

OpenSpan Studio Microsoft Office Connectors

TABLE OF CONTENTS

Confidentiality	2
Change History	2
HOW TO USE THE OPENSAN STUDIO MICROSOFT OFFICE CONNECTORS.....	3
Prerequisites.....	3
Project Outline.....	3
Microsoft Office Connectors.....	4
Microsoft Office Connectors.....	4
Installation Configuration	5
Exercise 1 – Log In Process.....	6
Exercise 2 – New Call Window and Export	7
Connector Details	16
Microsoft Excel Connector	16
Microsoft Outlook Appointment Connector	20
Microsoft Outlook Contact.....	24
Microsoft Outlook Notes	29

Confidentiality

This document contains information that OpenSpan, Inc regards as confidential. This information is provided with the understanding that it will not be disclosed to other parties without the prior written consent of OpenSpan.

Change History

Date	Amendment	Name
11/07/2007	Create draft	T. Ricks
2/20/2008	Revised.	T. Ricks
6/06/2008	Revised.	T. Ricks
8/29/2008	Revised	A. Ridgway
7/6/2009	Revised for 4.1	A. Ridgway

HOW TO USE THE OPENSPAN STUDIO MICROSOFT OFFICE CONNECTORS

This tutorial shows you how to use the Microsoft Office Connectors; including Microsoft Word, Excel and Outlook. In addition, this tutorial shows you how to use existing OpenSpan functionality with these connectors. OpenSpan Studio supports Office 2002, Office 2003, and Office 2007. When installing OpenSpan Studio, you will be asked to choose which version of MS Office you have.

Prerequisites

This tutorial requires a working knowledge of OpenSpan Studio. The tutorial also assumes that you are familiar with general Microsoft Office concepts.

Project Outline

This tutorial creates a solution containing a Win Form and two automations. The following list describes the project items of the solution.

Runtime Functions:

1. Create an automation that automatically logs into the CRM application.
Concepts: Reviews basic automation theory. It also prepares the system for the second automation.
2. Build an automation to populate the file in the CRM application New Call window. Instruct the automation to select the *Comments* tab in CRM and insert the comment 'I was here.'
Concepts: Using the Tab selection process of OpenSpan Studio. It also populates the data that will be exported to other systems.
3. Create a new worksheet in Excel and populate it with the last name entered into the CRM application.
Concepts: Using OpenSpan to export data to Microsoft Excel.
4. Build an automation to populate all of the bookmarked fields on a Microsoft Word document with the CRM account data.
Concepts: Using OpenSpan to populate bookmarked fields in a Microsoft Word document.
5. Design an automation to send an email confirming that the data has been sent.
Concepts: Using OpenSpan to send an email through Microsoft Outlook.

Microsoft Office Connectors

The following connectors enable OpenSpan Studio to work with Microsoft Office Word, Excel and Outlook.

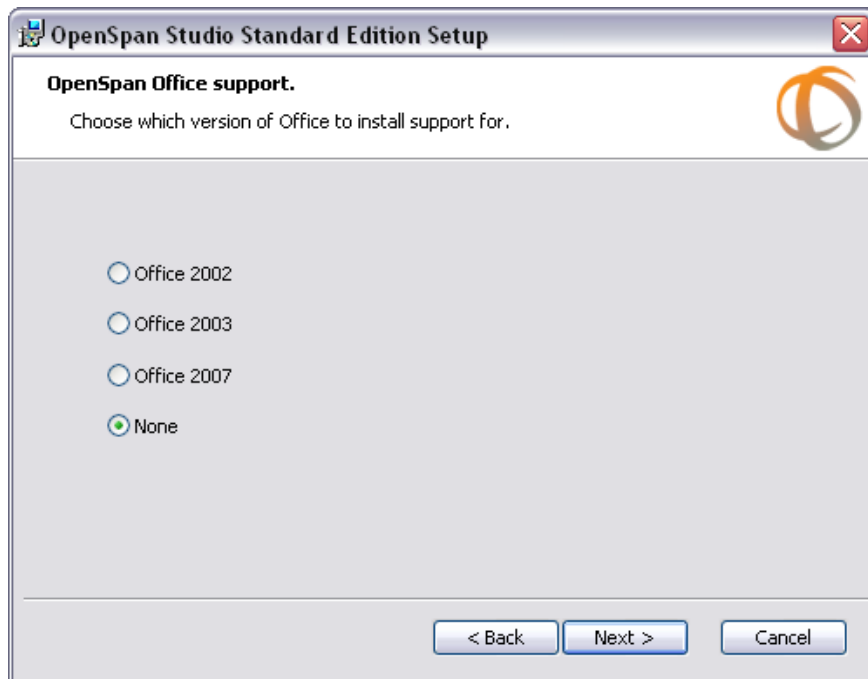


Microsoft Office Connectors

MicrosoftExcel	Allows you to interface with Microsoft Excel.
MicrosoftOutlook	Allows you to interface with the Microsoft Outlook application.
MicrosoftOutlook Appointment	Focuses on the Appointment function of Outlook.
MicrosoftOutlook Contact	Focuses on the Contact function of Outlook.
MicrosoftOutlookMail	Focuses on the Mail function of Outlook.
MicrosoftOutlook Contact	Focuses on the Contact function of Outlook.
MicrosoftOutlook Task	Focuses on the Task function of Outlook.
MicrosoftWord	Allows you to interface with the Microsoft Word application.

Installation Configuration

When installing OpenSpan Studio, you are given a choice of selecting which version of Microsoft Office you wish this installation of OpenSpan to support; Office 2002, Office 2003 or Office 2007.



See also:

Word Document Information:

[http://msdn2.microsoft.com/en-us/library/microsoft.office.interop.word.document_members\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/microsoft.office.interop.word.document_members(VS.80).aspx)

Excel Workbook Information:

[http://msdn2.microsoft.com/en-us/library/microsoft.office.interop.excel.workbook_members\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/microsoft.office.interop.excel.workbook_members(VS.80).aspx)

Exercise 1 – Log In Process

1. Create a New Blank Solution and name it CRM Office.
2. Add a Windows Adapter project item and name it CRM.
3. Set the path property to the CRM application in the OpenSpan Studio folder (as outlined in basic training.)
4. Interrogate the Login button and the User Name and Password fields of the CRM Login screen. Then interrogate all of the fields in the New Call Window screen (as outlined in basic training.)

Notes: Make sure to delete the Window Text match rule for the account number label and change the Mode property for the New Call window to Starts With and the Text to New Call. This will clear any matching issues you may have.

5. Create a new automation called Autx_Login automation as follows:

Object	Instructions
Login Created event	Connect this event to the txttxtCredentials Text property.
txttxtCredentials Text property	Connect this property to the Login Created event and the txttxtPassword Text property. Set the text property to UserName.
txttxtPassword Text property	Connect this property to the txttxtCredentials Text property and the btnLogin PerformClick method. Set the text property to Test.
btnLogin PerformClick method	Connect this method to the txttxtPassword text property.

