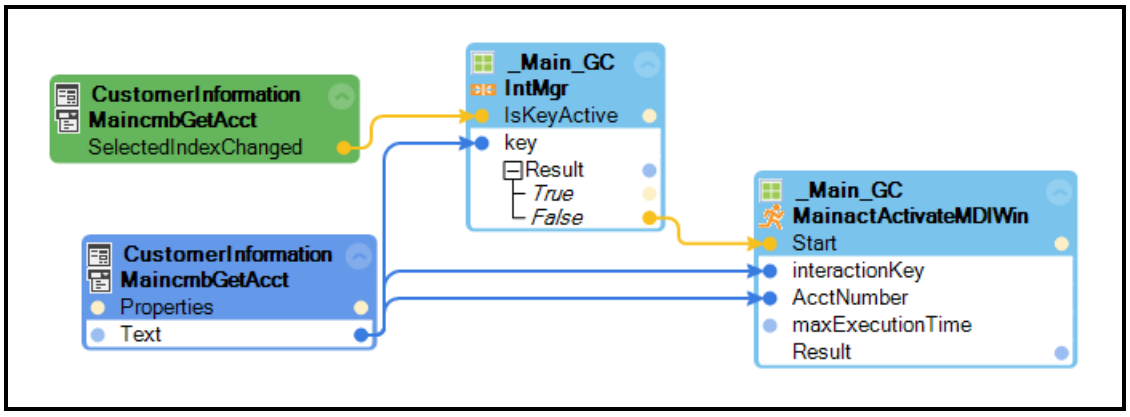
Follow these steps to start an activity and activate a window from a combo box selection.

Create an automation that starts an activity when the account number selected is not active

1. Add an activity to the _Main_GC.
   - Add an activity from **Toolbox > Interaction Management**.
   - Name the activity **MainactActivateMDIWin**
   - Select **ActivateMDIWin** from Activity Name property
2. Add a new folder to the **Main-UI** project.
   - Add a subfolder to the **Events** folder.
   - Name the subfolder **Controls**.
3. Add a new automation to the **Main-UI** project.
   - Add the new automation to the **Controls** folder.
   - Name the new automation **Main_E_cmbGetAcct_SelectedIndexChanged**.
4. Add the following design blocks to the new automation. If a property, event, or method does not exist for a control in the Object Inspector, click the **Configure Type** icon to select more options to display.

Project Item Source	Control	Description/Configuration
Customer Information	MaincmbGetAcct.SelectedIndexChanged event	The automation initiates when the combo box index changes. The SelectedIndexChanged is under the Behavior category. 
_Main_GC	IntMgr.IsKeyActive method	Checks to determine if the interaction key is active. Expand the Result. 
Customer Information	MaincmbGetAcct.Text property	Text provides the value to activate the CRM window. 
_Main_GC	MainactActivateMDIWin.Start method	

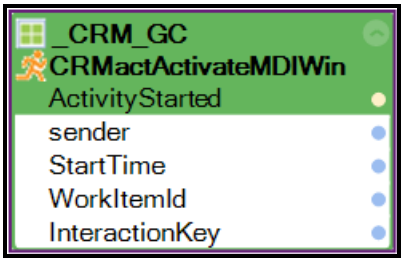
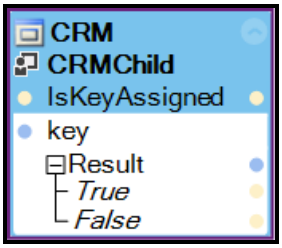
5. Connect the design blocks as shown in the following image.



