Official Guide

Add-in Express™ .net

Getting Started

Add-in Express 2007 for Microsoft® .NET

Add-in Express™ 2007 for Microsoft® .NET

Document version **3.0**

Revised at **27-Dec-06**

Product version **3.x**

Table of Contents

Introduction

Add-in Express .NET is a development tool designed to simplify and speed up the development of Office COM Add-ins, Run-Time Data servers (RTD servers), Smart Tags, and Excel Automation Add-ins in Visual Studio 2003 and Visual Studio 2005 through the consistent use of the RAD paradigm. It provides a number of specialized components that allow the developer to walk through the interface-programming phase to the functional programming phase with a minimal loss of time

.

Why Add-in Express?

Microsoft supplied us with another term – Office Extensions. This term covers all the customization technologies provided for Office applications. The technologies are:

- COM Add-ins
- Smart Tags
- Excel RTD Servers
- Excel Automation Add-ins

Add-in Express allows you to overcome the basic problem when customizing Office applications in .NET – building your solutions into the Office application. Based on the True RAD paradigm, Add-in Express saves the time that you would have to spend on research, prototyping, and debugging numerous issues of any of the above-said technologies in all versions and updates of all Office applications. The issues include safe loading / unloading, host application startup / shutdown, as well as user-interaction-related and deployment-related issues.

Add-in Express provides you with simple tools for creating version-neutral, secure, insolated, managed, deployable, and updatable Office extensions.

- **Managed Office Extensions**

You develop them in every programming language available for Visual Studio .NET.

- **Isolated Office Extensions**

Add-in Express allows loading Office extensions into separate application domains. So, they do not have a chance to break down the host application.

- **Version-neutral Office Extensions**

The Add-in Express programming model and its core are version-neutral. It gives you a chance to develop one Office extension for all available Office versions, from 2000 to the newest 2007.

- **Deployable Office Extensions**

Add-in Express automatically supplies you with a setup project making your solution ready-to-deploy.

- **Updatable Office Extensions**

Add-in Express uses the start-up and deployment model that allows updating your deployed solutions at run-time.

Add-in Express Extensions

Depending on a product package, Add-in Express includes plug-ins that allow a deeper customization of Office applications using Add-in Express COM Add-ins.

- **The Add-in Express Extensions .NET for Microsoft Outlook**

The Add-in Express Extensions allows Outlook COM Add-in developers to embed .NET forms into Outlook sub-panes for the Explorer and Inspector windows. See the product description at our web-site – http://www.add-in-express.com/outlook-extension/.

- **The Add-in Express Extensions for Microsoft Office Toolbars**

Office COM Add-in developers can use the Add-in Express Extensions for Microsoft Office Toolbars to place any .NET controls (such as grids, tree views, lists) on Microsoft Office toolbars. See the product description at our web-site – http://www.add-in-express.com/office-toolbar-controls/.

- **Outlook Security Manager**

This is a standalone product designed for Outlook solution developers. It allows controlling Outlook E-mail Security Guard by turning it off and on in order to suppress unwanted Outlook Security warnings.

System Requirements

Supported IDEs

Visual Studio 2003 Editions

- Visual Studio .NET 2003 Enterprise Architect Edition
- Visual Studio .NET 2003 Enterprise Developer Edition
- Visual Studio .NET 2003 Professional Edition
- Visual Studio .NET 2003 Standard Edition
- RemObjects Chrome 1.53 and higher

Visual Studio 2005 Editions

- Visual Studio .NET 2005 Team System
- Visual Studio .NET 2005 Professional Edition
- Visual Studio .NET 2005 Standard Edition
- Visual Basic .NET 2005 Express
- Visual C# 2005 Express
- RemObjects Chrome 1.53 and higher

Borland Delphi Editions

- Delphi 2005 Architect
- Delphi 2005 Enterprise
- Delphi 2005 Professional
- Delphi 2006 Architect
- Delphi 2006 Enterprise
- Delphi 2006 Professional

Not supported

Visual Studio 2003 Trial (Learning) Edition and Delphi Trial (Learning) Edition are not supported.

Supported Languages

- Visual Basic .NET 2003
- Visual Basic .NET 2005
- Visual Basic .NET 2005 Express
- Visual C# 2003
- Visual C# 2005
- Visual C# 2005 Express
- Visual C++ .NET 2003
- Visual C++ .NET 2005
- Delphi for .NET

Host Applications

COM Add-ins

- Microsoft Excel 2000 and higher
- Microsoft Outlook 2000 and higher
- Microsoft Word 2000 and higher
- Microsoft FrontPage 2000 and higher
- Microsoft PowerPoint 2000 and higher
- Microsoft Access 2000 and higher
- Microsoft Project 2000 and higher
- Microsoft MapPoint 2002 and higher
- Microsoft Visio 2002 and higher
- Microsoft Publisher 2003 and higher
- Microsoft InfoPath 2007

Real-Time Data Servers

- Microsoft Excel 2002 and higher

Smart Tags

- Microsoft Excel 2002 and higher
- Microsoft Word 2002 and higher
- Microsoft PowerPoint 2003 and higher

Office editions, service packs and updates

Add-in Express supports all Microsoft Office service packs and updates for all Microsoft Office editions including Student, Basic, Standard, Professional, Small Business, Enterprise, etc.

Technical Support

Add-in Express is developed and supported by the Add-in Express Team, a branch of Add-in Express Ltd. You can get technical support using any of the following methods.

Add-in Express Web Site

The Add-in Express web-site at www.add-in-express.com provides a wealth of information and software downloads for Add-in Express developers, including:

- The HOWTOs section that contains sample projects answering most common "how to" questions.
- Add-in Express Toys contains entire and "open sourced" add-ins for popular Office applications.
- Forums. We are actively participating in these forums. Really.

Internet E-mail Support

For technical support through the Internet, e-mail us at support@add-in-express.com.

Technical Support via Instant Messengers

If you are a subscriber of our Premium Support Service and need help immediately, you can request technical support via an instant messenger, e.g. Windows/MSN Messenger or ICQ. Please ask us at http://www.add-in-express.com/premium-area/contact-us.php.

Installing and Activating

There are two main points in the Add-in Express .NET installation. First off, you have to specify the development environments that you need to use Add-in Express .NET in. See Supported IDEs. Second, you need to activate the product. What follows below is a brief guide on activating.

Activation Basics

During software registration, the registration wizard prompts you to enter your license key. The key is a 25 character alphanumeric code shown in five groups of five characters each (for example, GBFTK-3UN78-MKF8G-T8GTY-NQS8R). Keep the license key in a safe location and do not share it with others. This product key forms the basis for your ability to use the software.

For purposes of product activation only, a non-unique hardware identifier is created from general information that is included in the system components. At no time are files on the hard drive scanned, nor is personally identifiable information of any kind used to create the hardware identifier. Product activation is completely anonymous. To ensure your privacy, the hardware identifier is created by what is known as a "one-way hash". To produce a one-way hash, information is processed through an algorithm to create a new alphanumeric string. It is impossible to calculate the original information from the resulting string.

Your product key and a hardware identifier are the only pieces of information required to activate the product.

If you choose the Automatic Activation Process option of the Activation Wizard, the wizard attempts to establish an online connection to the activation server, www.activatenow.com. If the connection is established, the wizard sends both the license key and the hardware identifier over the Internet. The activation service generates an activation key using this information and sends it back to the activation wizard. The wizard saves the activation key to the registry.

If an online connection cannot be established (or you choose the Manual Activation Process option), you can activate the software using your web-browser. In this case, you will be prompted to enter the product key and a hardware identifier on a Web page, and will get an activation key in return. This process finishes with saving the activation key to the registry.

Activation is completely anonymous; no personally identifiable information is required. The activation key can be used to activate the product on that computer an unlimited number of times. However, if you need to install the product on several computers, you will need to perform the activation process again on every PC. Please refer to your end-user license agreement for information about the number of computers you can install the software on.

Setup Package Contents

The Add-in Express .NET setup program installs the following folders on your PC:

- Bin – Add-in Express .NET binary files
- Demo Projects – demo projects for Visual Studio 2003 and Visual Studio 2005
- Docs - Add-in Express .NET documentation
- Docs \ Samples – sample documentation projects
- Images – Add-in Express .NET icons
- Redistributables – Add-in Express .NET redistributable files
- Sources - Add-in Express .NET source code (except for design-time source code).

Where is the Add-in Express .NET source code?

Please note we include the source code of Add-in Express .NET according to the product package you purchased. See the Packages & Pricing page for details.

Add-in Express .NET setup program installs the following text files on your PC:

- licence.txt – EULA
- readme.txt – short description of the product, support addresses and such
- whatsnew.txt – this file describes the latest information on the product features added and bugs fixed.

Getting Started

In this chapter, we guide you through the following steps of developing Add-in Express Projects:

- Create an Add-in Express project
- Add an Add-in Express Designer to the project
- Add Add-in Express components to the Add-in Express Designer
- Add some business logics
- Build, register, and debug the Add-in Express project
- Tune up the Add-in Express Loader based setup project
- Deploy your project to a target PC

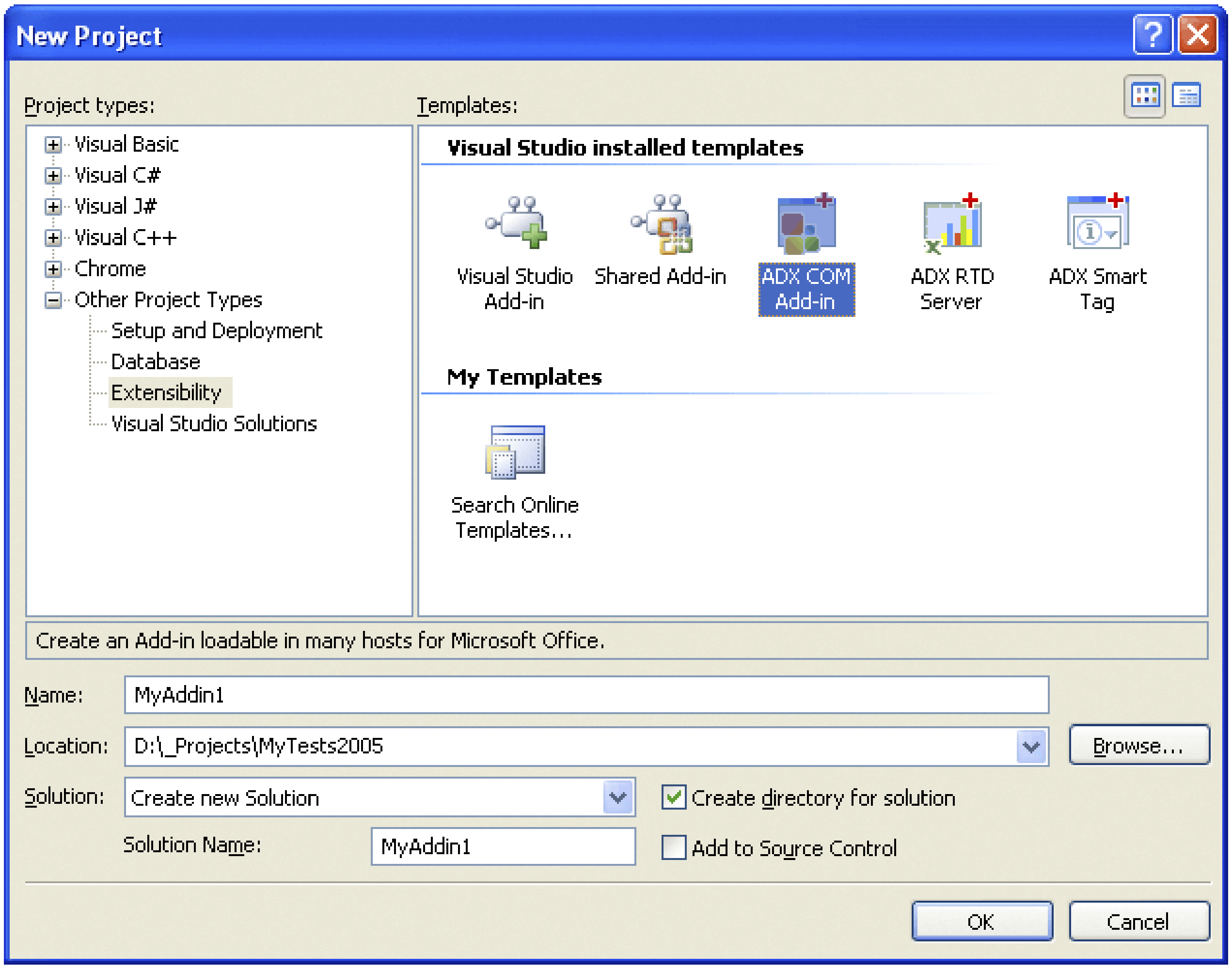
For the sake of time, in our support emails we use the ADX.NET acronym when referring to Add-in Express .NET.

You find all the below described samples in the following folder on your PC:

C:\Program Files\Add-in Express\Add-in Express .NET <ProductPackage>\Docs\Samples

Creating Add-in Express Projects

Add-in Express .NET adds several project templates to the Extensibility folder of the New Project dialog. To see the dialog, choose the “File | New | Project…” menu.


New Project

Project types:
- Visual Basic
- Visual C#
- Visual J#
- Visual C++
- Chrome
- Other Project Types
  - Setup and Deployment
  - Database
  - Extensibility
  - Visual Studio Solutions

Templates:

Visual Studio installed templates: Visual Studio Add-in, Shared Add-in, ADX COM Add-in, ADX RTD Server, ADX Smart Tag

My Templates: Search Online Templates…

Create an Add-in loadable in many hosts for Microsoft Office.

Name: MyAddin1
Location: D:_Projects\MyTests2005
Solution: Create new Solution
Solution Name: MyAddin1
[✓] Create directory for solution
[ ] Add to Source Control

Whichever Add-in Express Project template you use, it starts the Add-in Express Project Wizard that allows selecting a programming language for your Add-in Express project as well as other options. The Add-in Express Project Wizard creates the solution and adds two projects: an Add-in Express project and setup project. It also adds an appropriate Add-in Express Designer to the Add-in Express project (see Add-in Express Designers).

Add-in Express Designers are the core components of Add-in Express .NET. They allow adding other Add-in Express components and setting their properties at design-time as well as at run-time (see Add-in Express Components).

Add-in Express Designers

There are several Add-in Express Designer types responsible for common tasks in customizing Office in .NET. Below we list the Office automation tasks. Each task includes an introductory description, the Add-in Express Designer type available for the task, and Add-in Express Components available.

- COM Add-ins - ADXAddinModule
- RTD Servers - ADXRtdServerModule
- Smart Tags - ADXSmartTagModule
- Excel Automation Add-ins - ADXExcelAddinModule
- Excel Workbooks - ADXExcelSheetModule
- Word Documents - ADXWordDocumentModule

COM Add-ins

COM Add-ins have been around since Office 2000 when Microsoft allowed Office applications to extend their features with COM DLLs supporting the IDTExtensibility2 interface (it is a COM interface, of course). Since then thousands of developers have racked their brains over this interface and the Office Object Model that provided COM objects representing command bars, command bar controls, etc. These were the sources of Add-in Express.

ADXAddinModule represents a COM Add-in in any Office application. To add another add-in to your assembly, add another ADXAddinModule to your project. For the add-in, you specify its name, host application(s) and load behavior. The typical value for the LoadBehavior property is Connected & LoadAtStartup. For Outlook add-ins, you specify the Options page and Folder Property pages (see Outlook Property Page). See the following chapters for the Add-in Express components you add onto the ADXAddinModule: Office 2007 Ribbon Components, Command Bars, Command Bar Controls, Built-in Control Connector, Keyboard Shortcut, Outlook Bar Shortcut Manager, Outlook Forms Manager, and Application-level Events.

Use the AddinStartupComplete and AddinBeginShutdown events to handle add-in startup and shutdown.

RTD Servers

RTD Server is a technology introduced in Excel XP. It is a great way to display constantly changing data such as stock quotes, currency exchange rates, inventory levels, price quotes, weather information, sports scores, and so on.

A short terminology list follows below:

- An RTD server is a Component Object Model (COM) Automation server that implements the IRtdServer interface. Excel uses the RTD server to communicate with a real-time data source on one or more topics.
- A real-time data source is any source of data that you can access programmatically.
- A topic is a string (or a set of strings) that uniquely identifies a piece of data that resides in a real-time data source. The RTD server passes the topic to the real-time data source and receives the value of the topic from the real-time data source; the RTD server then passes the value of the topic to Excel for display. For example, the RTD server passes the topic "New Topic" to the real-time data source, and the RTD server receives the topic's value of "72.12" from the real-time data source. The RTD server then passes the topic's value to Excel for display.

ADXRtdServerModule represents an RTD Server. The only Add-in Express component allowed for this designer is the RTD Topic. The module provides the Interval property that indicates the time interval between updates (in milliseconds).