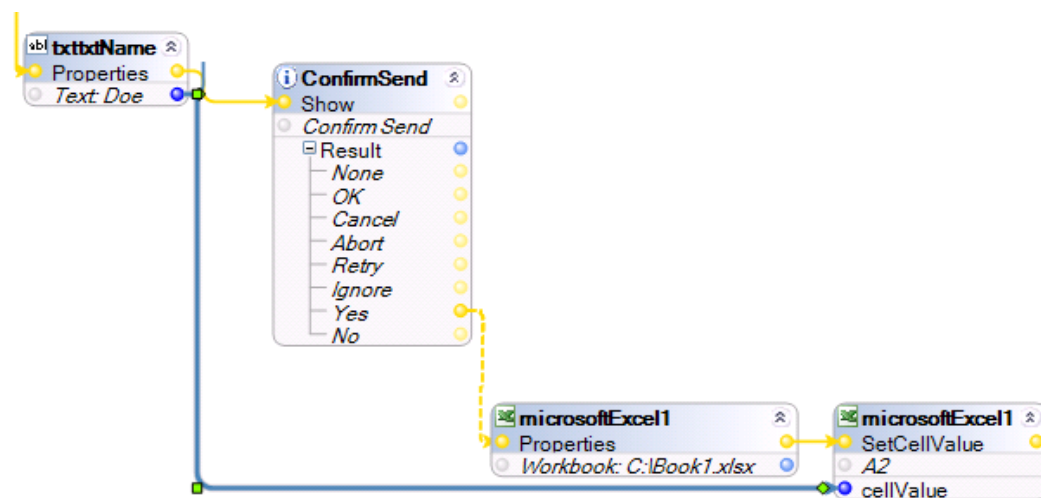
Your automation should appear as follows:



Exercise 2 – New Call Window and Export

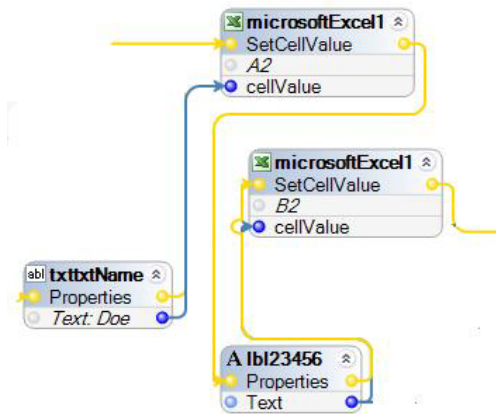
The following automation uses Excel, Outlook and Word to move data and showcase the Microsoft Office Connector capabilities. Use the following steps to create the solution.

1. Create a new automation called **Autx_Office Components**. Then add the Created event for the - Windows Form named **CRM_Subtraction_OpenSpan**.
2. Insert the New Call button Perform Click Event.
3. Use the tabMain SelectTab method for the Comments window.
4. Assign 'I was here!' property to interrogated txttxtComments window.
5. Assign City, State, Street Addr, Zip and Name aspects with their correct values.
6. Add a Message Dialog. Use the Show method with the text parameter set to "Confirm Send?" and the Button property changed to **YesNo**.
7. Add the Microsoft Excel connector. It will appear in the Local tab of the component tray by default. Highlight the component and select Make Global.
8. Create an Excel workbook.
9. Add the Workbook property for the selected workbook. Connect the Yes event from the ConfirmSend message dialog to the WorkBook Properties for the Excel object. Make the link asynchronous.
10. Link to the SetCellValue method for the A2 cell. Take the data from the Name field assigned earlier.



11. Connect this to the Account number property from the interrogated CRM object.
12. Connect to the SetCellValue Excel method for cell B2.

Microsoft Office Connectors



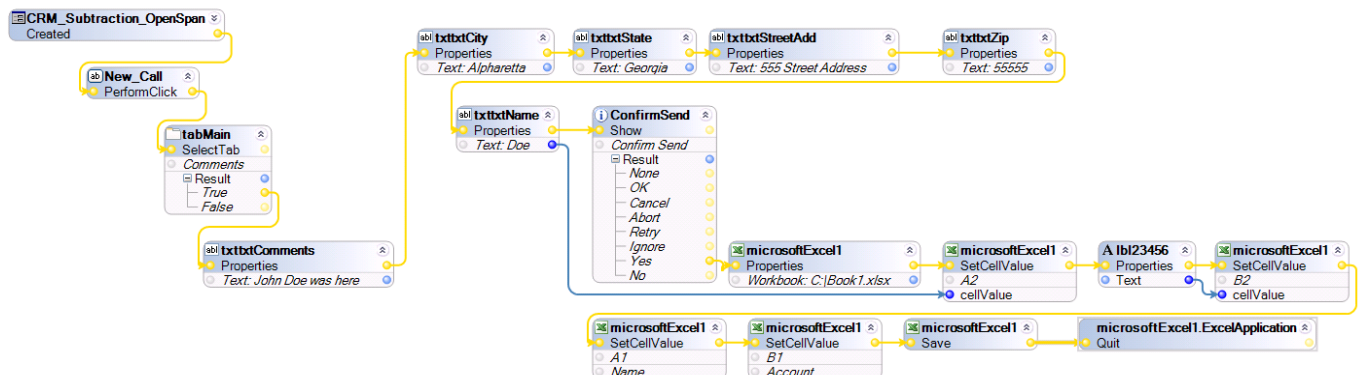
13. Create labels for these by setting the cell value for A1 and B1 to txttxtName and lbi23456 properties respectively.

14. Add the Excel Save method.

15. Use the Quit Method. To obtain the Quit method:

- Select the microsoftExcel1 connector in the Object Explorer.
- Click Explore Component Properties and highlight the **ExcelApplication** group.
- Click Configure Type. The Quit method may be selected by clicking Configure Type.

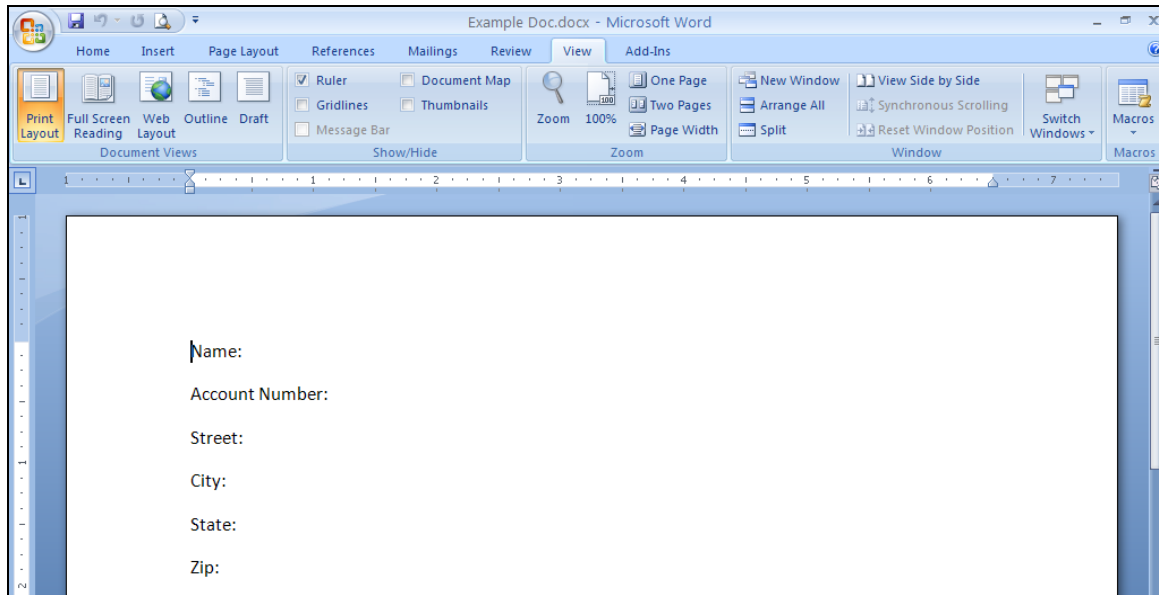
16. Select the **Quit** method and drag it out to the automation. Link to the save function. Your automation should look like the following:



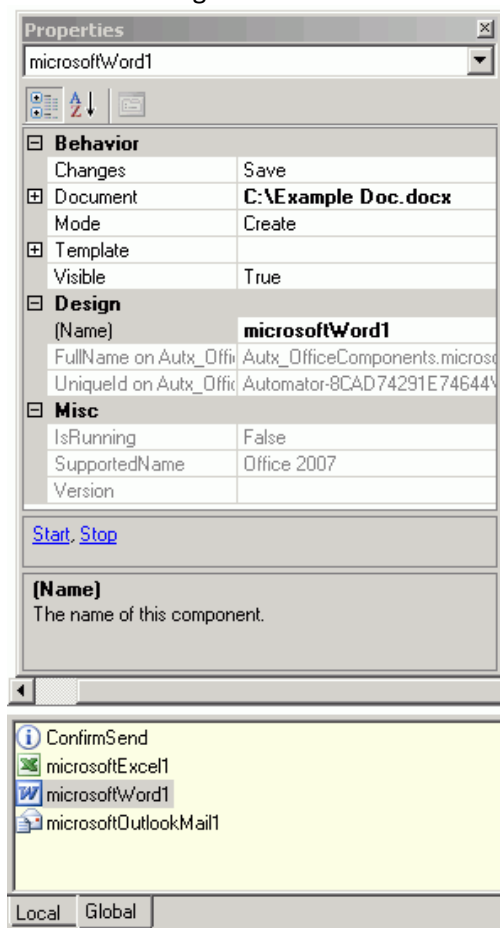
17. Prepare a word document with the text that you desire to plug in from the CRM application. In this case, "Name", "Street", "City", "State", "Zip" and "Account Number."

18. Insert a Bookmark for each field, just after the fieldname text, and save the file.

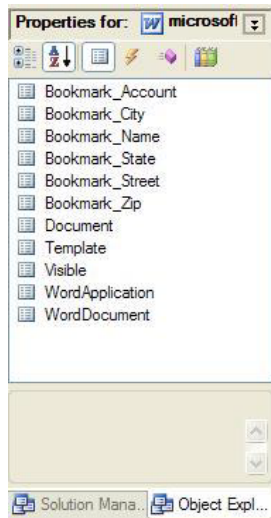
Microsoft Office Connectors



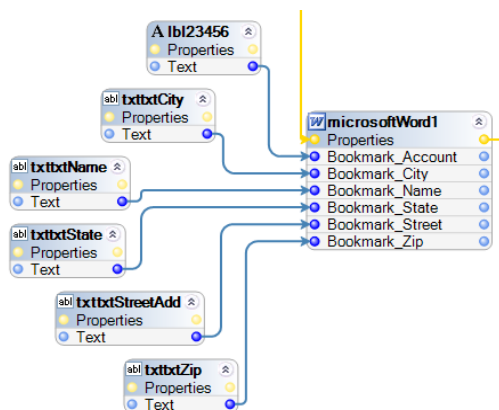
19. Drag the “Microsoft Word” connector over to the Global tab. Select the bookmarked document you just prepared by browsing in the document properties field. Word will open automatically. Close Word before returning to the solution.



20. The bookmarks should now appear in the properties of the word object. You can link the different bookmarks to create one Microsoft Word object by dragging the microsoftWord1 connector from Object Explorer onto the Design Pane and choosing the properties you want the block to contain.



21. Link the data from the CRM objects and plug it into the bookmarks. To gather all of the Word bookmark properties into one block, hold the Ctrl key down while selecting them.



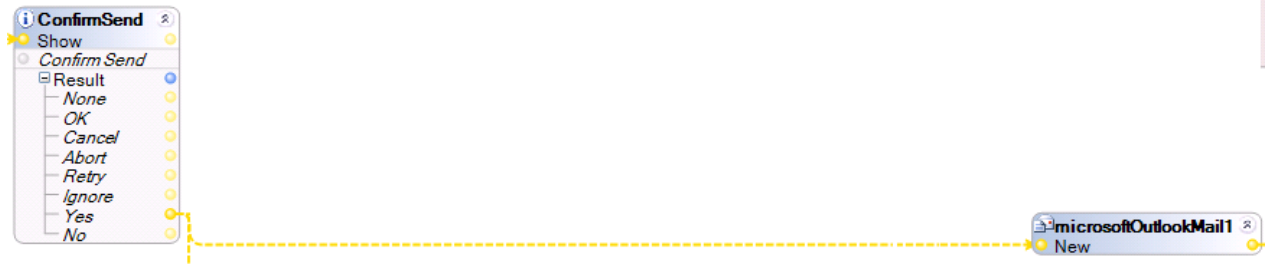
22. Connect the Yes result execution path from the Confirm Send Message Dialog to the Microsoft Word bookmark Properties object. Make the link Asynchronous.

23. Connect the MS Word bookmark Properties design block to the MS Word Save method.

24. Drag the Microsoft Outlook Mail connector over from the Connector ToolBar and drop it in the Global tab of the component tray.

25. Select the **New** method and connect it to the Yes event result from the Confirm Send message dialog. Make the link Asynchronous.

Microsoft Office Connectors



26. Drag the Microsoft Outlook Mail property Body to the automation and connect it to the New Method. Set the text Body to **This is the email that is sent when the confirmation button is clicked.**

27. Drag the microsoftOutlookMail1 To property on to the automation. Link the Body object to the execution path. Change the To parameter to your email address (for example `jdoue@openspan.com`). **This parameter must be a properly formatted email address or you will get an error.**

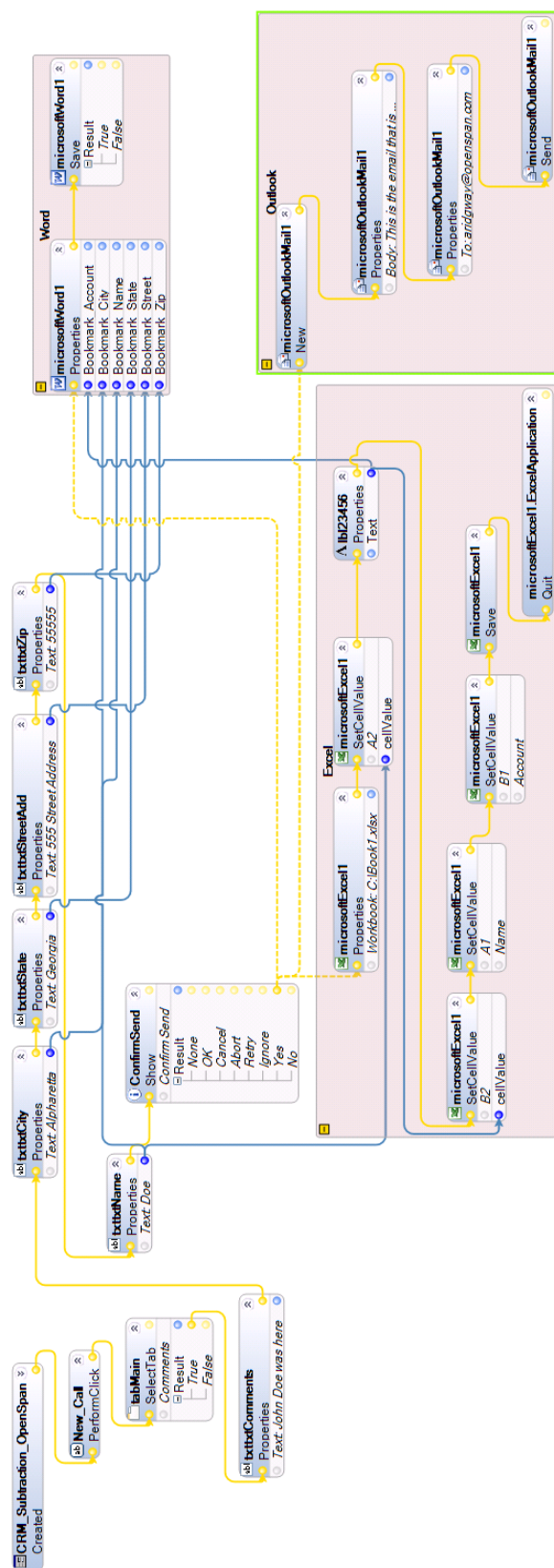
28. Drag the microsoftOutlookMail1 method Send to the automation and connect it to the microsoftOutLookMail1 To Property.