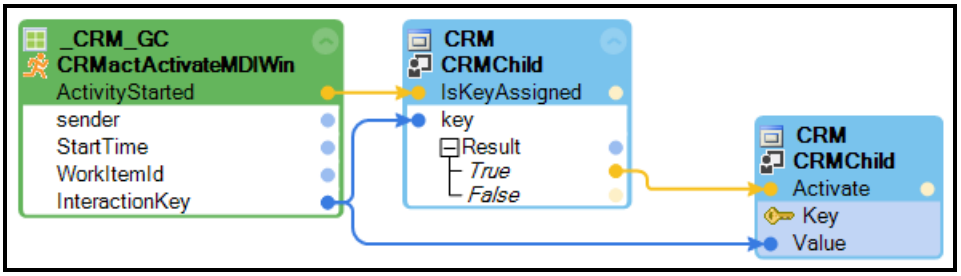
6. Select **File > Save All** to save the automation.

Create an automation that activates the window when the activity starts

1. Add an activity to the _CRM_GC.
   - Add the activity from **Toolbox > Interaction Management**.
   - Name the activity **CRMactActivateMDIWin**.
   - Select **ActivateMDIWin** from Activity Name property.
2. Add a new automation to the **CRMAdapter** project.
   - Add automation to the **Activities** folder.
   - Name the new automation **CRM_E_ActivateMDIWin_ActivityStarted**.
3. Add the following design blocks to the new automation. If a property, event, or method does not exist for a control in the Object Inspector, click the **Configure Type** icon to select more options to display.

Project Item Source	Control	Description/Configuration
_CRM_GC	CRMactActivateMDIWin.ActivityStarted event	The automation initiates when the activity starts. Click the down arrow to expose the interaction key. 
CRM	CRMChild.IsKeyAssigned method	Checks to determine if a CRM window exists. Expand the Result. 
CRM	CRMChild.Activate method (no parameters)	

4. Connect the design blocks as shown in the following image. The Key icon displays once you connect the automation links.



5. Select **File > Save All** to save the automation.
6. Test the solution.

- John Smith's information displays on the Customer Information window.
- In the CRM application, click User 2. The second account updates on the Customer Information window.
- On the Customer Information window from the **Get Account** combo box, select **23453** . The Customer Information updates with John Smith's account. In the CRM application, the window for John Smith activates.

Working with Interaction Framework

Exercise: Updating a project for the Framework

Scenario

The business case states that the search for the nearest store should begin when users click a button on the Customer Information window. The search completes and the nearest store address is displayed in the Customer Information window. Since the existing projects use the Interaction Framework, the TrainingWebAdapter requires the Interaction Framework as well. An activity starts on the button click, and the search begins in the ACMESearchSystem. The ACMESearchSystem runs only when the button clicks, so an automation determines the running of the system and the location of the user in the system. The activity requires a message and a Boolean result to return to the Main-UI so that a proper message displays to users when the search completes.

Assignment

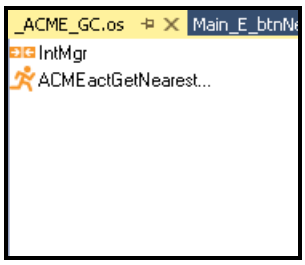
You created several automations earlier without the concept of Interaction Framework in the Training WebAdapter project. Refractor the project and automations to now use the Framework methodology of interactions, activities, and context values.

- Add a new activity to the _Main_GC and _ACME_GC to get the nearest store.
- Create an automation, Main_E_btnNearestStore_Click, to start the GetNearestStore activity.
- Create an automation, CRM_E_GetNearestStore_ActivityStarted, to complete the search for the nearest store based on account zip code.

Detailed Steps

Update projects for new activity and Framework

1. In the _Main_GC, add a new activity:
   - Rename the activity's Design Name to **MainactGetNearestStore**.
   - In the ActivityName, select **GetNearestStore**.
2. In the _ACME_GC, add a new activity:
   - In the Design name for the activity, enter **ACMEactGetNearestStore**.
   - In the ActivityName, select **GetNearestStore**.



3. From the menu, select **File > Save All**.

Create an automation to start an activity on a button click

1. In the Controls folder of the Main-UI project, add an automation.
2. In the Add Item window, name the automation **Main_E_btnNearestStore_Click.**
3. Add the design blocks shown in the following table to the automation.