You refer to an existing RTD Server using the RTD worksheet function in Excel:

```
   =RTD(ProgID, Server, String1, String2, ... String28)
```

The ProgID parameter is a required string value representing the programmatic ID (ProgID) of the RTD server. See attributes of the RTDServerModule class for the ProgId of your RTD Server. The current version of Add-in Express requires the Server parameter to be an empty string. Use two quotation marks (""). The String1 through String28 parameters represent topics of the RTD server. Only the String1 parameter is required; the String2 through String28 parameters are optional. In most cases, the String1 parameter will be enough for you. The actual values for the String1 through String28 parameters depend on the requirements of the real-time data server.

Smart Tags

Office XP bestowed Smart Tags upon us in Word and Excel. Office 2003 added PowerPoint to the list of smart tag host applications. This technology provides Office users with more interactivity for the content of their Office documents. A smart tag is an element of text in an Office document having custom actions associated with it. Smart tags allow recognizing such text using either a dictionary-based or a custom-processing approach. An example of such text might be an e-mail address you type into a Word document or an Excel workbook. When smart tag recognizes the e-mail address, it allows the user to choose one of the actions associated with the text. For e-mail addresses, possible actions are to look up additional contact information or send a new e-mail message to that contact.

ADXSmartTagModule lies at the base of the Add-in Express Smart Tags. It represents a set or a library of smart tag recognizers in Excel, Word, and PowerPoint. The only Add-in Express component you add to the designer is Smart Tag.

Excel Automation Add-ins

Excel 2002 brought in Automation Add-ins – a technology that allows writing user-defined functions for use in Excel formulas. Add-in Express .NET provides you with a specialized module, COM Excel Add-in Module, that reduces this task to just writing one ore more user-defined functions. A typical function accepts one or more Excel ranges and/or other parameters. Excel shows the resulting value of the function in the cell where the user calls the function.

Add-in Express .NET provides developing Excel Automation Add-ins using the COM Excel Add-in Module in COM Add-in projects. You add this module to your COM Add-in project by choosing the COM Excel Add-in Module item in the Add New Item Dialog. This adds an ADXExcelAddinModule to the project. The module represents an Excel Automation add-in in Add-in Express. It does not provide any properties. Also, there are no Add-in Express components to use with the module.

Excel Workbooks

Sometimes you need to automate a given Excel workbook (template). You can do it with ADXExcelSheetModule that represents one worksheet of the workbook. The Document property allows creating and browsing for the workbook. If you choose creating a new workbook, the dialog appears where you specify the name and location of the workbook as well as the Property Name and Property Value textboxes. Add-in Express adds this property to the list of custom properties of the workbook and uses the name and value of the property in order to recognize the workbook. Accordingly, you specify the PropertyId and PropertyValue properties of the module. The module provides a full set of events available for Excel workbook.

For the Add-in Express components available for the module see the following chapters: Command Bars, Command Bar Controls, Built-in Control Connector, and Application-level Events.

Note

We regard ADXExcelSheetModule and ADXWordDocumentModule as deprecated and do not recommend using them.

Word Documents

To automate a given Word document, you use the ADXWordDocumentModule. The module allows creating and browsing for the document. If you choose creating a new document, the dialog appears where you specify the name and location of the document as well as the Property Name and Property Value textboxes. Add-in Express adds this property to the list of custom properties of the document and uses the name and value of the property in order to recognize the document. Accordingly, you specify the PropertyId and PropertyValue properties of the module. The module provides a full set of events available for Word document.

For the Add-in Express components available for the module see the following chapters: Command Bars, Command Bar Controls, Built-in Control Connector, and Application-level Events.

Note

We regard ADXExcelSheetModule and ADXWordDocumentModule as deprecated and do not recommend using them.

Add-in Express Components

How to Add an Add-in Express Component to Add-in Express Designer

To add any Add-in Express .NET component onto an Add-in Express Module, activate the Add-in Express Module designer window and use commands available either in the Properties Window or in the context menu. To activate the Add-in Express Module designer window, in the Solution Explorer window, right-click the Add-in Express Module item and choose the View Designer popup menu item.

Office 2007 Ribbon Components

Office 2007 presented a new Ribbon user interface. Microsoft states that the interface makes it easier and quicker for users to get the results they want. The developers extend this interface by using the XML markup that the COM add-in should return to the host through the appropriate interface.

Add-in Express provides some 20 Ribbon-related components to give you the full power of the Ribbon UI customization features.

You start with ADXRibbonTab, ADXRibbonOfficeMenu and ADXRibbonQuickAccessToolbar that get the task of creating the markup upon them. You add controls to a tab or menu using the convenient tree-view-like editor that allows you to see all the items of the tab or menu at a glance. Please, note Microsoft require developers to use the StartFromScratch parameter (see the StartFromScratch property of AddinModule) when customizing Quick Access Toolbar.

ADXRibbonTab and ADXRibbonOfficeMenu support all types of Ribbon controls including regular and button groups; regular, edit, combo and check boxes; buttons and split buttons; labels and dropdown lists; galleries and menus; separators and dialog launchers.

Also, note, to use pre-Office2007 command bars in the Office 2007 add-ins, you must explicitly set to True the UseForRibbon property of the appropriate command bar components. In this case, your toolbars are added to the Add-ins ribbon tab.

Office 2007 Custom Task Panes

To allow further customization of its applications, Office 2007 provides custom task panes. Add-in Express supports task panes by equipping the COM Add-in module with the TaskPanes property. Add an instance of UserControl to your project, add an item to the TaskPanes collection, and set up the item by choosing the control in the ControlProgId property and filling in the Title property. Add your reaction to the TaskPaneXXX event series of the COM Add-in module and the DockPositionStateChange and VisibleStateChange events of the task pane.

Command Bars

Microsoft Office 2000-2003 supplied us with a common term for Office toolbars, menus, and context menus. This term is Command Bar. While look-n-feel of all Office command bars is the same, Outlook command bars are different from command bars of other Office applications. They are different for the two main Outlook window types – for Outlook Explorer and Outlook Inspector windows. Additionally, different Outlook Inspector window types show different command bars. Accordingly, Add-in Express provides you with ADXCommandBar, ADXOlExplorerCommandBar, ad ADXOlInspectorCommandBar components.

To add a command bar to your project, use one of the following ADXAddinModule commands:

- Add CommandBar
- Add ExplorerCommandBar - Outlook-specific
- Add InspectorCommandBar - Outlook-specific

The main property of any CommandBar component is CommandBarName. If its value is not equal to the name of any built-in command bar of the host application, then you are creating a new toolbar. If its value is equal to any built-in command bar of the host application, then you are connecting to a built-in command bar. To find out the built-in command bar names, use the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php).

To position your command bar, use the Position, Left, Top, and RowIndex properties.

To speed up add-in loading when connecting to an existing command bar, set the Temporary property to False. To make the host application remove the command bar when the host application quits, set the Temporary property to True.

Outlook-specific toolbars provide the FolderName, FolderNames, and ItemTypes properties that add context-sensitive features to the toolbar.

Command Bars in Office 2007

To see a pre-Office2007 command bar in the Ribbon UI, set the UseForRibbon property of the appropriate Add-in Express command bar component to true.

Command Bar Controls

The Office Object Model (OOM) includes the following command bar controls CommandBarButton, CommandBarComboBox, and CommandBarPopup. Using the correct property settings of the CommandBarComboBox component, you can extend the list with edits and dropdowns. Nevertheless, this list is extremely short. Add-in Express .NET allows extending this list with any control of your choice using the Add-in Express Extensions for Microsoft Office Toolbars, which is a plug-in for Add-in Express.

What follows below is a list of controls available in the ADXCommandBarControl Collection Editor dialog (it opens when you edit the Controls collection of a command bar):

- ADXCommandBarButton
- ADXCommandBarComboBox
- ADXCommandBarEdit
- ADXCommandBarPopup
- ADXCommandBarDropDownList
- ADXCommandBarControl (you use this item to add built-in controls to your command bars)
- ADXCommandBarAdvancedControl (you use this item with Add-in Express Extensions for Microsoft Office Toolbars, http://www.add-in-express.com/office-toolbar-controls/)

Please, note that due to the nature of command bars (remember a 'command bar' stands for toolbar, menu, and context menu), [context] menu items can be buttons, combo boxes, and pop-ups.

Populate your command bar using the Controls collection both at run-time and at design-time. Every control (built-in and custom) added to this collection will be added to the corresponding toolbar at your project startup.

The main property of any command bar control is the Id property. To add a built-in control to your toolbar, specify its Id in the Id property of the command bar control. To find out the Ids of every built-in control in the host application, use the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php). To add a custom control to the toolbar, leave the Id property unchanged.

Set up a control's appearance using a great number of its properties, such as Enabled and Visible, Style and State, Caption and ToolTipText, DropDownLines and DropDownWidth, etc. You also control the size (Top, Left, Height, Width) and location (Before, AfterId, and BeforeId) properties. To provide your command bar buttons with a default list of icons, drop an ImageList component to the Add-in Express Module and specify the ImageList in the Images property of the Add-in Express Module. Don't forget to set the button's Style property to either adxMsoButtonIconAndCaption or adxMsoButtonIcon. Use the DisableStandardAction property available for command bar buttons.

Use the OlExplorerItemTypes, OlInspectorItemTypes, and OlItemTypesAction properties to add context-sensitivity to controls on Outlook-specific command bars. The OlItemTypesAction property specifies an action that Add-in Express will perform on the control when the current item's type coincides with that specified by you. Use the Click event for Button and the Change event for Edit, ComboBox and DropDownList controls.

Built-in Control Connector

Built-in controls of an Office application have predefined IDs. You find the IDs using the free Built-in Controls Scanner utility (http://www.add-in-express.com/downloads/controls-scanner.php).

The Built-in Control Connector component allows overriding the standard action for any built-in control without the necessity to add it onto any command bar.

Add a BuiltinControlConnector onto ADXAddinModule. Set its Id property to the command bar control ID from your host application. To connect the component to the command bar control, leave its CommandBar property empty. To connect the component to the control on a given toolbar, specify the toolbar in the CommandBar property. To override the default action of the control, use the Action event. The component traces the context and when the context changes, it reconnects to the currently active instance of the command bar control with the given Id taking away this task from you.

Keyboard Shortcut

Every Office application provides built-in keyboard combinations that allow shortening the access path for commands, features, and options of the application. Add-in Express allows adding custom keyboard combinations and processing both custom and built-in ones.

Add the component onto ADXAddinModule, choose the keyboard shortcut you need in the ShortcutText property, set the HandleShortCuts property of the Add-in Express Module to true and process the Action event of the KeyboardShortcut component.

Outlook Bar Shortcut Manager

Outlook provides us with the Outlook Bar (Navigation Pane in Outlook 2003). The Outlook Bar displays Shortcut groups consisting of Shortcuts that you can target to a Microsoft Outlook folder, a file-system folder, or a file-system path or URL. You use the Outlook Bar Shortcut Manager to customize the Outlook Bar with your shortcuts and groups.

This component is available for ADXAddinModule. Use the Groups collection of the component to create a new shortcut group. Use the Shortcuts collection of a short group to create a new shortcut. To connect to an existing shortcut or shortcut group, set the Caption properties of the corresponding ADXOlBarShortcut and/or ADXOlBarGroup components equal to the caption of the existing shortcut or shortcut group. Please note, there are no other ways to identify the group or shortcut.

That is why your shortcuts and shortcut groups must be named uniquely for Add-in Express to remove them (and not those with the same names) when the add-in is uninstalled. That is why you have to do this yourself. Depending on the type of its value, the Target property of the ADXOlBarShortcut component allows you to specify different shortcut types. If the type is MAPIFolder, the shortcut represents a Microsoft Outlook folder. If the type is a String, the shortcut represents a file-system path or a URL. No events, thanks to MS.

Outlook Forms Manager

Customizing Outlook has a long-dated history. To customize Outlook forms you can use the built-in tools providing for designing and publishing the form. You can use HTML and VBScript to customize folder views, and Outlook Today is an example of this approach. Now you can embed custom .NET forms into the Outlook Explorer and Inspector windows and replace folder views with custom .NET forms.

The Outlook Forms Manager component is available for ADXAddinModule only. It is the core component of the Add-in Express Extensions .NET for Microsoft Outlook which is a plug-in for Add-in Express .NET. You find the detailed information on this product at http://www.add-in-express.com/outlook-extension/.

Outlook Property Page

Outlook allows extending its Options dialog with custom pages. You see this dialog when you choose Tools | Options menu. In addition, Outlook allows adding such a page to the Folder Properties dialog. You see this dialog when you choose the Properties item in the folder context menu. You create such pages using the Outlook Property Page component.

In the Add New Item Dialog, choose the Outlook Options Page item to add a class to your project. This class is a descendant of the System.Windows.Forms.UserControl class. It allows creating Outlook property pages using its visual designer. Just set up the property page properties, place your controls onto the page, and add your code. To add this page to the Outlook Options dialog, select the name of your control class in the PageType combo of ADXAddinModule and enter some characters into the PageTitle property.

To add a page to the Folder Properties dialog for a given folder(s), you use the FolderPages collection of the Add-in Express Module. Run its property editor and add an item (of the ADXOlFolderPage type). You connect the item to a given property page through the PageType property. Note, the FolderName, FolderNames, and ItemTypes properties of the ADXOlFolderPage component work in the same way as those of Outlook-specific command-bars.

Specify reactions required by your business logics in the Apply and Dirty event handlers. Use the OnStatusChange method to raise the Dirty event, the parameters of which allow marking the page as Dirty.

Smart Tag

The Kind property of the ADXSmartTag component allows you to choose one of two text recognition strategies: either using a list of words in the RecognizedWords string collection or implementing a custom recognition process based on the Recognize event of the component. Use the ActionNeeded event to change the Actions collection according to the current context. The component raises the PropertyPage event when the user clicks the Property button in the Smart Tags tab (Tools / AutoCorrect Options menu) for your smart tag.

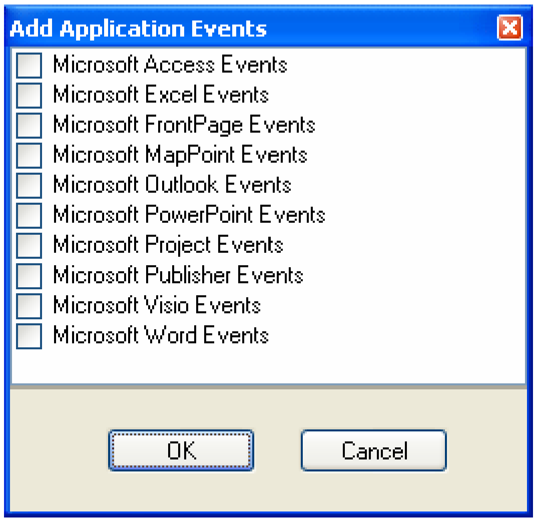
RTD Topic

Use the String## properties to identify the topic of your RTD server. To handle RTD Server startup situations nicely, specify the default value for the topic and, using the UseStoredValue property, specify, if the RTD function in Excel returns the default value (UseStoredValue = false) or doesn't change the displayed value

(UseStoredValue = true). The RTD Topic component provides you with the Connect, Disconnect, and RefreshData events. The last one occurs (for enabled topics only) whenever Excel calls the RTD function.

Application-level Events

ADXAddinModule provides events for all Office applications through the Add Events command that shows the following dialog:


Add Application Events

- [ ] Microsoft Access Events
- [ ] Microsoft Excel Events
- [ ] Microsoft FrontPage Events
- [ ] Microsoft MapPoint Events
- [ ] Microsoft Outlook Events
- [ ] Microsoft PowerPoint Events
- [ ] Microsoft Project Events
- [ ] Microsoft Publisher Events
- [ ] Microsoft Visio Events
- [ ] Microsoft Word Events

Select the application you need and click OK. This adds and/or removes appropriate Add-in Express event components to the module. Use the event handlers of an Add-in Express event component to respond to the host application's events. See also Add-in Express Event Classes.

Add-in Express Event Classes

Outlook and Excel differ from other Office applications because they have event-raising objects not only at the topmost level of their object models. These exceptions are Worksheet in Excel, and Folders, Items and all Item sorts in Outlook. Naturally, you need to handle events from these sources. Add-in Express Event Classes provide you with components that ease the pain of handling such events. Add-in Express Event classes are host-version independent. The Add-in Express event classes also handle releasing of COM objects required for their functioning.

At design-time, you add an Add-in Express event class to the project (see Add New Item Dialog) and use its event procedures to write the code for just one set of event handling rules for a given event source type (say, Items collection of an Outlook folder). To implement another set of event handling rules for the same event source type, you add another Add-in Express event class to your project.