29. Make sure Word and Excel are not open. Save and Start the solution.

30. Stop the solution. You should have a document with name, account and address details as well as a spreadsheet with two rows of information and a note in your email inbox from yourself.

The completed automation looks like the following (group boxes have been added for clarity):

Page 12/30



A summary of the connections follows:

Object	Instructions
CRM_Subtraction_OpenSpan Created event	Connect this to the New_Call PerformClick method.
New_Call PerformClick method	Connect this to the CRM_Subtraction_OpenSpan_Created event and the tabMain SelectTab method "Comment" parameter.
tabMain SelectTab method	Connect this to the New_Call PerformClick method and the txttxtComments Text property.
txttxtComments Text property	Connect this to the tabMain SelectTab method True event and the txttxtCity Text property. Set the text property to John Doe Was Here.
txttxtCity Text property	Connect this to the txttxtComment Text event link of the txttxtState Text property. Set the text property to Alpharetta. Connect the data pathway to the Bookmark_City parameter of the microsoftWord1 connector.
txttxtState Text property	Connect this to the txttxtCity Text event link of the txttxtStreetAdd Text property. Set the text property to Georgia. Connect the data pathway to the Bookmark_State parameter of the microsoftWord1 connector.
txttxtStreetAdd Text property	Connect this to the txttxtState Text event link of the txttxtZip Text property. Set the text property to 555 Street Address. Connect the data pathway to the Bookmark_Street parameter of the microsoftWord1 connector.
txttxtZip Text property	Connect this to the txttxtStreetAdd Text event link of the txttxtName Text property. Set the text property to 5555. Connect the data pathway to the BookMark_Zip parameter of the microsoftWord1 connector.
txttxtName Text property	Connect this to the txttxtZip Text event link of the Confirm Send Show method. Set the text property to Doe. Connect the data pathway to the Bookmark_Name parameter of the microsoftWord1 connector and the cellvalue of the microsoftExcel1 SetCellValue (A2) method.

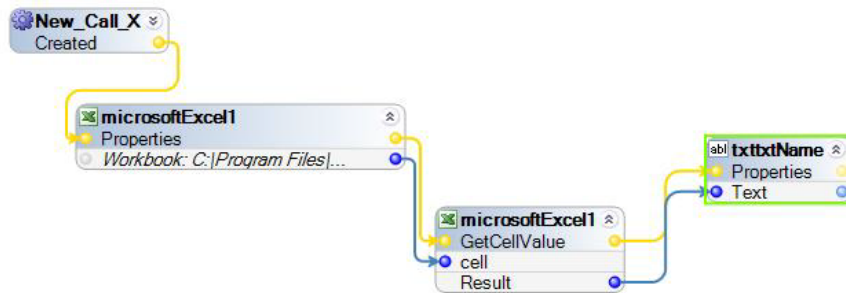
Object	Instructions
Confirm Send Message Dialog Show method Yes result event	Connect this method to the txttxtName Text property, the microsoftWord1 Connector, the microsoftExcel1 connector and the microsoftOutlookMail1 New method.
microsoftWord1 Connector (properties)	Connect this to the Confirm Send Show method and the microsoftWord1 Save method. Connect the Bookmark_Zip data parameter to the data connector of the txttxtZip Text property. Connect the Bookmark_Street data parameter to the data connector of txttxtStreetAdd text property. Connect the Bookmark_State parameter to the data connector of the txttxtState text property. Connect the Bookmark_Name parameter to the data connector of the txttxtName text property. Connect the Bookmark_City parameter to the txttxtCity text property. Connect the Bookmark_Account parameter to the data connector of the lbl23456 text property.
microsoftWord1 Save method	Connect this to the microsoftWord1 connector.
microsoftExcel1 connector (properties)	Connect this to the Confirm Send Show method and the microsoftExcel1 SetCellValue method. Set the Workbook property to the path of the created workbook.
microsoftExcel1 SetCellValue method (A2)	Connect this to the microsoftExcel1 connector and the lbl23456 Text property. Set the cell value to A2 . Connect the cellValue parameter to the data connector of the txttxtName Text property.
Lbl23456 Text property	Connect this to the microsoftExcel1 SetCellValue (A2) method and the microsoftExcel1 SetCellValue (B2) method. Connect the text data path to the cell value of the microsoftExcel1 Set Cell Value (B2) method and the BookMark_Account parameter of the microsoftWord1 connector.
microsoftExcel1 SetCellValue method (B2)	Connect this to the lbl23456 Text property and the microsoftExcel1 SetCellValue (A1) method. Set the Cell parameter to B2 . Connect the cellValue parameter to the data connector of the lbl23456 Text property.

Object	Instructions
microsoftExcel1 SetCellValue method (A1)	Connect this to the microsoftExcel1 SetCellValue method (B2) and the microsoftExcel1 SetCellValue method (B1). Set the cell parameter to A1 . Set the cellValue parameter to Last Name .
microsoftExcel1 SetCellValue method (B1)	Connect this to the microsoftExcel1 SetCellValue method (A1) and the microsoftExcel1 SaveAs method. Set the cell parameter to B1 and the Cell Value parameter to Account Number .
microsoftExcel1 SaveAs method	Connect this to the microsoftExcel1 SetCellValue method (B1) and the microsoftExcel1.ExcelApplication Quit method. Set the file name to ExcelFun2.
microsoftExcel1.ExcelApplication Quit method	Connect this to the microsoftExcel1 SaveAs method.
microsoftOutlookMail1 New method	Connect this to the Confirm Send Show method and the microsoftOutLokMail1 Body property.
microsoftOutLookMail1 Body property	Connect this to the microsoftOutLookMail1 New method and the microsoftOutLookMail1 To property. Set the Body parameter to This is the email that is being sent when the confirm button is pressed .
microsoftOutlookMail1 To property	Connect this to the microsoftOutlookMail1 Body property and the microsoftOutlookMail1 Send method. Set the To property to the desired email.
microsoftOutlookMail1 Send method	Connect this to the microsoftOutlookMail1 To property.

Connector Details

Microsoft Excel Connector

The 'GetCellValue' method returns the value of the most recently selected cell from an Excel spreadsheet. The following automation is an example of how to use the Microsoft Excel Connector. The solution retrieves this data and populates the Name field in the CRM application whenever a new call window is opened.



Exercise

The 'ImportData' method imports data into the current workbook. The 'Visible' property registers whether or not the Excel window is open. The 'SheetChange' event occurs whenever the current worksheet changes. The 'SheetActivate' event occurs whenever a workbook sheet is activated. Finally, the 'Workbooksaved' event occurs whenever the current workbook is saved.

Use the following steps to create an automation that works with the Import and Export methods of the Excel connector.

Steps:

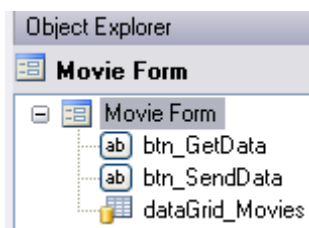
- 1) Create an Excel spreadsheet called OpenSpanTest.xls. Save it in the C:\ root directory. Populate it with the data and format it as shown in this image.