Source project Items	Design Name	Description
Customer Information	MainbtnNearestStore.Click event	When the user clicks the button, the automation initiates.
Customer Information	MaincmbGetAcct.Text property	The text property of the combo box ensure the active interaction for the activity.
_Main_GC	MainactGetNearestStore.Start method	This starts the activity on the button click.

4. Connect the automation and data links as shown in the following image.

5. From the menu, select **File > Save All**.

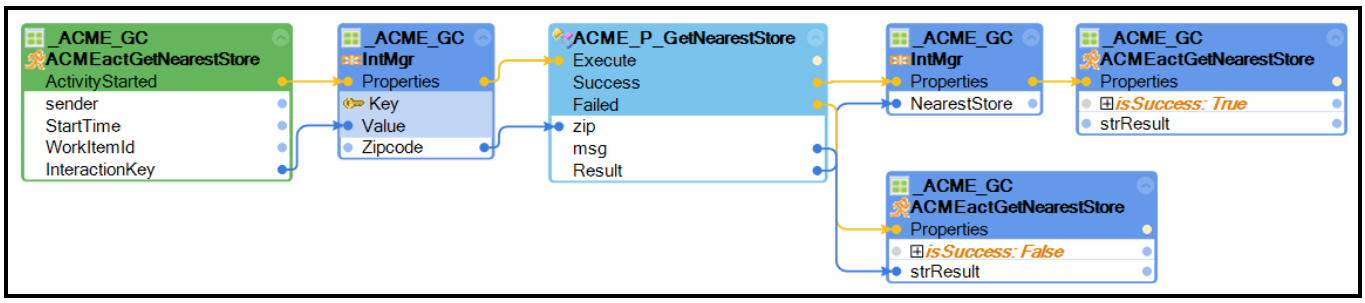
Create automations to complete the activity and to validate the ACMESearchSystem

1. In the Solution Explorer, add two folders to the TrainingWebAdapter.
   - Name the first folder **Events**.
   - Under the Events folder, name the subfolder **Activities**.
2. In the new **Activities** folder, add a new automation.
3. Name the automation **ACME_E_GetNearestStore_ActivityStarted**. The automation opens in a Design window.
4. Add the design blocks shown in the following table to the automation.

Source project Items	Design Name	Description
_ACME_GC	ACMEactGetNearestStore.ActivityStarted event	Click the down arrow in the design block to expose the interactionkey

		parameter.
_ACME_GC	IntMgr.ZipCode property	This collects the zip code from the Framework to pass through the search automations.
ACME_P_ GetNearestStore	Execute method	This calls the logic to perform the search of the nearest store based on account zip code.
_ACME_GC	IntMgr.Nearest property	This is used to store the address in the Framework context value.

5. From the Object Explorer, click and drag the **ACMEactGetNearestStore** activity component to the automation. The Select Activity Member window is displayed.
6. In the Select Activity Member window, select **isSuccess** and **strResult**, and click **OK**.
7. Configure the two design blocks as follows:
   - For one design block on the IsSuccess property, select **True**. On the other design block, select **False**.
8. Connect the automation and data links as shown in the following image .



9. From the menu, select **File > Save All.**

Verify your work

1. Run the debugger.
2. On the Customer Information window, click **Find Store**.
3. The web application launches, logs in, and searches for the nearest store based on the zip code.
4. The Customer Information window does not return the address as expected.

Exercise: Returning new values

Scenario

The business case states that the search for the nearest store should begin when users click a button on the Customer Information window. The search completes and returns the nearest store address and displays the information on the Customer Information window. If the search fails, the system

displays a message informing user. The GetNearestStore activity contains the two parameters to use for the failed activity message.

Your assignment

At this point in the solution, complete the following:

- Create an automation that displays a message to the user if the nearest store activity fails.
- Modify the Main_P_LoadCustInfoValues to have the nearest store value display during simultaneous interactions.
- Create an automation to update the user interface when the context value changes.