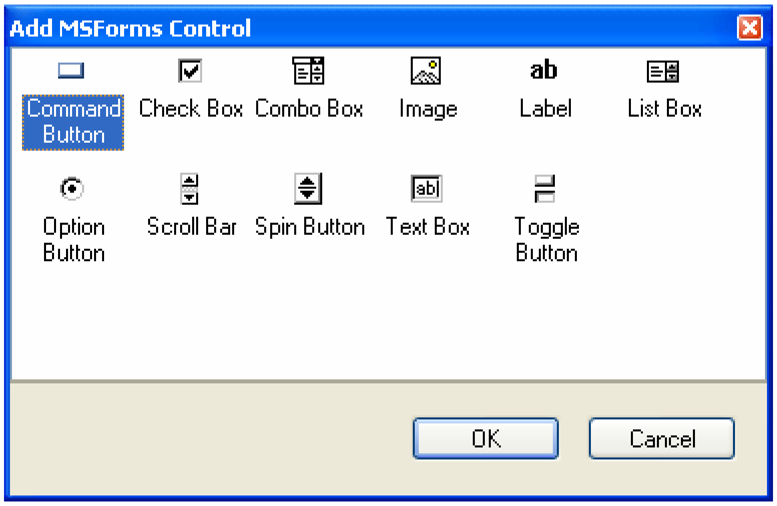
At run-time, you connect an Add-in Express event class instance to an event source using its ConnectTo method. To disconnect the Add-in Express event class from the event source you use the RemoveConnection method. To apply the same business rules to another event source of the same type (say, to items of another folder), you create a new instance of the same event class.

What follow below is the source of a newly added Event Class that processes the events of the Items collection of the MAPIFolder class in Outlook.

```
Imports System

'Add-in Express Outlook Items Events Class
Public Class OutlookItemsEventsClass1
   Inherits AddinExpress.MSO.ADXOutlookItemsEvents

   Public Sub New(ByVal ADXModule As AddinExpress.MSO.ADXAddinModule)
      MyBase.New(ADXModule)
   End Sub

   Public Overrides Sub ProcessItemAdd(ByVal Item As Object)
      'TODO: Add some code
   End Sub

   Public Overrides Sub ProcessItemChange(ByVal Item As Object)
      'TODO: Add some code
   End Sub

   Public Overrides Sub ProcessItemRemove()
      'TODO: Add some code
   End Sub
End Class
```

MSForms Control

This command is available for ADXExcelSheetModule and ADXWordDocumentModule. When run, it displays the following dialog:



Select the control you need to connect to and click OK. Add-in Express adds an appropriate MS Forms Control Connector to the module. Use the Control Name property of the connector to specify the underlying control on the Excel worksheet or Word document. Respond to the events provided by the control connector.

Note

We regard ADXExcelSheetModule and ADXWordDocumentModule as deprecated and do not recommend using them.

Your First Microsoft Office COM Add-in

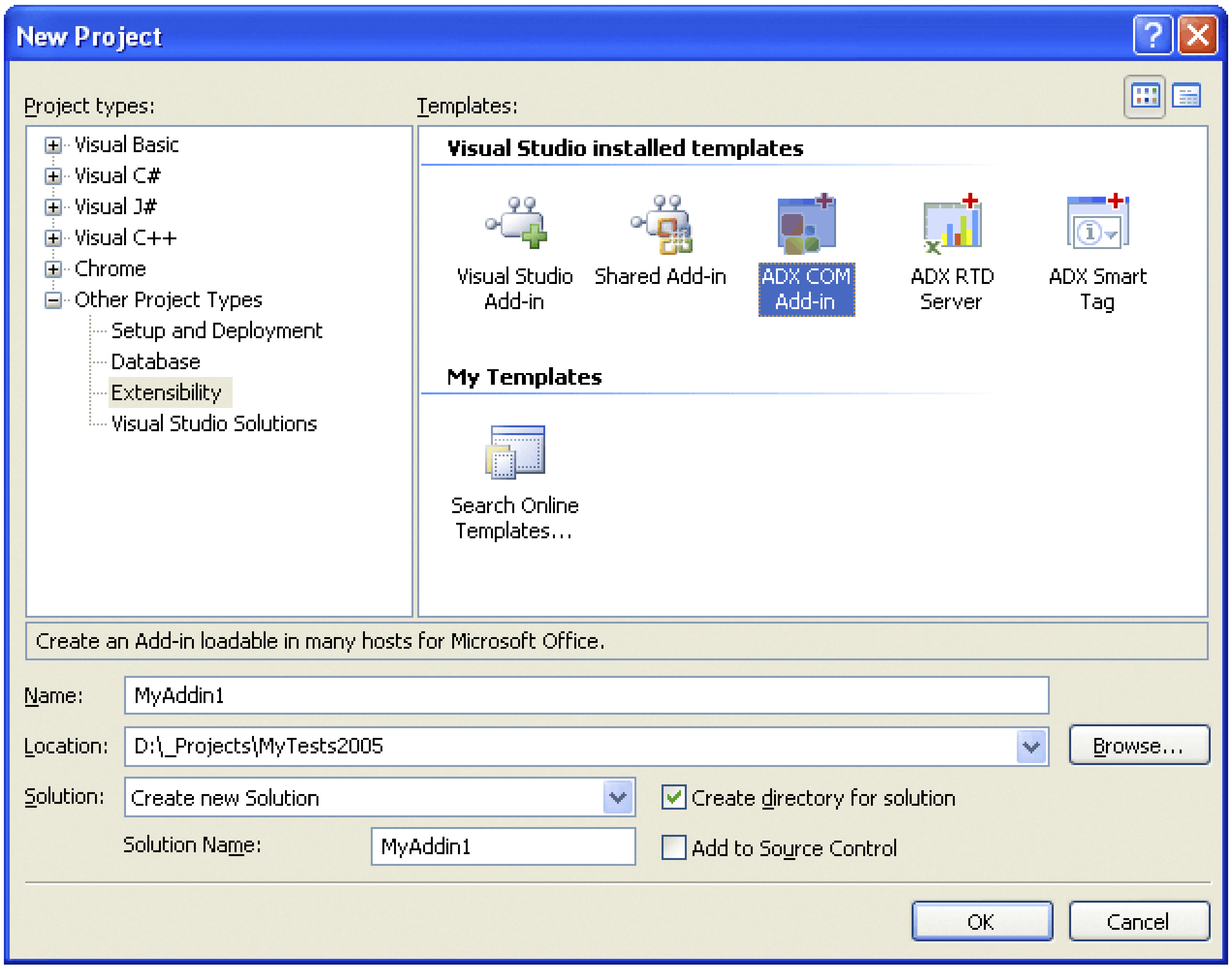
This chapter highlights almost every aspect of creating COM Add-ins for Microsoft Office applications. You find this sample in the Demo Projects folder of the Add-in Express .NET install folder.

Outlook and Add-in Express

Please note, Add-in Express provides additional components for COM Add-ins in Outlook. See Your First Microsoft Outlook COM Add-in.

Step #1 - Creating an Add-in Express COM Add-in Project

Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.



Create an Add-in loadable in many hosts for Microsoft Office.

Name: MyAddin1

Location: D:_Projects\MyTests2005

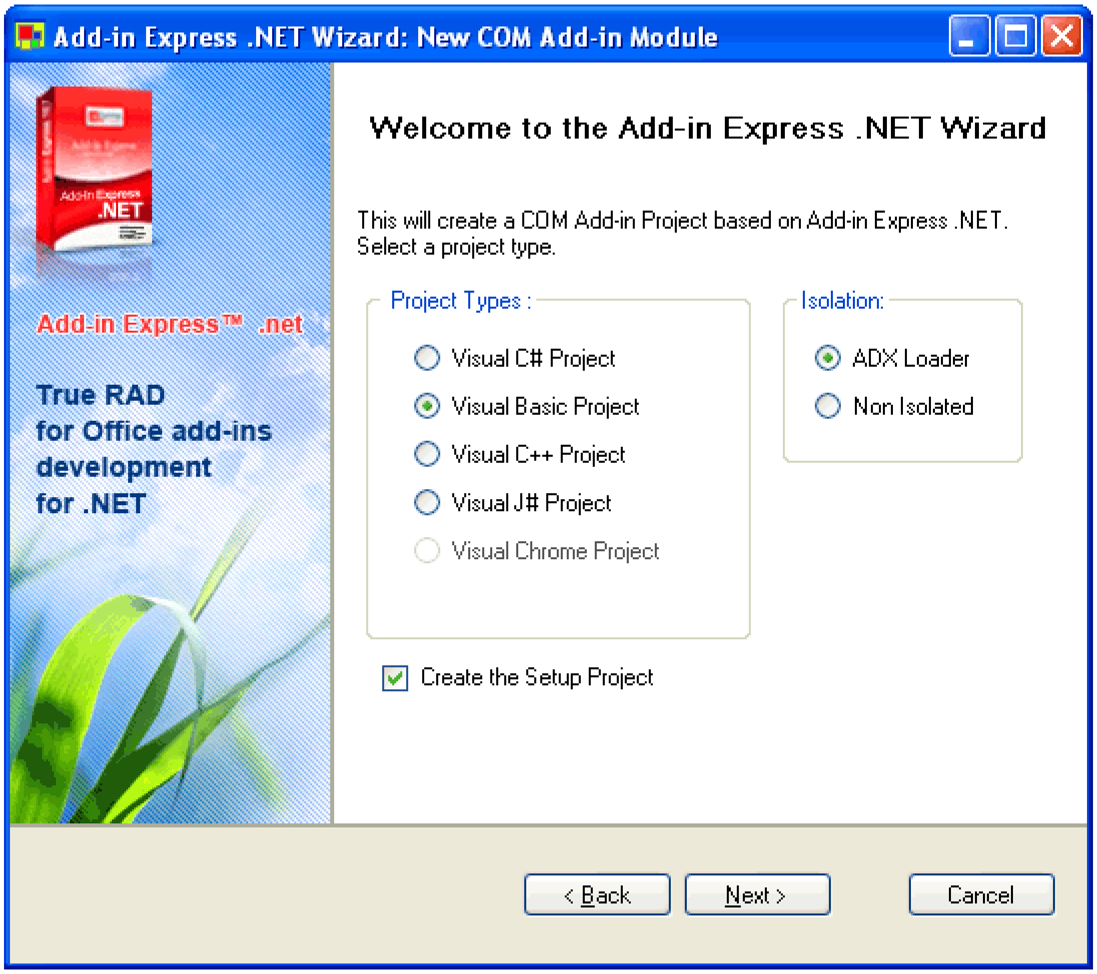
Solution: Create new Solution

Solution Name: MyAddin1

[✓] Create directory for solution

[ ] Add to Source Control

When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, and supported applications of your add-in.


Add-in Express .NET Wizard: New COM Add-in Module

Welcome to the Add-in Express .NET Wizard

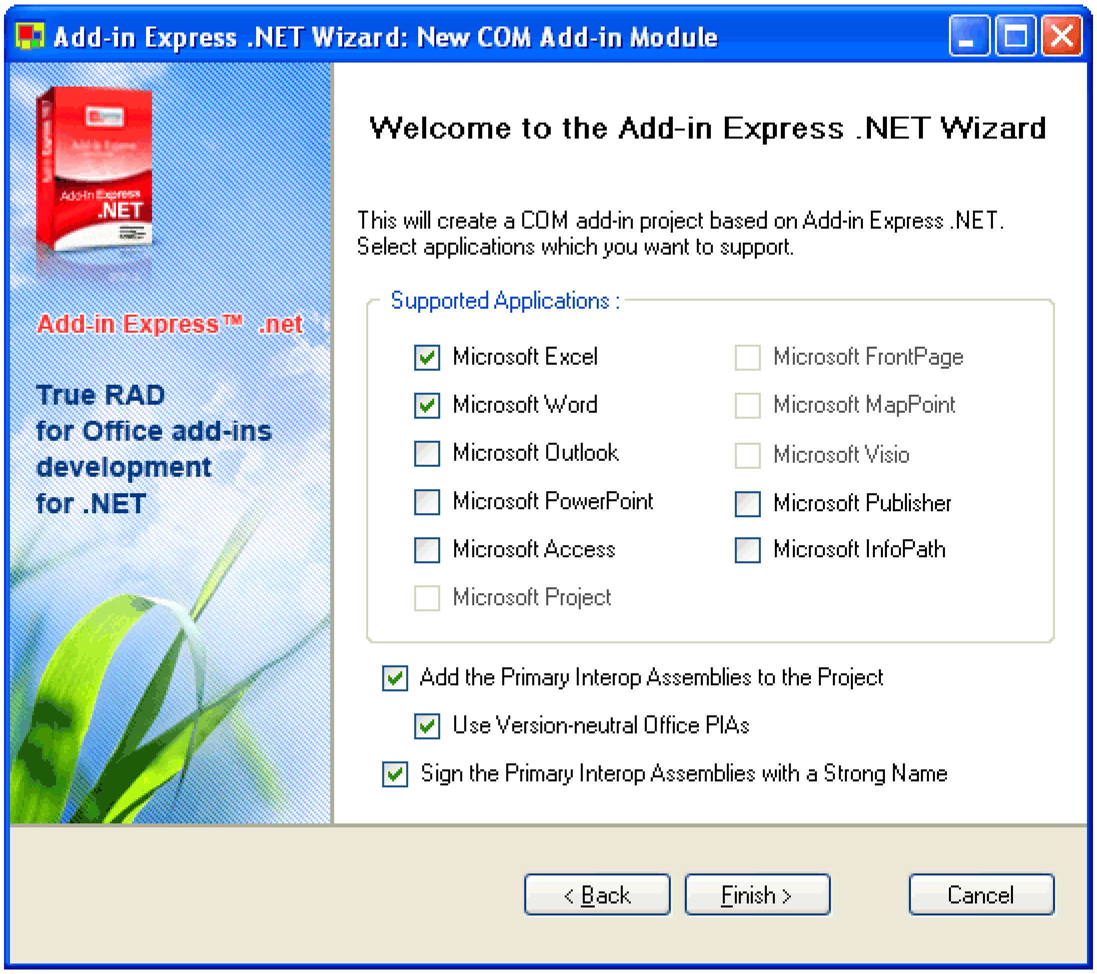
This will create a COM Add-in Project based on Add-in Express .NET.
Select a project type.

Project Types :
( ) Visual C# Project
(●) Visual Basic Project
( ) Visual C++ Project
( ) Visual J# Project
( ) Visual Chrome Project

Isolation:
(●) ADX Loader
( ) Non Isolated

[✓] Create the Setup Project

This VB.NET sample shows an Add-in Express COM Add-in project implementing a COM add-in for Excel and Word with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.



Add-in Express .NET Wizard: New COM Add-in Module

Welcome to the Add-in Express .NET Wizard

This will create a COM add-in project based on Add-in Express .NET.
Select applications which you want to support.

Supported Applications :

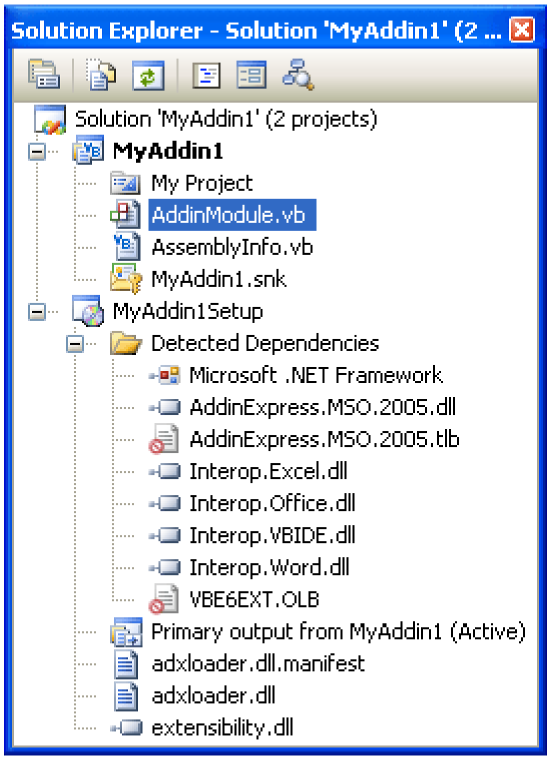
- [✓] Microsoft Excel
- [✓] Microsoft Word
- [ ] Microsoft Outlook
- [ ] Microsoft PowerPoint
- [ ] Microsoft Access
- [ ] Microsoft Project
- [ ] Microsoft FrontPage
- [ ] Microsoft MapPoint
- [ ] Microsoft Visio
- [ ] Microsoft Publisher
- [ ] Microsoft InfoPath

- [✓] Add the Primary Interop Assemblies to the Project
- [✓] Use Version-neutral Office PIAs
- [✓] Sign the Primary Interop Assemblies with a Strong Name

The Add-in Express Project Wizard creates and opens the COM Add-in solution in the IDE. The solution includes the COM Add-in project and the setup project.

Your add-ins are version-neutral

All previous versions of the core Add-in Express were version-neutral. Now we are ready to represent the version-neutral programming model for all our customers who develop extensions for Office 2000+. Add-in Express 2007 delivers version-neutral Office interop assemblies. To use them, simply check the Use Version-neutral Office PIAs check box when creating your projects.