	A	B
1	Rank	Movie
2	1	Ocean's 13
3	2	Die Hard 4
4	3	Transformers: The Movie
5	4	Bourne Ultimatum, The
6	5	Shrek The Third
7	6	Simpsons Movie, The
8	7	Spider-man 3
9	8	Taxi 4
10	9	300
11	10	Resident Evil: Extinction
12		

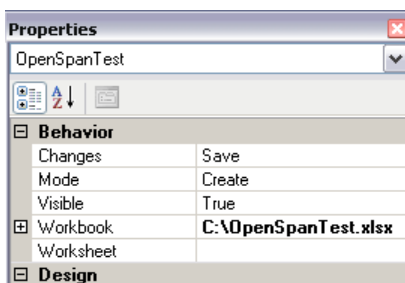
- 2) Create a second blank workbook at the root directory called OpenSpanTest2.xls.
- 3) In OpenSpan Studio, create a new solution named Excel Test.
- 4) Create a Windows form named Movie Form with two buttons and a data grid:



- 5) Name the design components as follows:



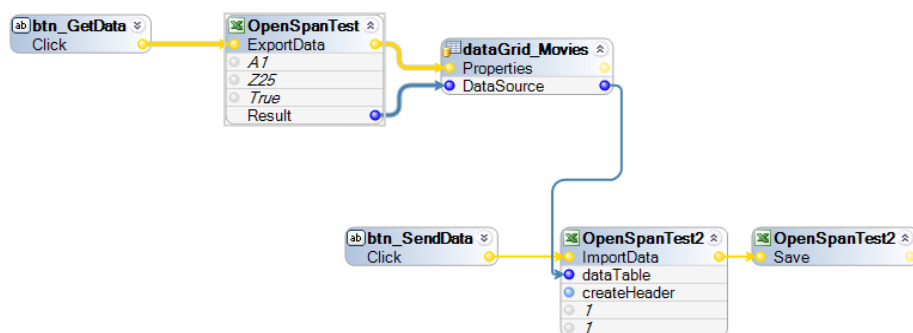
- 6) Create an automation called **Autx_Movies**.
- 7) Insert the Click event for btn_GetData.
- 8) Add an Excel connector to the Global tab of the component tray of the automation. Name this **OpenSpanTest**.
- 9) Set the workbook property for the Excel connector to **C:\Openspantest.xls**.



- 10) Export the data using the Export data method for the Excel connector. Set the range to be exported from **A1 to B10**. Set the Create Header parameter to **True**.

- 11) Connect the Click event from the btn_GetData to the Excel Connector ExportData method.
- 12) Connect the Result of the Excel Export data method to the dataGrid_Movies Data Source incoming data port.
- 13) Add a second Excel connector to the Global tab of the component tray.
- 14) Name this connector **OpenSpanTest2**.
- 15) Set the Workbook property to **C:\OpenSpanTest2.xlsx (or xls for Office 2003 or 2002)**.
- 16) Add the Click event for btn_SendData.
- 17) Add the ImportData method for OpenSpanTest2.
- 18) Connect the dataGrid_Movies object DataSource output to the DataTable parameter on the OpenSpanTest2 Import data method.
- 19) Set the RowStart to **1** and the Column Start parameter to **1**. Set the Create Header parameter to **True**.
- 20) Connect the Save Method from OpenSpan2 Import Data method to the OpenSpanTest2 Import Data method.
- 21) Save the solution.
- 22) Start the solution.
- 23) Press the **Get Data** button.
- 24) Press the **Send Data** button.
- 25) Stop the solution.

Your completed automation should look like the following:



A summary of the connections in the automation follows:

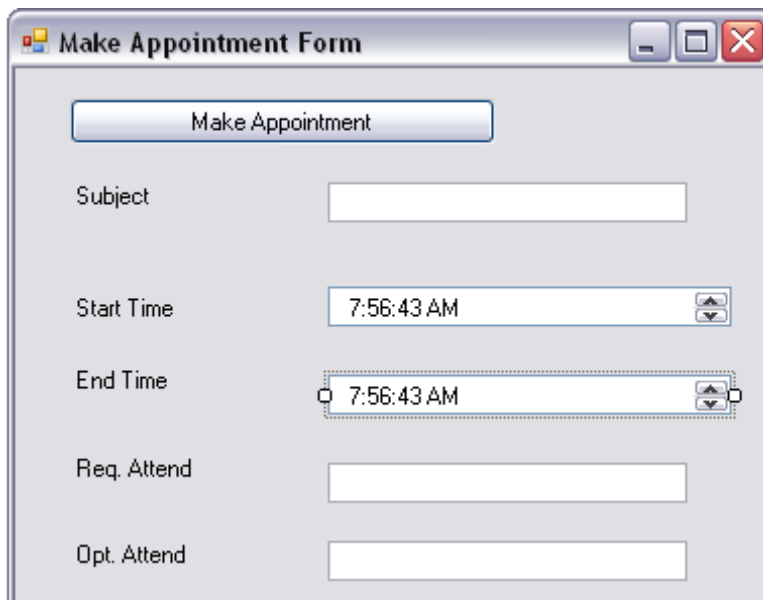
Object	Instructions
Btn_GetData Click event	Connect this to the OpenSpanTest ExportData method.

OpenSpanTest ExportData method	Connect this to the btn_GetData Click event and the dataGrid_Movies DataSource property. Connect the Result parameter to the data connector on the dataGrid_Movies Data Source property. Set the CellStart parameter to A1 . Set the CellEnd parameter to B10 . Set the createHeader parameter to True .
dataGrid_Movies DataSource property	Connect this to the OpenSpanTest ExportData method. Connect the incoming data connector to the result parameter from the OpenSpanTest ExportData method. Connect the outgoing data connector to the dataTable connector of the OpenSpanTest2 Import Data method.
Btn_SendData Click Event	Connect this to the OpenSpanTest2 ImportData method.
OpenSpanTest2 ImportData method	Connect this to the btn_SendData Click method and the OpenSpanTest2 Save method. Connect the dataTable parameter to the outgoing data connector for the dataGrid_Movies Data Source property. Set the rowStart parameter to 1 and the ColumnStart parameter to 1 .
OpenSpanTest2 Save method.	Connect this to the OpenSpanTest2 ImportData method.

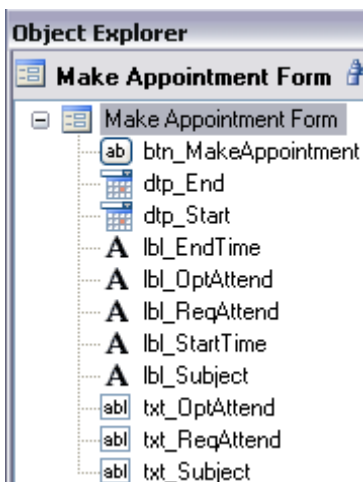
Microsoft Outlook Appointment Connector

The Microsoft Outlook Appointment Connector allows you to initiate and modify appointments in the Microsoft Outlook Calendar. Please note, when you create a new appointment, it will not appear in the calendar until you save the data. If you want the appointment window to come up for modification, use the **Display** method. Use the following steps to create this solution.