Detailed steps

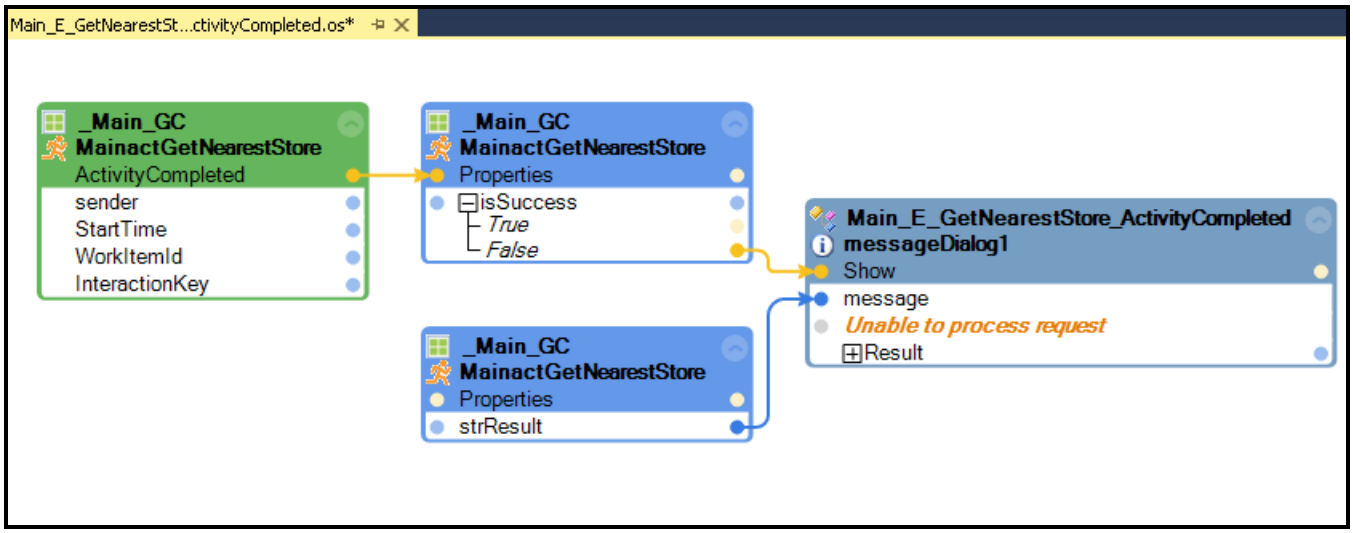
Create an automation to display a fail message to the user

1. In the Main-UI project, create an automation.
   - Add the automation to the Main-UI **Activities** folder.
   - Name the automation **Main_E_actGetNearestStore_ActivityCompleted**.
2. Add the following components from the Toolbox and modify their properties as shown in the following table. .

Source project Items	Design Name	Description
_Main_GC	MainactGetNearestStore.ActivityCompleted event	When the activity thread completes, initiate this automation
_Main_GC	MainactGetNearestStore.isSuccess property	
_Main_GC	MainactGetNearestStore.strResult property	

3. From the Advanced section of the Toolbox, add a **MessageDialog** component. The Overflow window displays.
4. On the Overflow window, select the third radio button to display a string message and a caption.
5. In the messageDialog1 design block, click **Caption** and enter "**Unable to process request**".

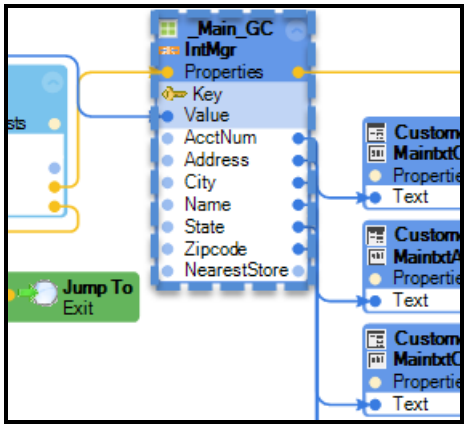
6. Connect the automation and data links as shown in the following image. .