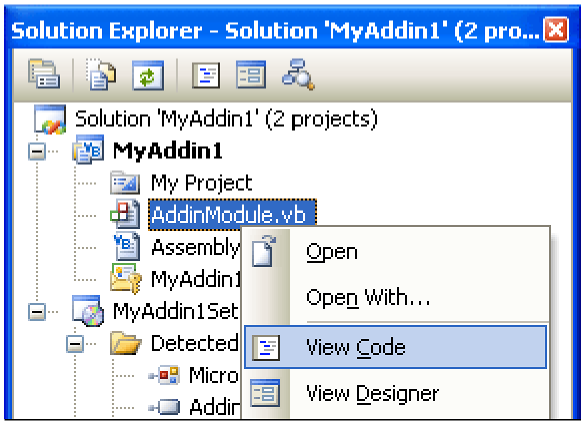

Solution Explorer - Solution 'MyAddin1' (2 ...

Solution 'MyAddin1' (2 projects)
MyAddin1
My Project
AddinModule.vb
AssemblyInfo.vb
MyAddin1.snk
MyAddin1Setup
Detected Dependencies
Microsoft .NET Framework
AddinExpress.MSO.2005.dll
AddinExpress.MSO.2005.tlb
Interop.Excel.dll
Interop.Office.dll
Interop.VBIDE.dll
Interop.Word.dll
VBE6EXT.OLB
Primary output from MyAddin1 (Active)
adxloader.dll.manifest
adxloader.dll
extensibility.dll

The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in the next step.

Step #2 - Add-in Express COM Add-in Module

The AddinModule.vb (or AddinModule1.cs) is a COM Add-in Module that is the core part of the COM add-in project (see COM Add-ins). It is the placeholder of the Add-in Express components, which allow you to concentrate on the functionality of your add-in. You specify the add-in properties in the module's properties, add the Add-in Express components to the module's designer, and write the functional code of your add-in in this module. To review its source code, in the Solution Explorer window, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.



The code for AddinModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Add-in Module
<GuidAttribute("AB07BADE-56F7-414B-ACA0-D3E2DDAABCCB"), _
    ProgIdAttribute("MyAddin1.AddinModule")> _
Public Class AddinModule
    Inherits AddinExpress.MSO.ADXAddinModule

#Region " Component Designer generated code. "
   'Required by designer
   Private components As System.ComponentModel.IContainer

   'Required by designer - do not modify
   'the following method
   Private Sub InitializeComponent()
      Me.components = New System.ComponentModel.Container
      '
      'AddinModule
      '
      Me.AddinName = "MyAddin1"
      Me.SupportedApps = CType((AddinExpress.MSO.ADXOfficeHostApp.ohaExcel _
          Or AddinExpress.MSO.ADXOfficeHostApp.ohaWord), _
          AddinExpress.MSO.ADXOfficeHostApp)
   End Sub
#End Region

#Region " Add-in Express automatic code "

    'Required by Add-in Express - do not modify
    'the methods within this region

    Public Overrides Function GetContainer() As _
        System.ComponentModel.IContainer
        If components Is Nothing Then
            components = New System.ComponentModel.Container
        End If
        GetContainer = components
    End Function

    <ComRegisterFunctionAttribute()> _
    Public Shared Sub AddinRegister(ByVal t As Type)
        AddinExpress.MSO.ADXAddinModule.ADXRegister(t)
    End Sub

    <ComUnregisterFunctionAttribute()> _
```

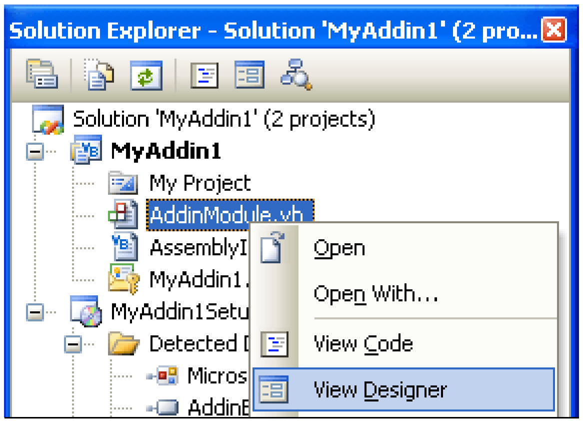
```
        Public Shared Sub AddinUnregister(ByVal t As Type)
            AddinExpress.MSO.ADXAddinModule.ADXUnregister(t)
        End Sub

        Public Overrides Sub UninstallControls()
            MyBase.UninstallControls()
        End Sub

    #End Region

        Public Sub New()
            MyBase.New()

            'This call is required by the Component Designer
            InitializeComponent()

            'Add any initialization after the InitializeComponent() call

        End Sub

        Public ReadOnly Property ExcelApp() As Excel._Application
            Get
                Return CType(HostApplication, Excel._Application)
            End Get
        End Property

        Public ReadOnly Property WordApp() As Word._Application
            Get
                Return CType(HostApplication, Word._Application)
            End Get
        End Property
```

Please, pay attention to the ExcelApp and WordApp properties of the module generated by the Add-in Express Project Wizard. You can use them in your code to get access to the host application objects.

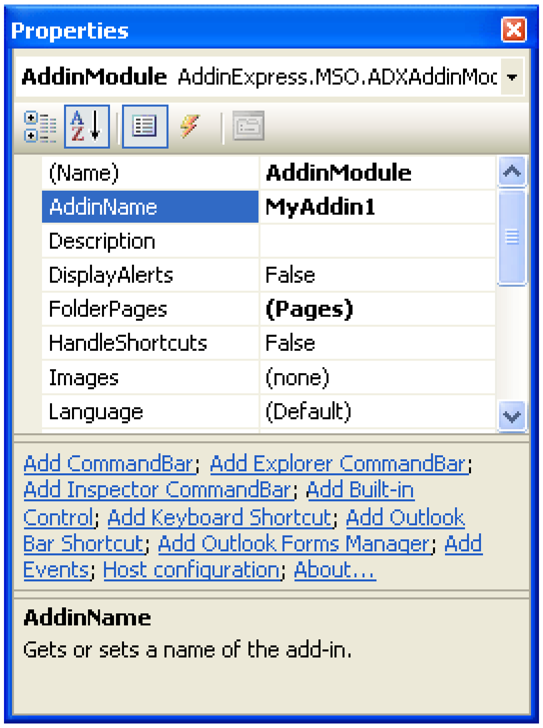
Step #3 – Add-in Express COM Add-in Designer

The Add-in Express COM Add-in Designer allows setting add-in properties and adding components to the module.

In the Solution Explorer window, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.



In the Properties window, you set the name and description of your add-in module (see COM Add-ins).


Properties

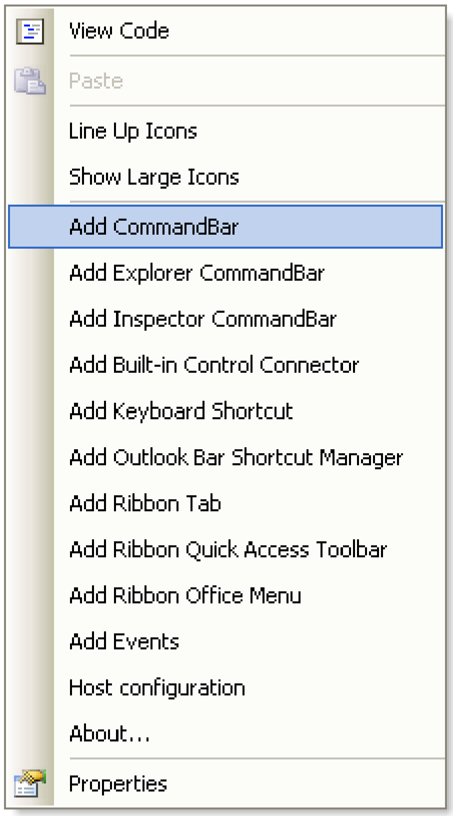
AddinModule AddinExpress.MSO.ADXAddinMod

(Name)	AddinModule
AddinName	MyAddin1
Description	
DisplayAlerts	False
FolderPages	(Pages)
HandleShortcuts	False
Images	(none)
Language	(Default)

Add CommandBar; Add Explorer CommandBar; Add Inspector CommandBar; Add Built-in Control; Add Keyboard Shortcut; Add Outlook Bar Shortcut; Add Outlook Forms Manager; Add Events; Host configuration; About...

AddinName
Gets or sets a name of the add-in.

To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.



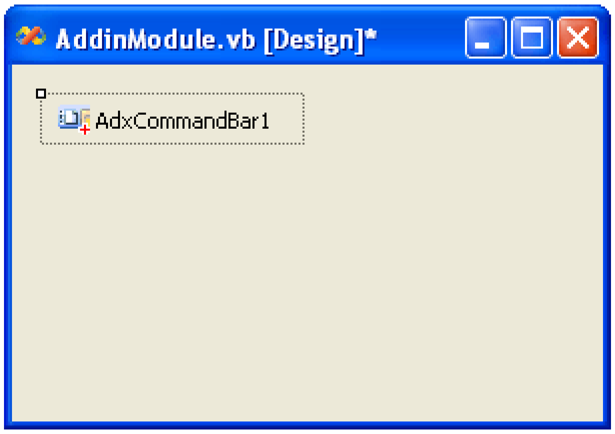
The following commands add the following components to the module:

- Add CommandBar – adds a command bar to your add-in (see Command Bars)
- Add Explorer CommandBar – adds an Outlook Explorer command bar to your add-in (see Command Bars)
- Add Inspector CommandBar – adds an Outlook Inspector command bar to your add-in (see Command Bars)
- Add Built-in Control Connector – adds a component that allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- Add Keyboard Shortcut– adds a component that allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)
- Add Outlook Bar Shortcut Manager – adds a component that allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- Add Outlook Forms Manager – adds a component that allows embedding custom .NET forms into Outlook windows (see Outlook Forms Manager)
- Add Ribbon Tab – adds a Ribbon tab to your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Quick Access Toolbar – adds a component that allows customizing the Ribbon Quick Access Toolbar in your add-in (see Office 2007 Ribbon Components)

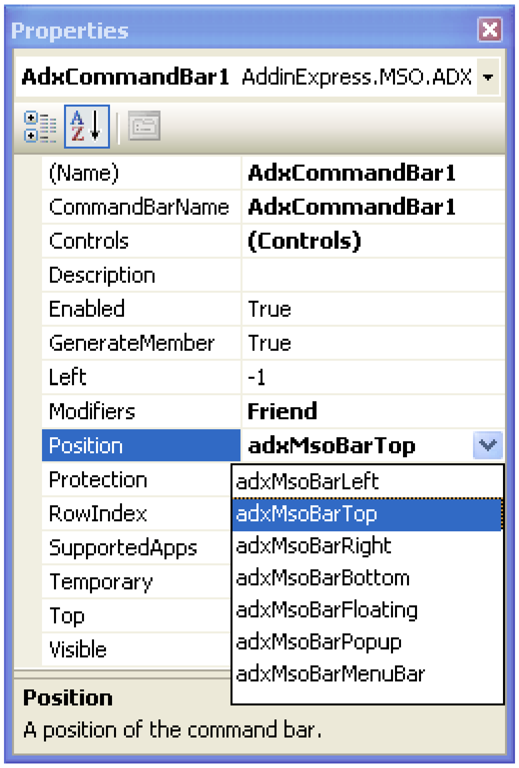
- Add Ribbon Office Menu – adds a component that allows customizing the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)
- Add Events – adds or deletes components that provide access to application-level events of the add-in host applications (see Application-level Events)

Step #4 - Adding a New Command Bar

To add a command bar to your add-in, use the Add CommandBar command that adds an ADXCommandBar component to the COM Add-in Module (see Command Bars).



Select the command bar component and, in the Properties window, specify the command bar name using the CommandBarName property. In addition, you select its position in the Position property.



Properties

AdxCommandBar1 AddinExpress.MSO.ADX

(Name)	AdxCommandBar1
CommandBarName	AdxCommandBar1
Controls	(Controls)
Description	
Enabled	True
GenerateMember	True
Left	-1
Modifiers	Friend
Position	adxMsoBarTop
Protection	
RowIndex	
SupportedApps	
Temporary	
Top	
Visible	

adxMsoBarLeft
adxMsoBarTop
adxMsoBarRight
adxMsoBarBottom
adxMsoBarFloating
adxMsoBarPopup
adxMsoBarMenuBar

Position
A position of the command bar.

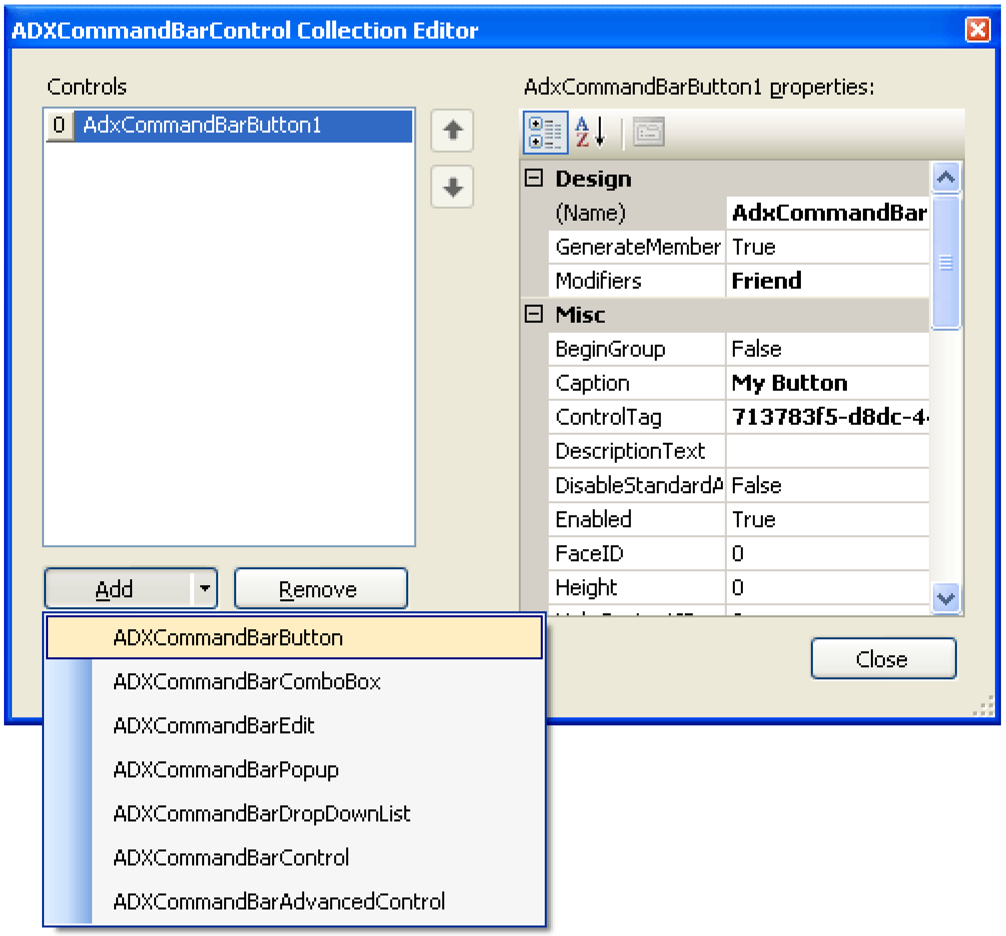
Command Bars in Office 2007

To display the command bar in Office 2007 you must explicitly set the UseForRibbon property of the command bar component to True.

Step #5 - Adding a New Command Bar Button

To add a new button to the command bar, in the Properties window, you select the Controls property of an appropriate command bar component and click the property editor button (the button in the property value field).

In the ADXCommandBarControl Collection Editor, you select the command bar control type in the Add button (see also Command Bar Controls).



AdxCommandBarButton1 properties:

Design	
(Name)	**AdxCommandBar**
GenerateMember	True
Modifiers	**Friend**
Misc	
BeginGroup	False
Caption	**My Button**
ControlTag	**713783f5-d8dc-4**
DescriptionText	
DisableStandardA	False
Enabled	True
FaceID	0
Height	0

Specify the button's Caption property, set the Style property (default value = adxMsoButtonCaption), and close the collection editor. To handle the Click event of the button, select the added button in the topmost combo of the Properties window and add the Click event handler: The code of the event handler follows below (it's empty as you can see)

```
Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
    Handles AdxCommandBarButton1.Click

End Sub
```

Step #6 - Accessing Host Application Objects

The Add-in Module provides the HostApplication property that returns the Application object (of the Object type) of the host application the add-in is currently running in. For your convenience, the Add-in Express Project Wizard adds host-related properties to the Add-in module. You use these properties to access host application objects. For instance, this sample add-in includes the following properties in the Add-in Module:

```
Public ReadOnly Property ExcelApp() As Excel._Application
    Get
        Return CType(HostApplication, Excel._Application)
```

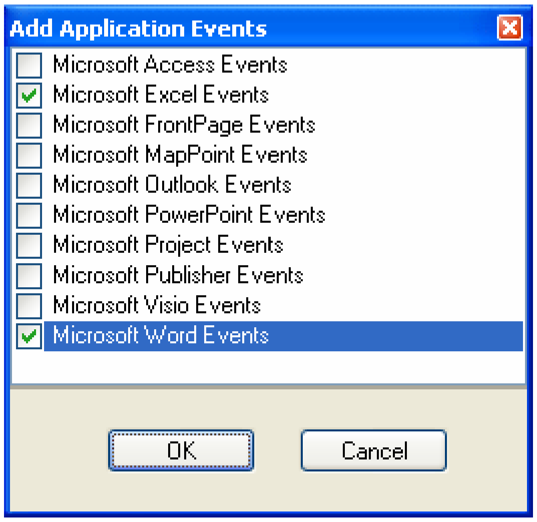
```
            End Get
        End Property

        Public ReadOnly Property WordApp() As Word._Application
            Get
                Return CType(HostApplication, Word._Application)
            End Get
        End Property
```

The _Application object provides the same properties and methods as the Application object but it doesn't provide events.

This allows us to write the following code to the Click event of the button just added.