1. Create a new blank solution named Outlook Appointment.
2. Add a Windows Form Project Item.



3. Name the Project Item **Make Appointment Form** and name the components as follows:

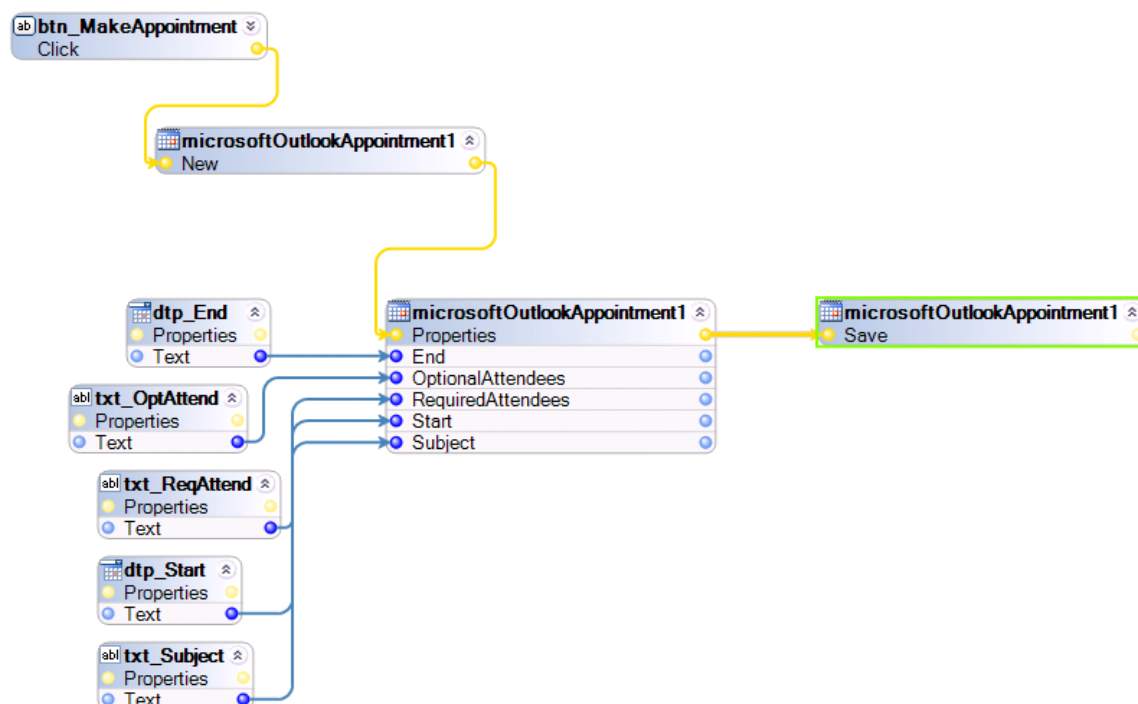


4. Set the Format property on the Date Time Picker control to Time.
5. Set the ShowUpDown property on the Date Time Picker to **True**.
6. Add a new automation named autx_MakeAppointment.

7. Add the Click event for the Make Appointment button.
8. Add the Microsoft Outlook Appointment connector to the Global tab of the component tray of the automation.
9. Add the New method for the Microsoft Outlook Appointment connector. Connect the Click event to the New method.
10. Hold the CTRL key down and select the following properties for the Microsoft Outlook Appointment: end, Optional Attendees, Required Attendees, Start, and Subject. Drag these properties as a group to the automation.
11. Connect the text property for the subject, start, end, ReqAttend and OptAttend fields to the Subject, Start, End, RequiredAttendees and OptionalAttendees properties for the Microsoft Outlook Appointment connector respectively.
12. Connect the New method output event node to the Microsoft Outlook Appointment input event node.
13. Connect the Microsoft Outlook Appointment connector Save method to the Microsoft Outlook Appointment block.
14. Save the automation.
15. Start the solution.
16. Enter a subject.
17. Enter a start time a half hour into the future and an end time two hours into the future.
18. Enter required and optional attendees (You might consider putting your own name in both fields just to avoid sending others meeting requests.)
19. Click the Make Appointment button.
20. Check your Outlook Calendar. You should see an appointment with the appropriate data.
21. Stop the solution.

Your automation should look like the following:

Microsoft Office Connectors



A summary of the connections made in the automation follows:

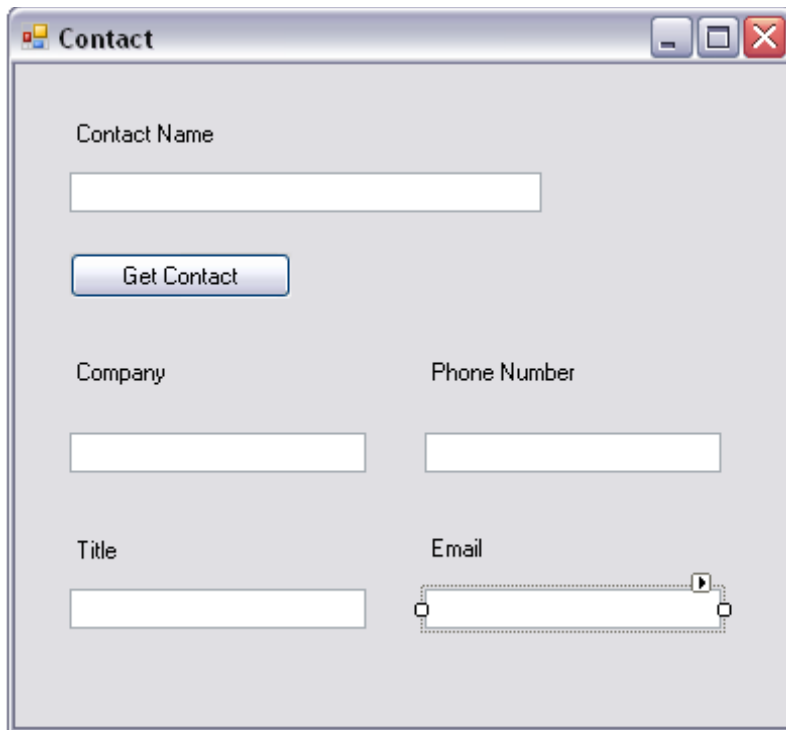
Object	Instructions
btn_MakeAppointment Click event	Connect this to the microsoftOutlookAppointment1 New method.
microsoftOutlookAppointment1 New method	Connect this to the btn_MakeAppointment Click method and the microsoftOutlookAppointment1 Subject property.
microsoftOutlookAppointment1 Subject property	Connect this to the microsoftOutlookAppointment1 New method and the microsoftOutlookAppointment1 Start property. Connect the incoming data connector to the txt_Subject Text property.
txt_Subject Text property	Connect the outgoing data connector to the incoming data connector of the microsoftOutlookAppointment1 Subject property.
microsoftOutlookAppointment1 Start property	Connect this to the microsoftOutlookAppointment1 Subject property and the microsoftOutlookAppointment1 End property. Connect the incoming data connector to the dtp_Start Text property.

dtg_Start Text property	Connect the outgoing data connector to the incoming data connector of the microsoftOutlookAppointment1 Start property.
microsoftOutlookAppointment1 End Property	Connect this to the microsoftOutlookAppointment1 Start property and the microsoftOutlookAppointment1 RequiredAttendees property. Connect the incoming data connector to the dtg_End Text property.
dtg_End Text property	Connect the outgoing data connector to the incoming data connector of the microsoftOutlookAppointment1 End property.
microsoftOutlookAppointment1 RequiredAttendees property	Connect this to the microsoftOutlookAppointment1 End property and the microsoftOutlookAppointment1 OptionalAttendees property. Connect the incoming data connector to the txt_ReqAttend Text property.
txt_ReqAttend Text Property	Connect the outgoing data connector to the incoming data connector of the microsoftOutlookAppointment1 RequiredAttendees property.
microsoftOutlookAppointment1 OptionalAttendees property	Connect this to the microsoftOutlookAppointment1 RequiredAttendees property and the microsoftOutlookAppointment1 Save method. Connect the incoming data connector to the txt_OptAttend Text property.
txtOptAttend Text property	Connect the outgoing data connector to the incoming data connector of the microsoftOutlookAppointment1 OptionalAttendees property.
microsoftOutlookAppointment1 Save method	Connect this to the microsoftOutlookAppointment1 OptionalAttendees property.

Microsoft Outlook Contact

This exercise creates a windows form that extracts contact information from Microsoft Outlook.