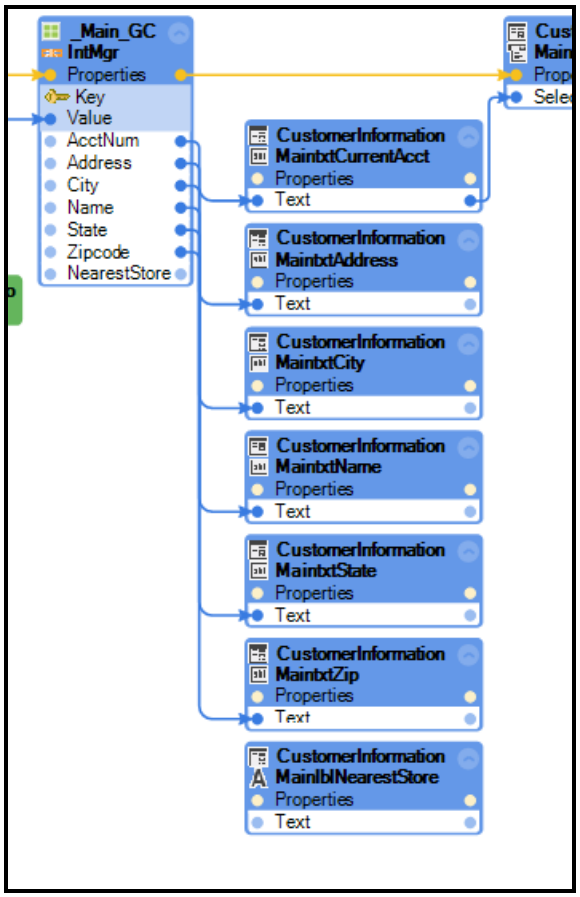
7. From the menu, select **File > Save All**.

Modify the Main_P_LoadCustInfoValues to display the nearest store

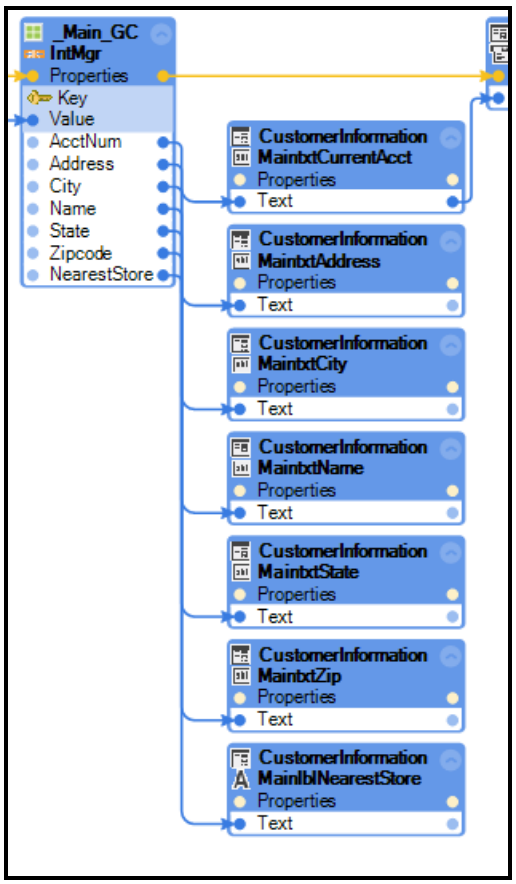
1. Open the Main_P_LoadCustInfoValues automation to display in a design window.
2. In the Object Hierarchy, click on **IntMgr**.
3. In the Object Inspector, click **Show Properties Only**.
4. Click and drag the **NearestStore** property to the _Main_GC.IntMgr design block in the automation. A dotted black line surrounds the_Main_GC_IntMgr design block.