```
      Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
         Handles AdxCommandBarButton1.Click
         If Me.HostName = "Excel" Then
            MsgBox("The current cell is " + _
               Me.ExcelApp.ActiveCell.AddressLocal(False, False)) 'relative
         ElseIf Me.HostName = "Word" Then
            MsgBox("There are " + _
               Me.WordApp.Selection.Range.Words.Count.ToString() + _
                  " words currently selected")
         Else
            MsgBox(Me.AddinName + " doesn't support " + Me.HostName)
         End If
      End Sub
```

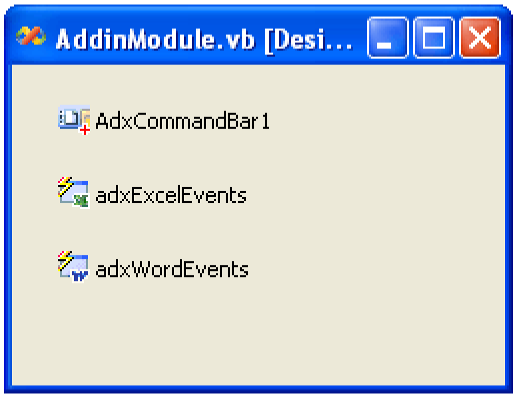
Step #7 - Handling Host Application Events


Add Application Events

- [ ] Microsoft Access Events
- [x] Microsoft Excel Events
- [ ] Microsoft FrontPage Events
- [ ] Microsoft MapPoint Events
- [ ] Microsoft Outlook Events
- [ ] Microsoft PowerPoint Events
- [ ] Microsoft Project Events
- [ ] Microsoft Publisher Events
- [ ] Microsoft Visio Events
- [x] Microsoft Word Events

The COM Add-in Designer provides the Add Events command that adds (and remove) event components that allow handling application-level events. When you choose this command, the Add Application Events dialog box opens allowing you to select the application Events components you need.

This adds selected Events components to the COM Add-in Module (see Application-level Events).



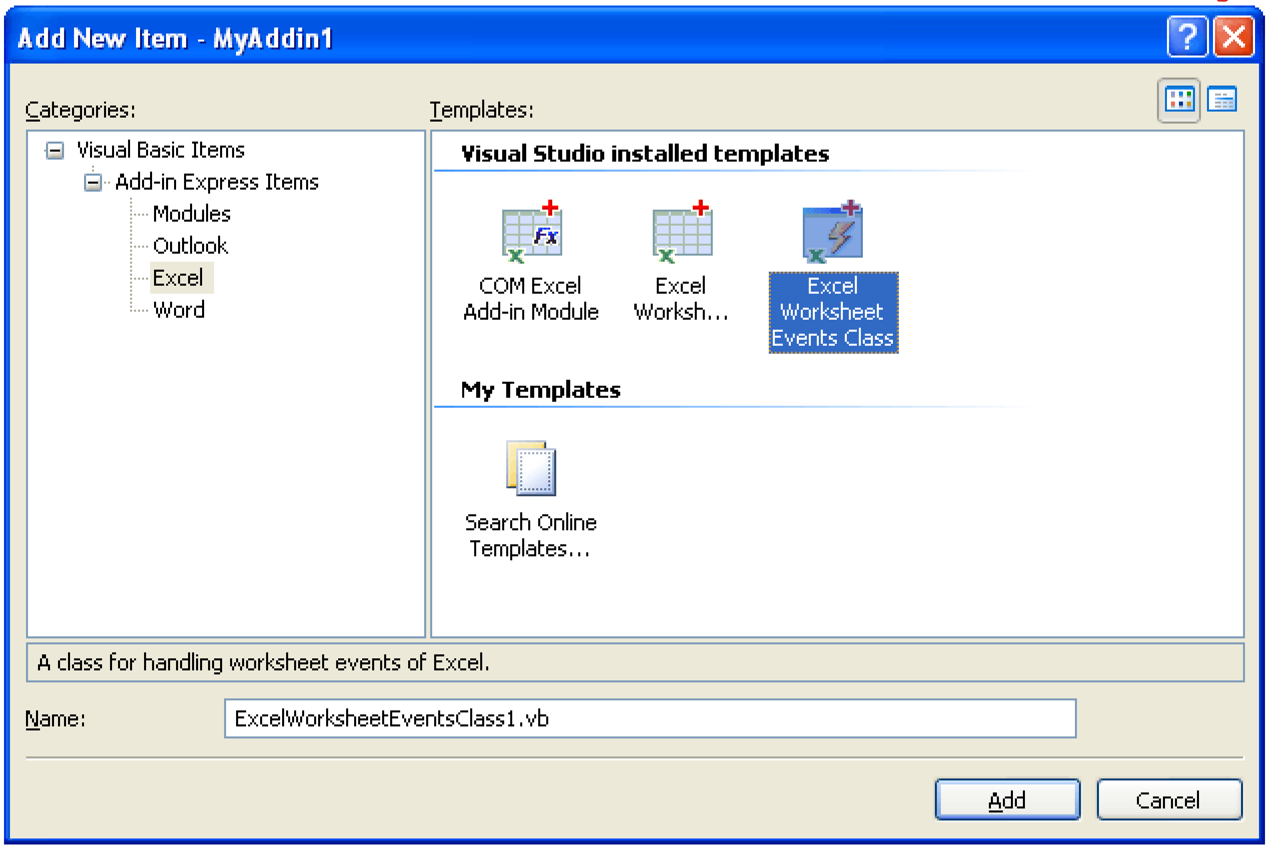
With the Events components, you handle any application-level events of the host application. You might see that the Click event handler in the previous step will fire an exception when there are no workbooks or documents open. To prevent this, you disable the button when a window deactivates and enable it when a window activates. This covers the situations mentioned above.

The code is as follows:

```
      Private Sub Deactivate(ByVal sender As Object, ByVal hostObj As Object, _
         ByVal window As Object) _
         Handles adxWordEvents.WindowDeactivate, _
             adxExcelEvents.WindowDeactivate
         Me.AdxCommandBarButton1.Enabled = False
      End Sub

      Private Sub Activate(ByVal sender As Object, ByVal hostObj As Object, _
         ByVal window As Object) _
         Handles adxWordEvents.WindowActivate, _
             adxExcelEvents.WindowActivate
         Me.AdxCommandBarButton1.Enabled = True
      End Sub
```

Step #8 - Handling Excel Worksheet Events

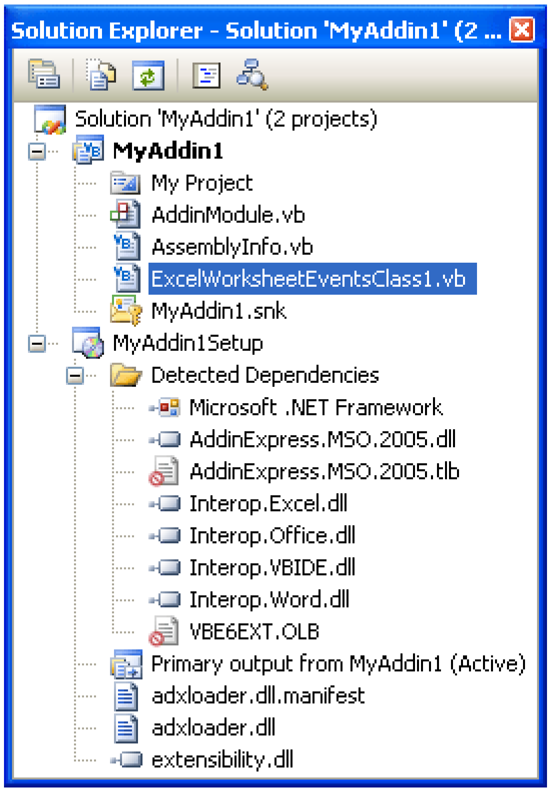
Add-in Express provides the Excel Worksheet event class that allows implementing a set of business rules for an Excel worksheet by handling its events. You add an event class to your project using the Add New Item dialog:



A class for handling worksheet events of Excel.

Name: ExcelWorksheetEventsClass1.vb

This will add an event class to your project (see Add-in Express Event Classes).



```
Solution 'MyAddin1' (2 projects)
  MyAddin1
    My Project
    AddinModule.vb
    AssemblyInfo.vb
    ExcelWorksheetEventsClass1.vb
    MyAddin1.snk
  MyAddin1Setup
    Detected Dependencies
      Microsoft .NET Framework
      AddinExpress.MSO.2005.dll
      AddinExpress.MSO.2005.tlb
      Interop.Excel.dll
      Interop.Office.dll
      Interop.VBIDE.dll
      Interop.Word.dll
      VBE6EXT.OLB
    Primary output from MyAddin1 (Active)
    adxloader.dll.manifest
    adxloader.dll
    extensibility.dll
```

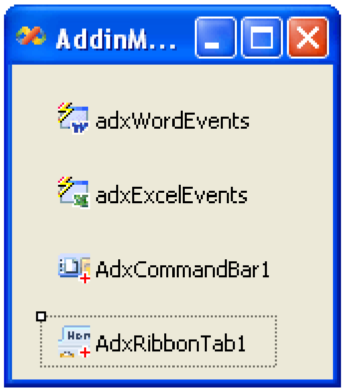
In the code of the event class, you add the following code to the procedure that handles the BeforeRightClick event of the Worksheet class

```
      Public Overrides Sub ProcessBeforeRightClick(ByVal Target As Object, _
         ByVal E As AddinExpress.MSO.ADXCancelEventArgs)
         Dim R As Excel.Range = CType(Target, Excel.Range)
         'Cancel right-clicks for the first column only
         If R.Address(False, False).IndexOf("A") = 0 Then
            MsgBox("Context menu will not be shown!")
            E.Cancel = True
         Else
            E.Cancel = False
         End If
      End Sub
```

Also, you modify the Activate and Deactivate procedures as follows:

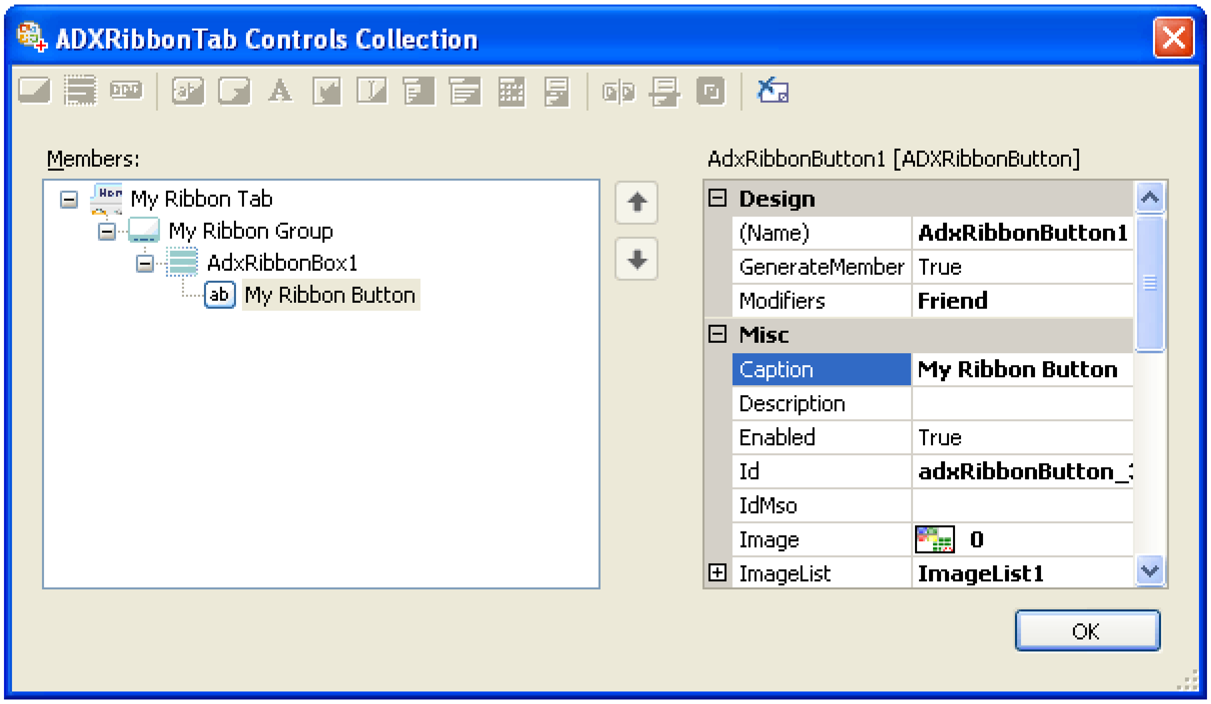
```
      Dim MyEventClass As ExcelWorksheetEventsClass1 = _
          New ExcelWorksheetEventsClass1(Me)
      Dim Sheet As Excel.Worksheet
   ...
      Private Sub Deactivate(ByVal sender As Object, ByVal hostObj As Object, _
         ByVal window As Object) _
         Handles adxWordEvents.WindowDeactivate, _
             adxExcelEvents.WindowDeactivate
         Me.AdxCommandBarButton1.Enabled = False
         Select Case Me.HostName
            Case "Excel"
               If Sheet IsNot Nothing Then
                  MyEventClass.RemoveConnection()
                  Sheet = Nothing
               End If
            Case "Word"
            Case Else
               MsgBox(Me.AddinName + " doesn't support " + Me.HostName)
         End Select
      End Sub