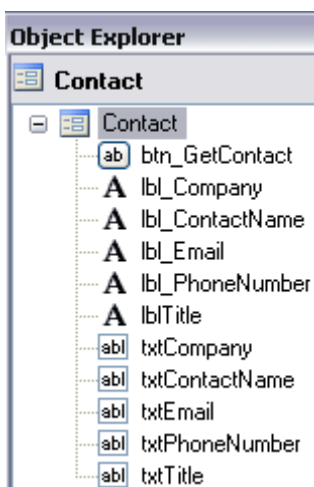
1. Create a new blank solution named Outlook Contact.
2. Either enter a new contact to use for testing purposes in Microsoft Outlook, or find an existing one that has all of the information needed (Name, Title, Company, Phone Number and Email.)
3. Create a Windows form as follows:



The screenshot shows a Windows form titled "Contact". It contains the following elements:

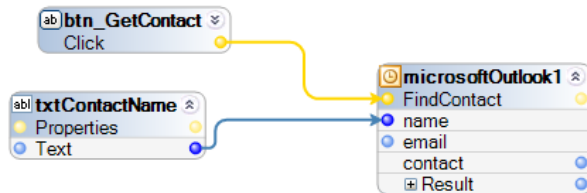
- A text box labeled "Contact Name" with a single-line input field below it.
- A button labeled "Get Contact" below the "Contact Name" text box.
- Two text boxes side-by-side: "Company" on the left and "Phone Number" on the right, each with a single-line input field below it.
- Two text boxes side-by-side: "Title" on the left and "Email" on the right. The "Email" text box has a single-line input field below it with a scroll bar on the right.

4. Name the Design Form **Contact** and Name the components as follows:

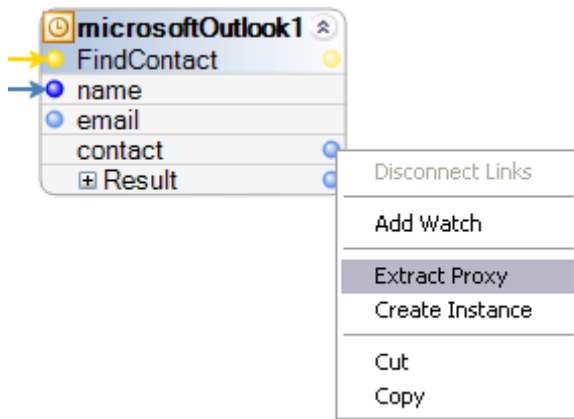


5. Add a new automation called autx_GetContact.

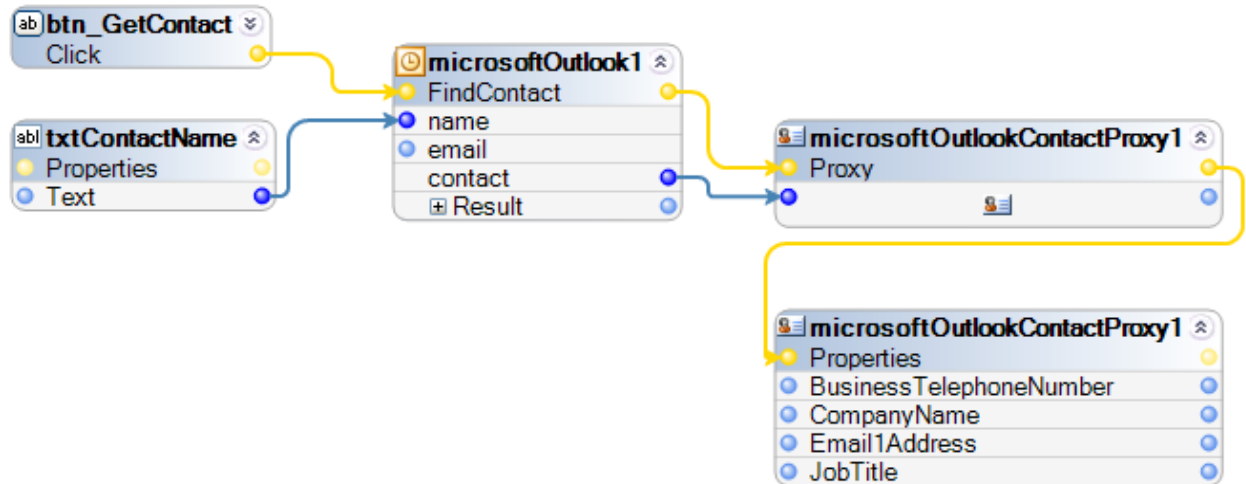
6. Drag and drop the btn_GetContact Click event onto the autx_GetContact automation.
7. Insert the txtContactName Text property.
8. Add the Microsoft Outlook connector to the Global tab of the component tray of the automation.
9. Connect these to the microsoftOutlook FindContact method as shown below:



10. Right click on the data connector next to the contract parameter and select Extract Proxy. This allows you to access extra properties, methods and events for the selected object.

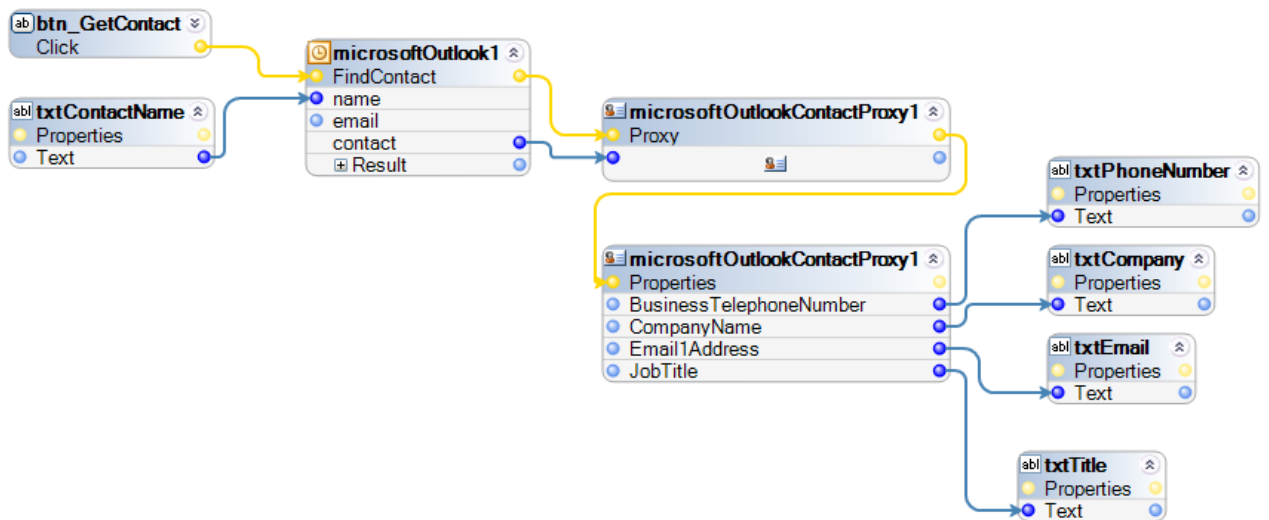


11. From the properties, methods and events of the proxy, add the CompanyName, Email1Address, JobTitle and BusinessTelephoneNumber properties. Note: Hold the CTRL key down while selecting the properties and then drag them over as a block onto the automation. Connect the event links from the Find Contact method to the Proxy and then to the Properties block as shown below:



12. Connect the txtEmail, txtCompany, txtPhoneNumber and txtTitle fields to their corresponding Outlook Contact properties.

13. Your completed automation should look like the following:



14. Save the solution.

15. Start the solution.

16. Type the name of someone that you have as an actual contact into the Contact Name field. If you don't have one, add a contact into Microsoft Outlook.