5. Add the **CustomerInformation.MainlblNearestStore.Text** property to the automation.



6. Connect the _Main_GC.IntMgr.NearestStore data output to the CustomerInformation.MainlblNearestStore.Text data input.



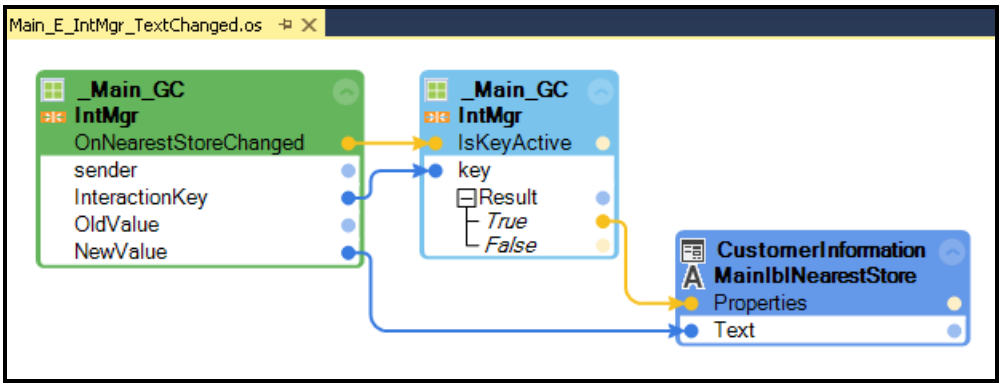
7. From the menu, select **File > Save All**.

Update user interface when a context value changes

1. In the Main-UI project, create an automation.
   - Add the automation to the Main-UI **Interactions** folder.
   - Name the automation **Main_E_IntMgr_TextChanged**.
2. Add the following components from the Toolbox and modify their properties as shown in the following table.

Source project Items	Design Name	Description
_Main_GC	IntMgr.OnNearestStoreChanged event	When the value of the nearest store changes, initiate this automation.
_Main_GC	IntMgr.IsKeyActive method	If value changes is from the active key, display the value immediately. If it not, it displays from the Main_P_LoadCustInfoValues automation.
Customer Information	MainlblNearestStore.Text property	

3. Connect the automation and data links as shown in the following image. .



4. From the menu, select **File > Save All**.
5. Debug the solution. The nearest store now displays on the Customer Information window.

DEPLOYMENT

This lesson group includes the following lessons:

- Solution deployment

Solution deployment

Exercise: Deploying a solution

Scenario

Now with the TrainingCertification solution complete, you can deploy the solution to the pilot group for further testing. Specific properties need configuring before you can create the deployment package. Because the Main-UI serves as the controller project for this solution, only that project requires modification.