      Private Sub Activate(ByVal sender As Object, ByVal hostObj As Object, _
         ByVal window As Object) _
         Handles adxWordEvents.WindowActivate, _
             adxExcelEvents.WindowActivate
         Me.AdxCommandBarButton1.Enabled = True
         Select Case Me.HostName
            Case "Excel"
               Sheet = Me.ExcelApp.ActiveSheet
               MyEventClass.ConnectTo(Sheet, True)
            Case "Word"
            Case Else
               MsgBox(Me.AddinName + " doesn't support " + Me.HostName)
         End Select
      End Sub
```

Step #9 - Customizing the Office 2007 Ribbon User Interface



To add a new tab to the Ribbon, you use the Add Ribbon Tab command that adds an ADXRibbonTab component to the module.

In the Properties window, run the editor for the Controls collection of the tab. In the editor, use the toolbar buttons or context menu to add or delete Add-in Express components that form the Ribbon interface of your add-in. First, you add a Ribbon tab and change its caption to My Ribbon Tab. Then, you select the tab component, add a Ribbon group, and change its caption to My Ribbon Group. Next, you select the group, and add a button group. Finally, you select the button group and add a button. Set the button caption to My Ribbon Button. Use the ImageList and Image properties to set the icon for the button.



AdxRibbonButton1 [ADXRibbonButton]

Design	
(Name)	**AdxRibbonButton1**
GenerateMember	True
Modifiers	**Friend**
Misc	
Caption	**My Ribbon Button**
Description	
Enabled	True
Id	**adxRibbonButton_**
IdMso	
Image	**0**
ImageList	**ImageList1**

Click OK, and, in the Properties window, find the newly added Ribbon button. Now add the event handler to the Click event of the button. Write the following code:

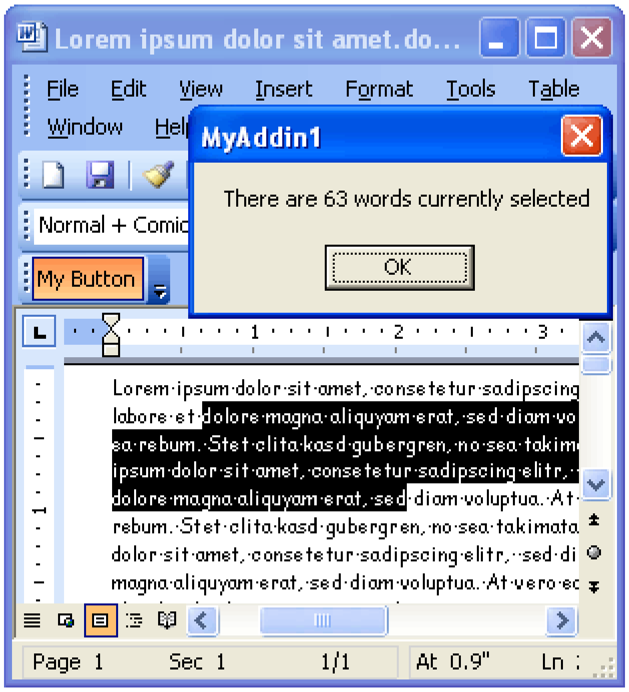
```
        Private Sub AdxRibbonButton1_OnClick(ByVal sender As System.Object, _
            ByVal control As AddinExpress.MSO.IRibbonControl, _
            ByVal pressed As System.Boolean) Handles AdxRibbonButton1.OnClick
            AdxCommandBarButton1_Click(Nothing)
        End Sub
```

Remember, the ADXRibbonTab Controls editor performs the XML-schema validation automatically, so from time to time you will run into the situation when you cannot add a control to some Ribbon level. It is a restriction of the Ribbon XML-schema.

See also Office 2007 Ribbon Components.

Step #10 - Running the COM Add-in

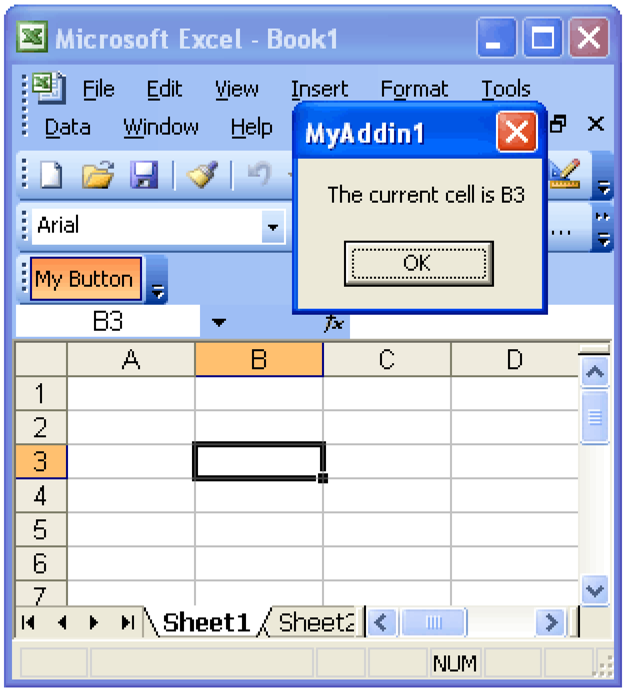
Choose the Register Add-in Express Project item in the Build menu, restart the host application(s) you selected, find your toolbar and click the button.



MyAddin1

There are 63 words currently selected

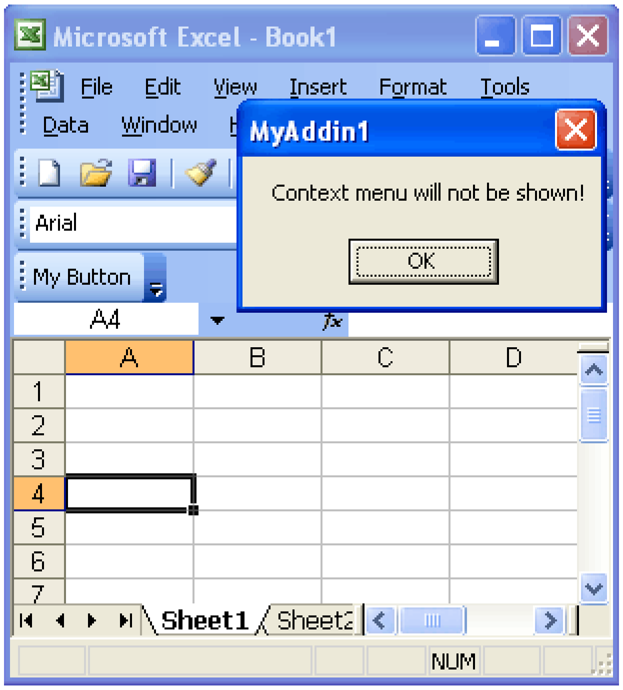
OK



MyAddin1

The current cell is B3

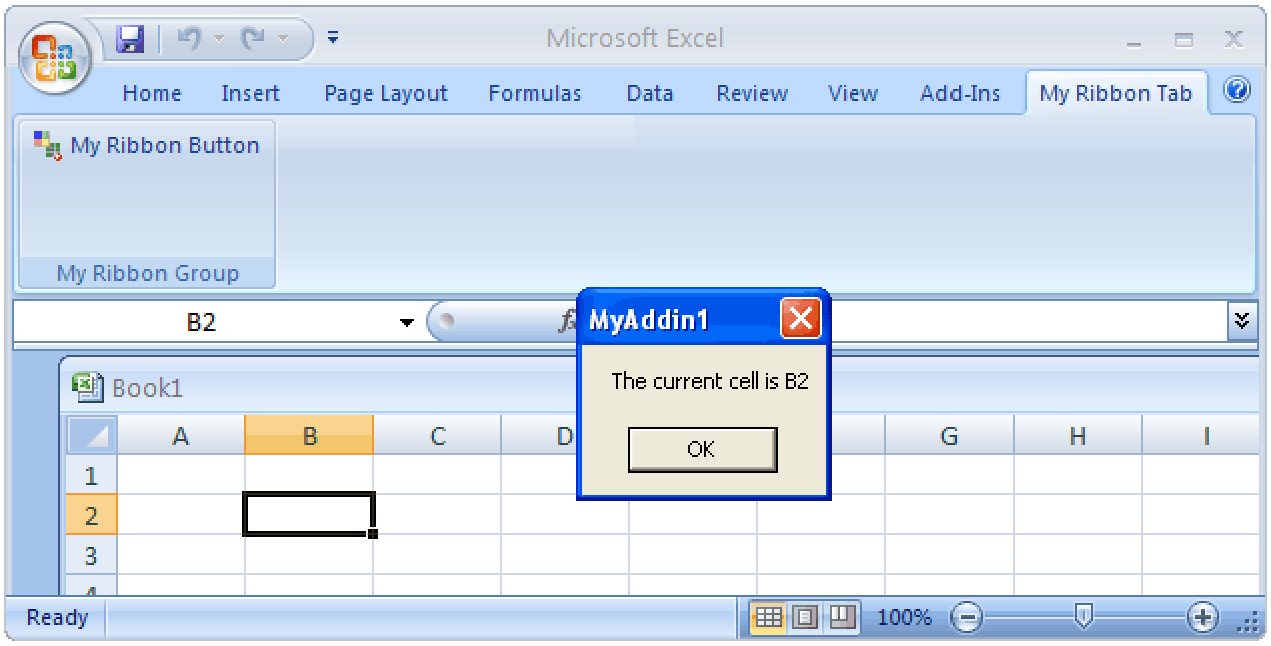
OK



MyAddin1

Context menu will not be shown!

OK

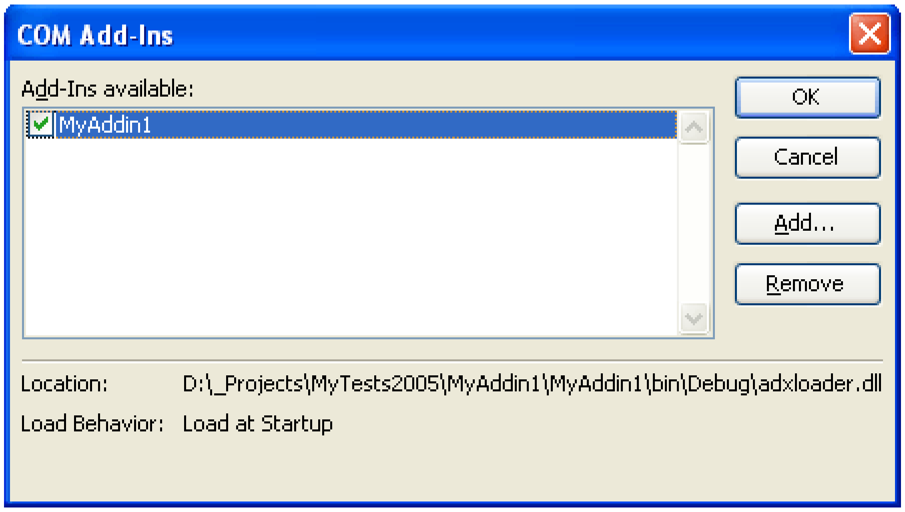

Microsoft Excel
Home Insert Page Layout Formulas Data Review View Add-Ins My Ribbon Tab
My Ribbon Button
My Ribbon Group
B2
Book1
Ready
100%

MyAddin1

The current cell is B2

OK

You find your add-in in the COM Add-ins dialog:



Add-Ins available:

[✓] MyAddin1

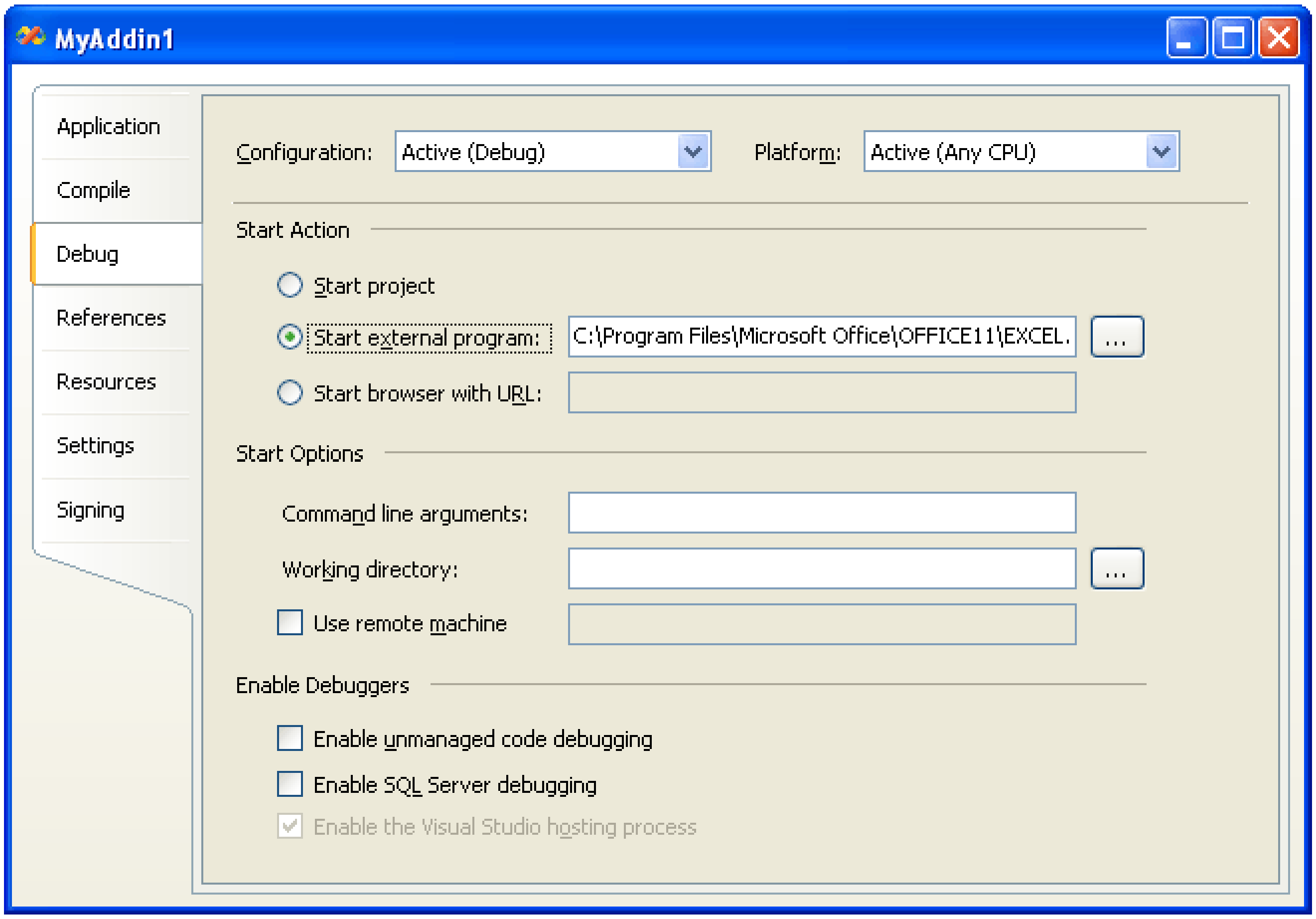
Location: D:_Projects\MyTests2005\MyAddin1\MyAddin1\bin\Debug\adxloader.dll

Load Behavior: Load at Startup

See also Add the COM Add-ins Command to a Toolbar or Menu.

Step #11 - Debugging the COM Add-in

To debug your add-in, just indicate the add-in host application as the Start Program in the Project Options window.



Configuration: Active (Debug)

Platform: Active (Any CPU)

Start Action

- Start project
- [✓ Start external program:] C:\Program Files\Microsoft Office\OFFICE11\EXCEL.
- Start browser with URL:

Start Options

Command line arguments:

Working directory:

☐ Use remote machine

Enable Debuggers

☐ Enable unmanaged code debugging

☐ Enable SQL Server debugging

✓ Enable the Visual Studio hosting process

However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

Step #12 - Deploying the COM Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in.

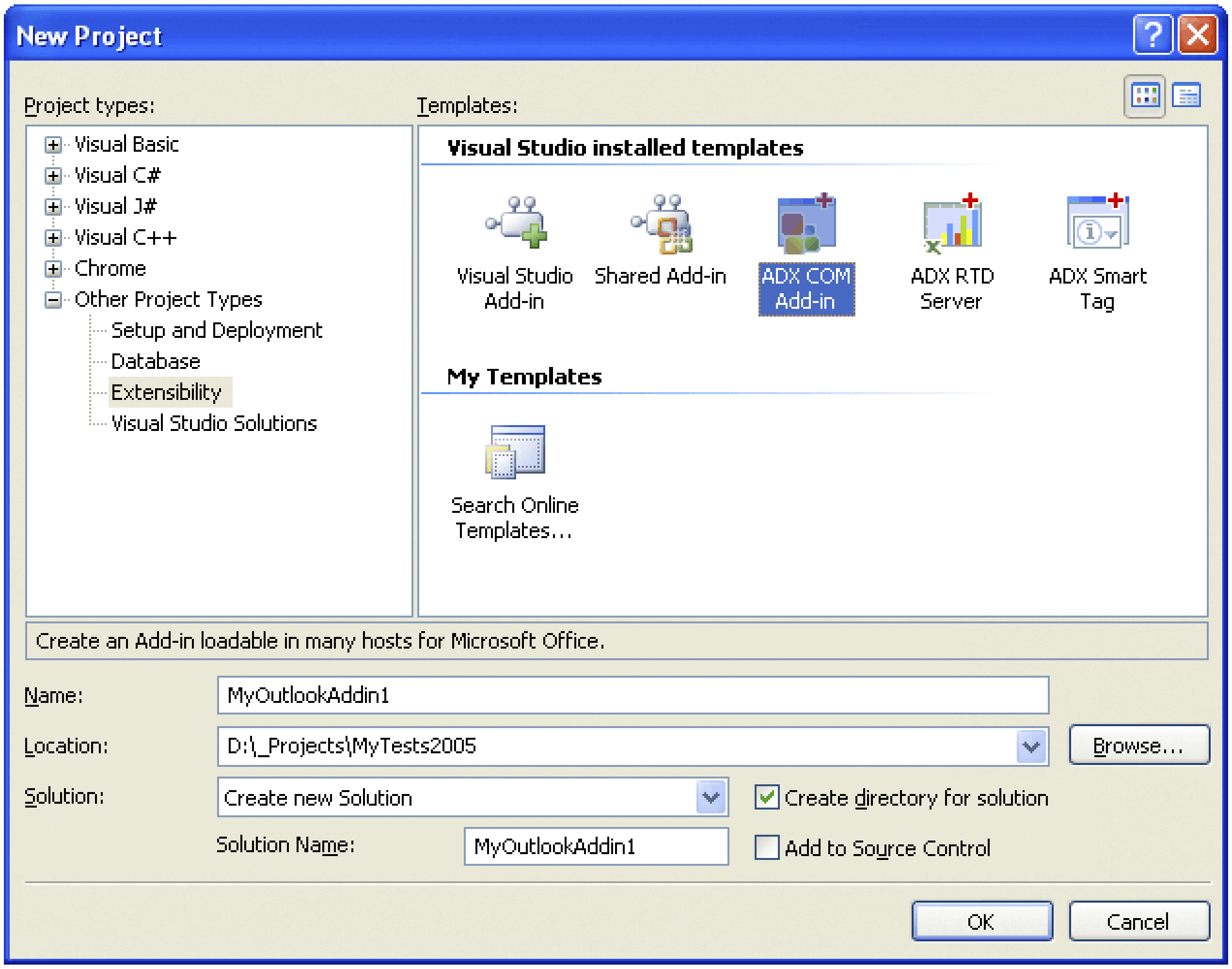
Your First Microsoft Outlook COM Add-in

Add-in Express provides two Outlook specific command bars: ADXOlExplorerCommandBar and ADXOlInspectorCommandBar. The first adds a command bar to the Outlook Explorer window and solves many problems with custom Outlook command bars. The latter adds a command bar to the Outlook Inspector window. Both ADXOlExplorerCommandBar and ADXOlInspectorCommandBar have the FolderName(s) and ItemTypes properties that add context-sensitivity to Outlook command bars. The OlExplorerItemTypes, OlInspectorItemTypes, and OlItemTypeAction properties add context-sensitivity to Outlook command bar controls.

Additionally, Add-in Express supplies the Outlook Property Page component that helps you to add your pages to the Options (Tools | Options menu) and folder Properties dialogs.

Step #1 - Creating an Add-in Express COM Add-in Project

Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.



Create an Add-in loadable in many hosts for Microsoft Office.

Name: MyOutlookAddin1

Location: D:_Projects\MyTests2005

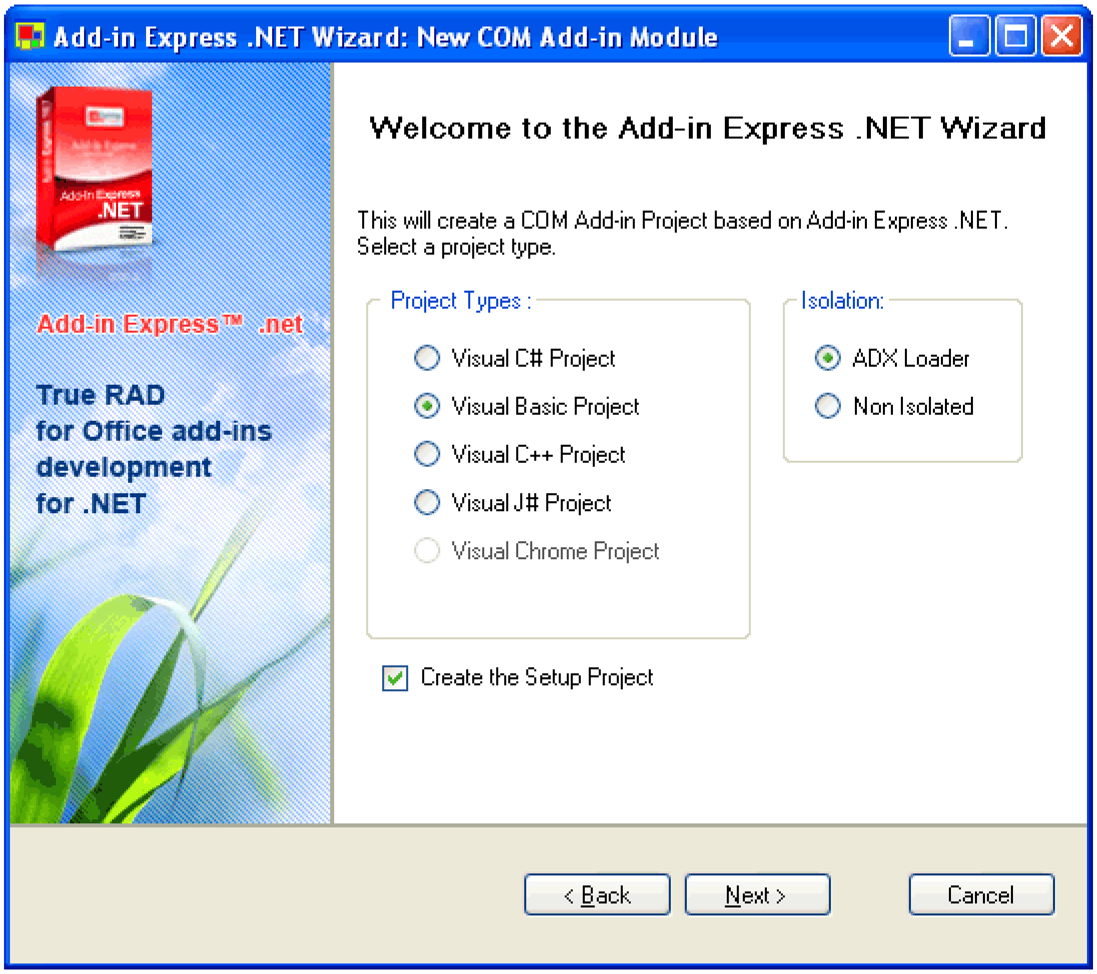
Solution: Create new Solution

Solution Name: MyOutlookAddin1

[✓] Create directory for solution

[ ] Add to Source Control

When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, and supported applications of your add-in.


Add-in Express .NET Wizard: New COM Add-in Module
Add-in Express™ .net
True RAD for Office add-ins development for .NET
< Back
Next >
Cancel

Welcome to the Add-in Express .NET Wizard

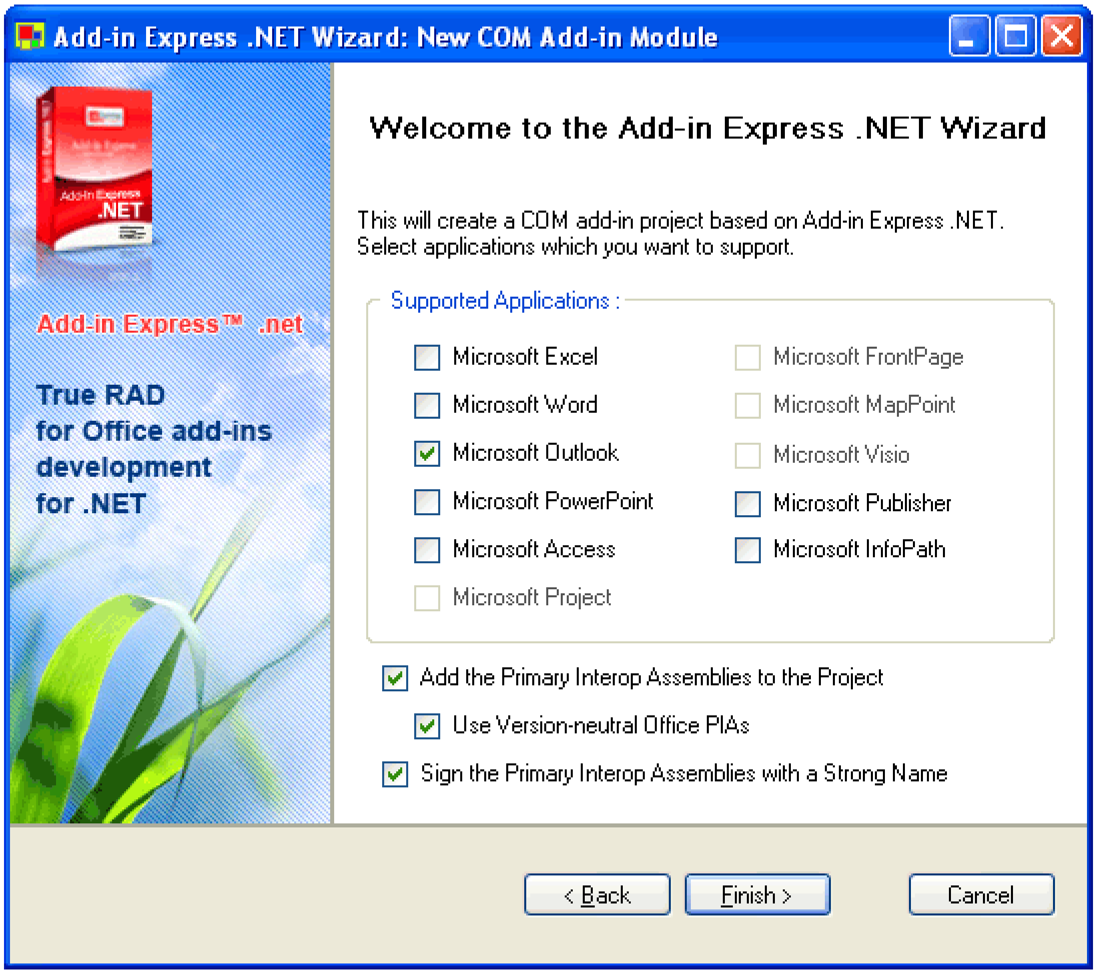
This will create a COM Add-in Project based on Add-in Express .NET.
Select a project type.

Project Types :
- ○ Visual C# Project
- ● Visual Basic Project
- ○ Visual C++ Project
- ○ Visual J# Project
- ○ Visual Chrome Project

Isolation:
- ● ADX Loader
- ○ Non Isolated

☑ Create the Setup Project

This VB.NET sample shows an Add-in Express COM Add-in project implementing an Outlook COM add-in with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.


Add-in Express .NET Wizard: New COM Add-in Module

Welcome to the Add-in Express .NET Wizard

This will create a COM add-in project based on Add-in Express .NET.
Select applications which you want to support.

Supported Applications :

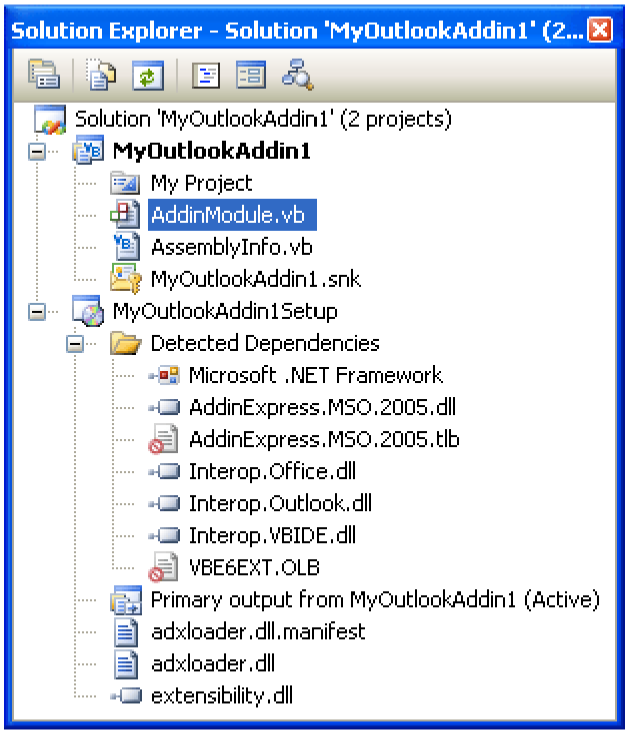
- [ ] Microsoft Excel
- [ ] Microsoft Word
- [x] Microsoft Outlook
- [ ] Microsoft PowerPoint
- [ ] Microsoft Access
- [ ] Microsoft Project
- [ ] Microsoft FrontPage
- [ ] Microsoft MapPoint
- [ ] Microsoft Visio
- [ ] Microsoft Publisher
- [ ] Microsoft InfoPath

- [x] Add the Primary Interop Assemblies to the Project
  - [x] Use Version-neutral Office PIAs
- [x] Sign the Primary Interop Assemblies with a Strong Name

The Add-in Express Project Wizard creates and opens the COM Add-in solution in IDE. The solution includes the COM Add-in project and the setup project.

Your add-ins are version-neutral

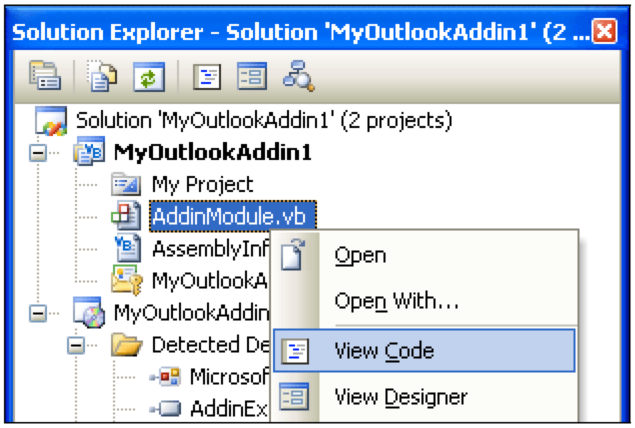
All previous versions of the core Add-in Express were version-neutral. Now we are ready to represent the version-neutral programming model for all our customers who develop extensions for Office 2000+. Add-in Express 2007 delivers version-neutral Office interop assemblies. To use them, simply check the Use Version-neutral Office PIAs check box when creating your projects.



The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in the next step.

Step #2 - Add-in Express COM Add-in Module

The AddinModule.vb (or AddinModule1.cs) is a COM Add-in Module that is the core part of the COM add-in project (see COM Add-ins). It is the placeholder of the Add-in Express components, which allow you to concentrate on the functionality of your add-in. You specify add-in properties in the module's properties, add Add-in Express components to the module's designer, and write the functional code of your add-in in this module. To review its source code, in the Solution Explorer window, right-click the AddinModule1.vb (or AddinModule1.cs) file and choose the View Code popup menu item.



The code for AddinModule1.vb is as follows:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Add-in Module
<GuidAttribute("3BDF26A5-74E4-42CB-A93A-E88435BC0AD3"), _
   ProgIdAttribute("MyOutlookAddin1.AddinModule")> _
Public Class AddinModule
   Inherits AddinExpress.MSO.ADXAddinModule


#Region " Component Designer generated code. "
   'Required by designer
   Private components As System.ComponentModel.IContainer

   'Required by designer - do not modify
   'the following method
   Private Sub InitializeComponent()
      Me.components = New System.ComponentModel.Container
      '
      'AddinModule
      '
      Me.AddinName = "MyOutlookAddin1"
      Me.SupportedApps = AddinExpress.MSO.ADXOfficeHostApp.ohaOutlook

   End Sub

#End Region

#Region " Add-in Express automatic code "

   'Required by Add-in Express - do not modify
   'the methods within this region

   Public Overrides Function GetContainer() As _
      System.ComponentModel.IContainer
      If components Is Nothing Then
         components = New System.ComponentModel.Container
      End If
      GetContainer = components
   End Function

   <ComRegisterFunctionAttribute()> _
   Public Shared Sub AddinRegister(ByVal t As Type)
      AddinExpress.MSO.ADXAddinModule.ADXRegister(t)
```