17. Press the **Get Contact** button. The fields will fill with the relevant contact data.

The screenshot shows a window titled "Contact" with a standard Windows-style title bar. Inside the window, there are five text input fields arranged in a form. The first field is labeled "Contact Name" and contains the text "John Doe". Below this field is a button labeled "Get Contact". The second field is labeled "Company" and contains "Generic Company". The third field is labeled "Phone Number" and contains "(555) 555-5555". The fourth field is labeled "Title" and contains "Manager". The fifth field is labeled "Email" and contains "jdoe@genericcompany.com".

18. Stop the solution.

A summary of automation connections follows:

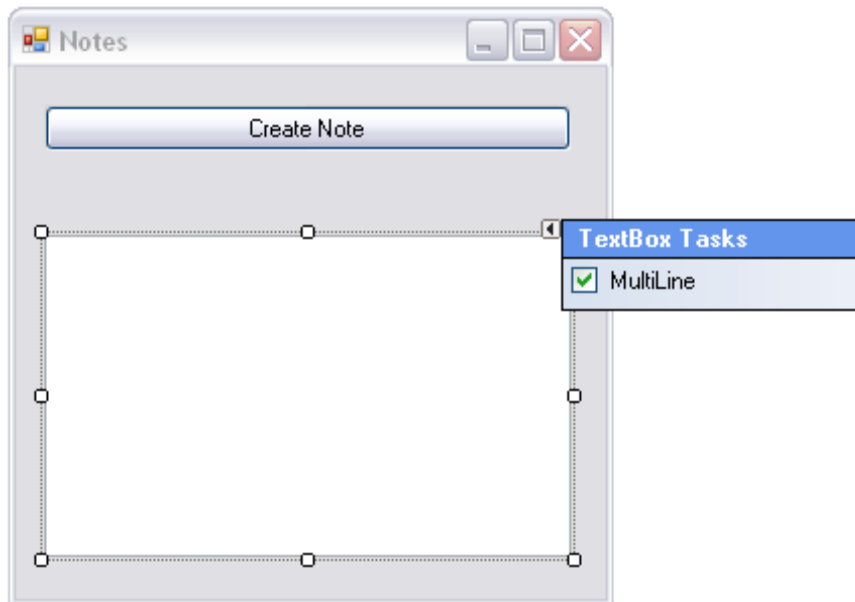
Object	Instructions
btn_GetContact Click Event	Connect this to the microsoftOutlook1 FindContact method.
txtContactName Text property	Connect the outgoing data connector to the name parameter of the microsoftOutlook1 FindContact method.
microsoftOutlook1 FindContact method	Connect this to the btn_GetContact Click event and the microsoftOutlookContactProxy1. Connect the outgoing contact parameter data connector to the microsoftOutlookContactProxy1. Connect the outgoing data connector of the txtContactName Text property to the name parameter.
microsoftOutlookContactProxy1	Connect this to the microsoftOutlook1 Find Contact method and the microsoftOutlookContactProxy1 CompanyName property. Connect the outgoing contact data connector to the incoming data connector.

microsoftOutlookContactProxy1 CompanyName property	Connect this to the microsoftOutlookContactProxy1 and the microsoftOutlookContactProxy1 Email1Address property. Connect the outgoing data port to the txtCompany Text property.
microsoftOutlookContactProxy1 Email1Address property	Connect this to the microsoftOutlookContactProxy1 CompanyName property and the microsoftOutlookContactProxy1 JobTitle property. Connect the outgoing data port to the txtEmail Text property.
microsoftOutlookContactProxy1 JobTitle property.	Connect this to the microsoftOutlookContactProxy1 Email1Address property and the microsoftOutlookContactProxy1 Business TelephoneNumber property. Connect the outgoing data port to the txtPhoneNumber Text property.
microsoftOutlookContactProxy BusinessTelephoneNumber property	Connect this to the microsoftOutlookContactProxy1 JobTitle property. Connect the outgoing data connector to the txtPhoneNumber Text property.
txtCompany Text property	Connect the incoming data connector to the outgoing data connector of the microsoftOutlookContactProxy1 CompanyName property.
txtEmail Text property	Connect the incoming data connector to the outgoing data connector of the microsoftOutlookContactProxy1 Email1Address property.
txtTitle Text property	Connect the incoming data connector to the outgoing data connector of the microsoftOutlookContactProxy1 JobTitle property.
txtPhoneNumber text property	Connect the incoming data connector to the outgoing data connector of the microsoftOutlookContactProxy1 BusinessTelephoneNumber property.

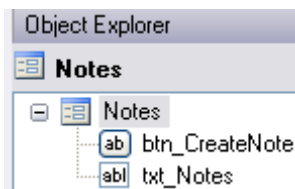
Microsoft Outlook Notes

Use the following steps to add a note to Outlook.

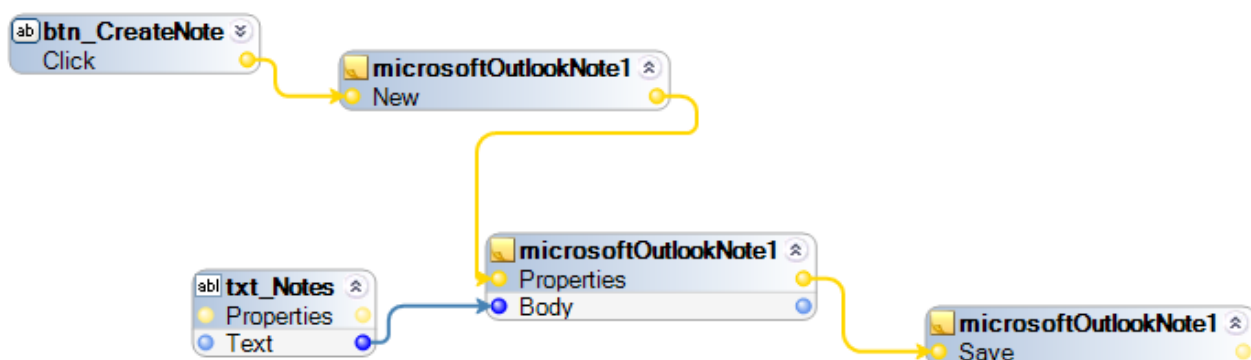
1. Create a new blank solution named Outlook Notes.
2. Create the Windows form as follows:



3. Name the form Notes and name the components as follows:



3. Add an automation called autx_CreateNote.
4. Add the btn_CreateNote Click event.
5. Add a MicrosoftOutlookNote connector to the Global tab of the component tray.
6. Add the New method.
7. Connect the Click event to the New method.
8. From the microsoftOutlookNote1 object, add the Body property. Connect the output event link from the New method to the Body property.
9. Add the Save method. Connect the output event node from the Body property to the Save method.
10. Connect the txt_Notes text property to the data point of the microsoftOutlookNote1 Body property.
11. Your solution should look like the following:



12. Save the solution.
13. Start the solution.
14. Enter a note into the text box.
15. Click the Create Note button.
16. Stop the solution.
17. Check your Outlook box for the new note. There should be a note visible with the information you that you entered into the Create Note form.

A summary of the connections in the automation follows:

Object	Instructions
btn_CreateNote Click event	Connect this to the microsoftOutlookNote1 New Method.
microsoftOutlookNote1 New method	Connect this to the btn_CreateNote Click event and the microsoftOutlookNote1 Body property.
txt_Notes Text property	Connect the outgoing data connector to the Body parameter of the microsoftOutlookNote1 Body property.
microsoftOutlookNote1 Body property	Connect this to the microsoftOutlookNote1 New method and the microsoftOutlookNote1 Save method. Connect the incoming data connector to the txt_Notes Text Property.
microsoftOutlookNote1 Save method	Connect this to the microsoftOutlookNote1 Body property.