Your Assignment

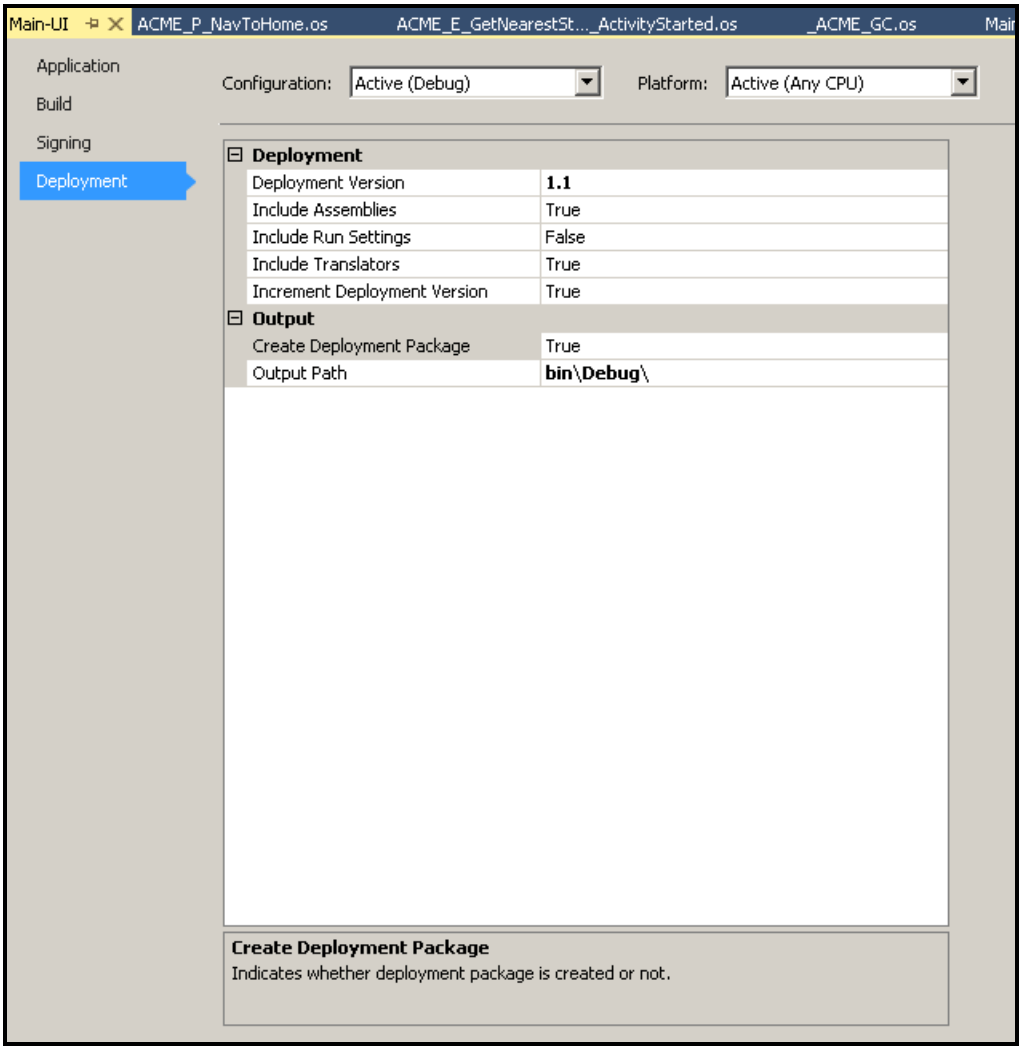
For this exercise, complete the following on the Project Properties Deployment tab for the Main-UI project:

Property	Value
Deployment Version	1.0
Include Assemblies	True
Include Run Settings	False
Include Translators	True
Increment Deployment Version	True
Create Deployment Package	True
Output Path	bin\Deploy\

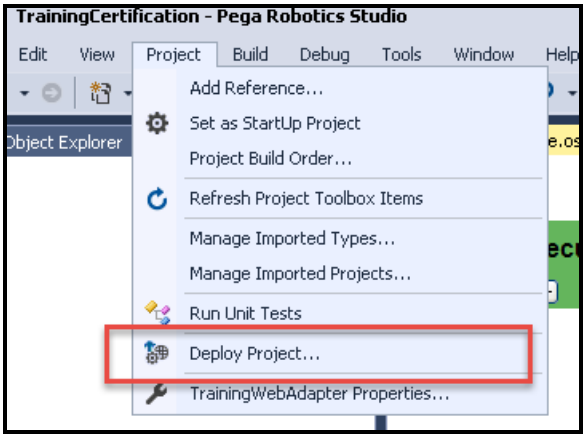
Detailed Steps

Follow these steps to deployment the TrainingCertification solution.

1. From the Solution Explorer, right-click the **Main-UI** project to open the Properties window.
2. On the Deployment tab on the Main-UI properties window, modify the properties as described in the Assignment section.



3. Select **File > Save All**.
4. From the menu, select **Project > Deploy** Project.



5. Click **OK** to begin the deployment process.
6. Click **OK** to close the status window.