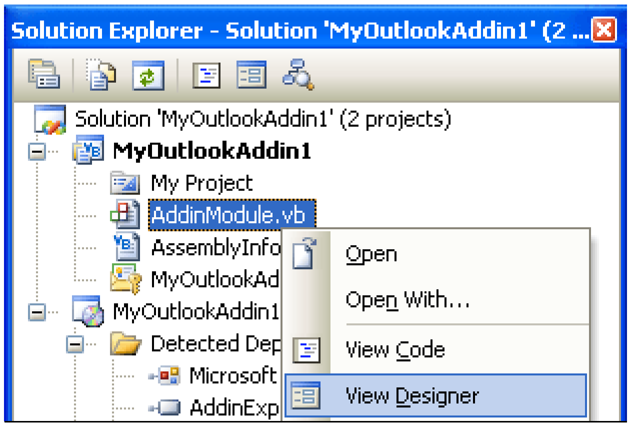
```
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub AddinUnregister(ByVal t As Type)
       AddinExpress.MSO.ADXAddinModule.ADXUnregister(t)
    End Sub

    Public Overrides Sub UninstallControls()
       MyBase.UninstallControls()
    End Sub

 #End Region

    Public Sub New()
       MyBase.New()

       'This call is required by the Component Designer
       InitializeComponent()

       'Add any initialization after the InitializeComponent() call

    End Sub

    Public ReadOnly Property OutlookApp() As Outlook._Application
       Get
          Return CType(HostApplication, Outlook._Application)
       End Get
    End Property

 End Class
```

Please, pay attention to the OutlookApp property of the module generated by the Add-in Express Project Wizard. You can use it in your code to get access to Outlook objects.

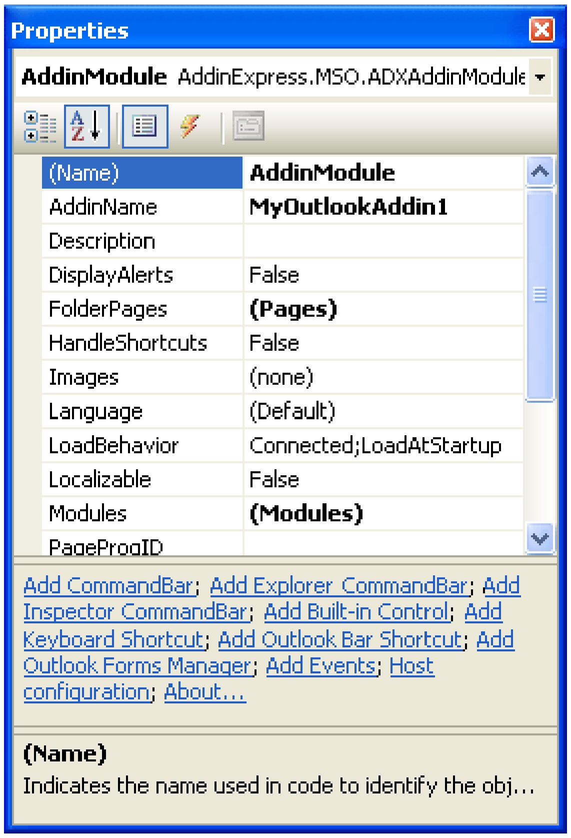
Step #3 - Add-in Express COM Add-in Designer

The Add-in Express COM Add-in Designer allows setting add-in properties and adding components to the module.

In the Solution Explorer window, right-click the AddinModule.vb (or AddinModule.cs) file and choose the View Designer popup menu item.



In the Properties window, you set the name and description of your add-in module (see COM Add-ins).


Properties

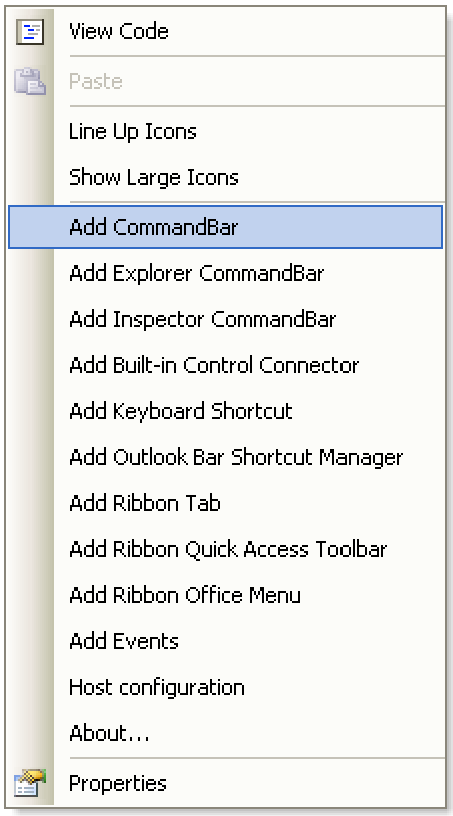
AddinModule AddinExpress.MSO.ADXAddinModule

(Name)	AddinModule
AddinName	MyOutlookAddin1
Description	
DisplayAlerts	False
FolderPages	(Pages)
HandleShortcuts	False
Images	(none)
Language	(Default)
LoadBehavior	Connected;LoadAtStartup
Localizable	False
Modules	(Modules)
PageProgID	

Add CommandBar; Add Explorer CommandBar; Add Inspector CommandBar; Add Built-in Control; Add Keyboard Shortcut; Add Outlook Bar Shortcut; Add Outlook Forms Manager; Add Events; Host configuration; About...

(Name)
Indicates the name used in code to identify the obj...

To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.



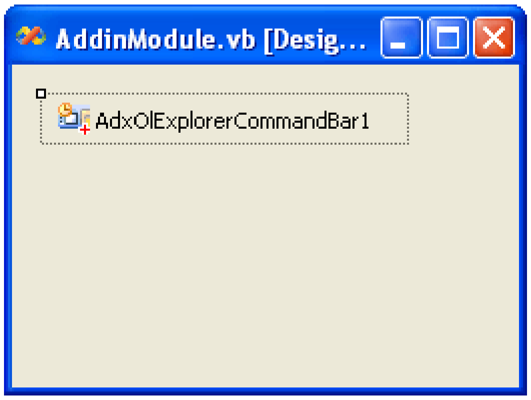
The following commands add the following components to the module:

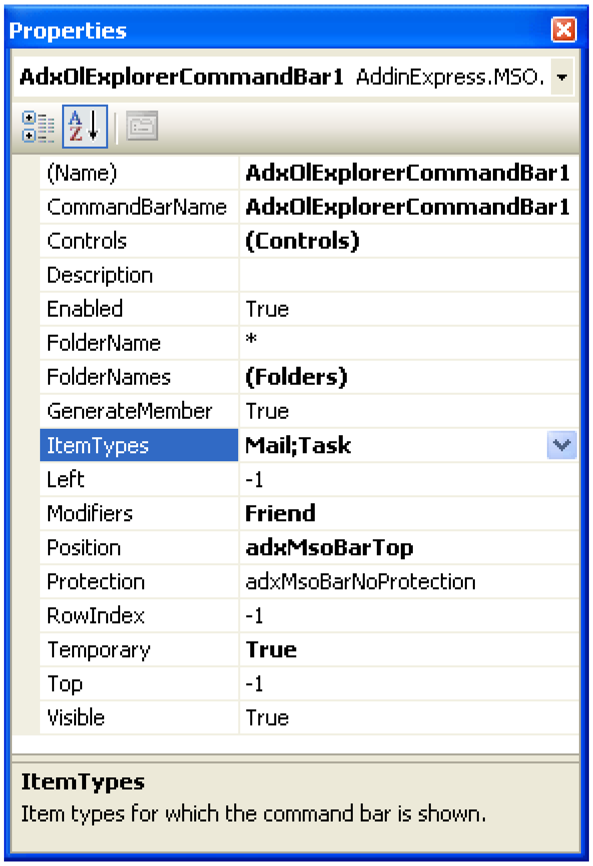
- Add CommandBar – adds a command bar to your add-in (see Command Bars)
- Add Explorer CommandBar – adds an Outlook Explorer command bar to your add-in (see Command Bars)
- Add Inspector CommandBar – adds an Outlook Inspector command bar to your add-in (see Command Bars)
- Add Built-in Control Connector – adds a component that allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- Add Keyboard Shortcut– adds a component that allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)
- Add Outlook Bar Shortcut Manager – adds a component that allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- Add Outlook Forms Manager – adds a component that allows embedding custom .NET forms into Outlook windows (see Outlook Forms Manager)
- Add Ribbon Tab – adds a Ribbon tab to your add-in (see Office 2007 Ribbon Components)
- Add Ribbon Quick Access Toolbar – adds a component that allows customizing the Ribbon Quick Access Toolbar in your add-in (see Office 2007 Ribbon Components)

- Add Ribbon Office Menu – adds a component that allows customizing the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)
- Add Events – adds or deletes components that provide access to application-level events of the add-in host applications (see Application-level Events)

Step #4 - Adding a New Explorer Command Bar

To add a command bar to the Outlook Explorer window, use the Add Explorer CommandBar command that adds an ADXOlExplorerCommandBar component to the COM Add-in Module.





(Name)	AdxOlExplorerCommandBar1
CommandBarName	AdxOlExplorerCommandBar1
Controls	(Controls)
Description	
Enabled	True
FolderName	*
FolderNames	(Folders)
GenerateMember	True
ItemTypes	Mail;Task
Left	-1
Modifiers	Friend
Position	adxMsoBarTop
Protection	adxMsoBarNoProtection
RowIndex	-1
Temporary	True
Top	-1
Visible	True

ItemTypes
Item types for which the command bar is shown.

Select the Add-in Express Explorer Command Bar component, and, in the Properties window, specify the command bar name using the CommandBarName property and choose its position (see the Position property). Outlook-specific versions of Add-in Express Command Bar component provides context-sensitive properties. They are FolderName, FolderNames, and ItemTypes (see Outlook CommandBar Visibility Rules and COM Add-ins for Outlook – Template Characters in FolderName).

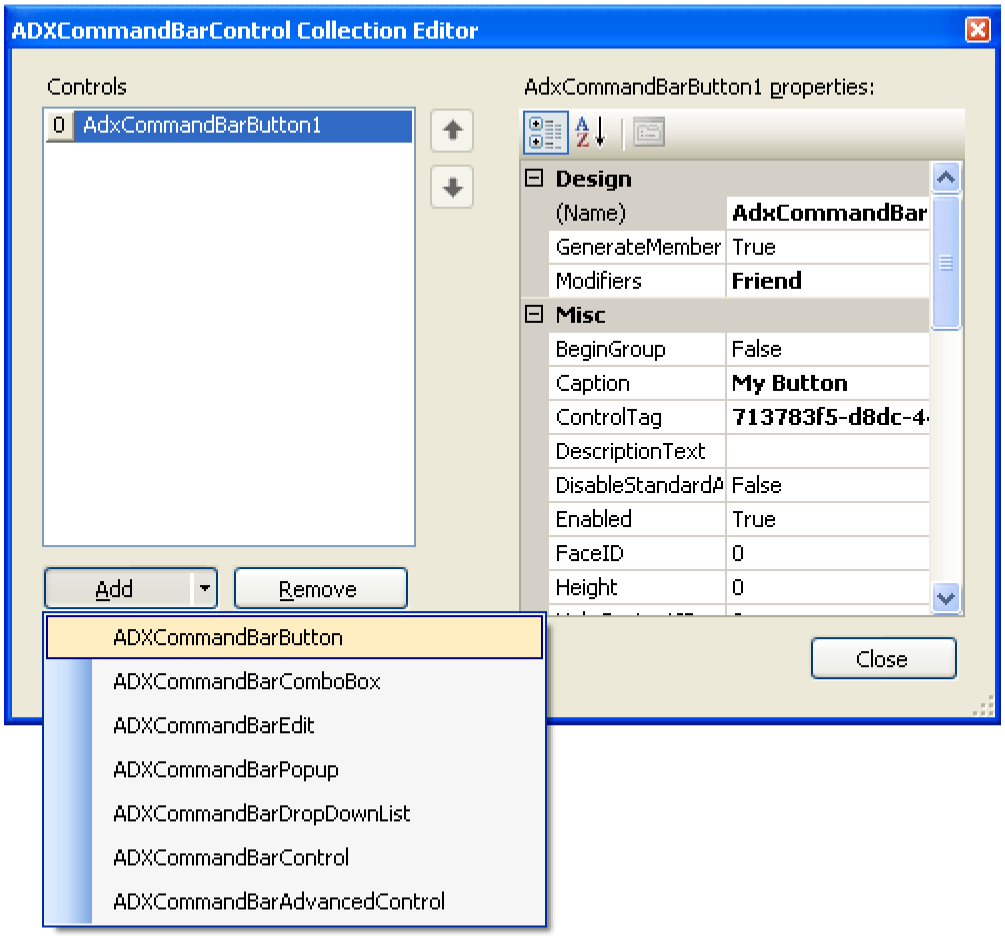
In the screenshot, you see the Outlook Explorer command bar that will be shown for every Outlook folder (FolderName = "*") the default item types of which are Mail or Task.

See also Command Bars.

Step #5 – Adding a New Command Bar Button

To add a new button to the Explorer command bar, in the Properties window, you select the Controls property and click the property editor button (the button in the property value field).

In the ADXCommandBarControl Collection Editor, you select the command bar control type in the Add button (see also Command Bar Controls).



AdxCommandBarButton1 properties:

Design	
(Name)	**AdxCommandBar**
GenerateMember	True
Modifiers	**Friend**
Misc	
BeginGroup	False
Caption	**My Button**
ControlTag	**713783f5-d8dc-4**
DescriptionText	
DisableStandardA	False
Enabled	True
FaceID	0
Height	0

Specify the button's Caption property, set the Style property (default value = adxMsoButtonCaption), and close the collection editor. To handle the Click event of the button, select the added button in the topmost combo of the Properties window and add the Click event handler. The code of the event handler follows below (it is empty as you can see)

```
    Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
        Handles AdxCommandBarButton1.Click

    End Sub
```

Step #6 – Accessing Outlook Objects

The Add-in Module provides the HostApplication property that returns the Application object (of the Object type) of the host application the add-in is currently running in. For your convenience, the Add-in Express Project Wizard adds host-related properties to the Add-in module. You use these properties to access host application objects. For instance, this sample add-in includes the OutlookApp property in the Add-in Module:

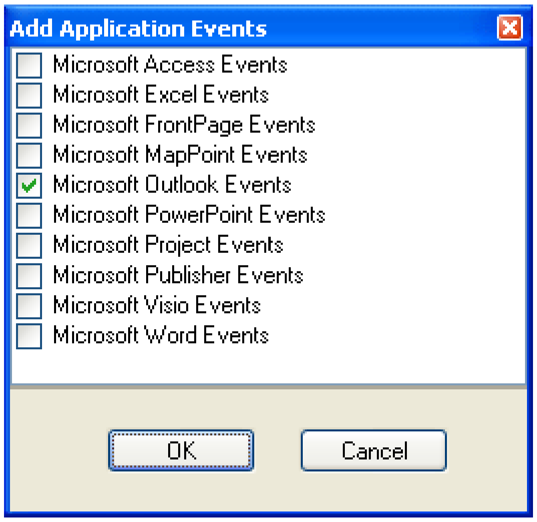
```
      Public ReadOnly Property OutlookApp() As Outlook._Application
         Get
            Return CType(HostApplication, Outlook._Application)
         End Get
      End Property
```

The _Application object provides the same properties and methods as the Application object but it doesn't provide events. This allows you to write the following code to the Click event of the button just added:

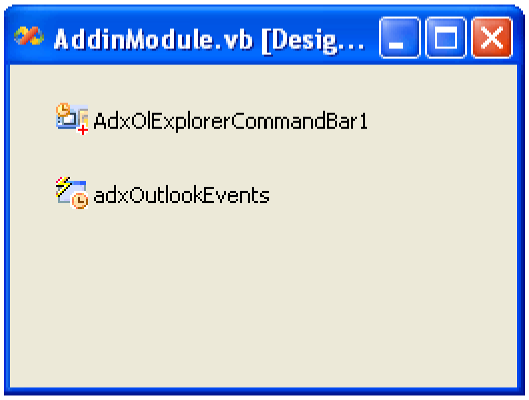
```
      Private Sub AdxCommandBarButton1_Click(ByVal sender As System.Object) _
         Handles AdxCommandBarButton1.Click
         MsgBox("The subject is:" _
            + vbCrLf _
            + Me.OutlookApp.ActiveExplorer.Selection.Item(1).Subject)
      End Sub
```

Step #7 – Handling Outlook Events

The COM Add-in Designer provides the Add Events command that adds (and remove) event components that allow handling application-level events. When you choose this command, the Add Application Events dialog box opens allowing you to select the application Events components you need. This adds the Outlook Events component to the COM Add-in Module.


Add Application Events

- [ ] Microsoft Access Events
- [ ] Microsoft Excel Events
- [ ] Microsoft FrontPage Events
- [ ] Microsoft MapPoint Events
- [✓] Microsoft Outlook Events
- [ ] Microsoft PowerPoint Events
- [ ] Microsoft Project Events
- [ ] Microsoft Publisher Events
- [ ] Microsoft Visio Events
- [ ] Microsoft Word Events


AddinModule.vb [Desig...

AdxOlExplorerCommandBar1

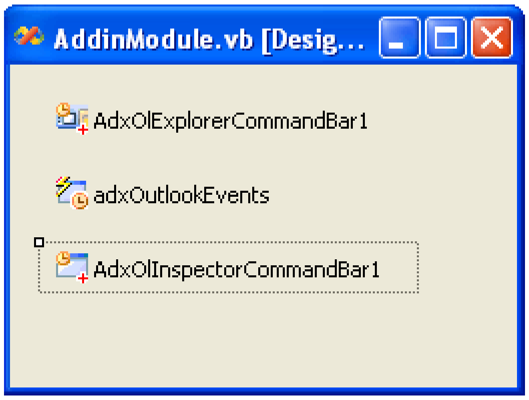
adxOutlookEvents

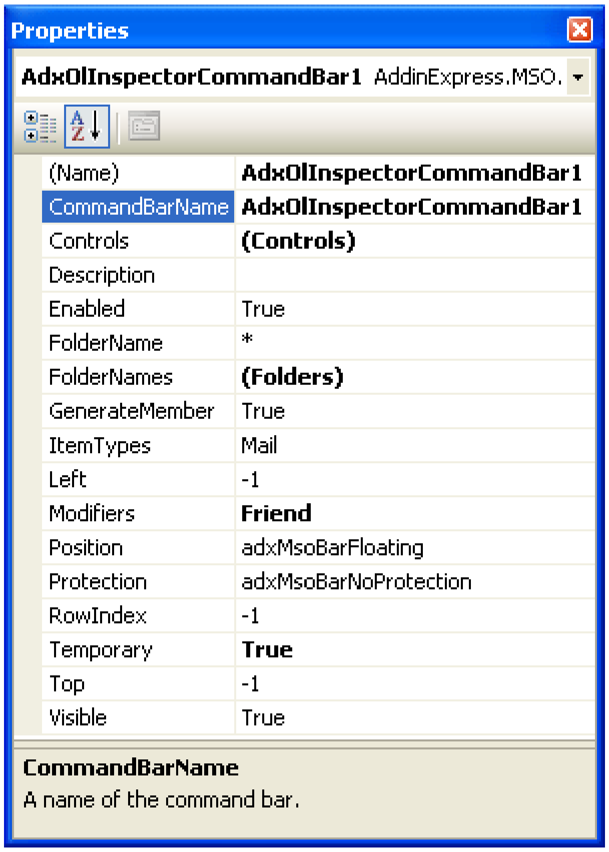
With the Outlook Events component, you handle any application-level events of Outlook. For instance, the following code handles the BeforeFolderSwitch event of the Outlook Explorer class:

```
      Private Sub adxOutlookEvents_ExplorerBeforeFolderSwitch _
         (ByVal sender As Object, _
         ByVal e As _
             AddinExpress.MSO.ADXOlExplorerBeforeFolderSwitchEventArgs) _
         Handles adxOutlookEvents.ExplorerBeforeFolderSwitch
         MsgBox("You are switching to the " + e.NewFolder.Name + " folder")
      End Sub
```

Step #8 - Adding a New Inspector Command Bar

To add a command bar to the Outlook Inspector window, use the Add Inspector CommandBar command that adds an ADXOlInspectorCommandBar component to the COM Add-in Module.





(Name)	AdxOlInspectorCommandBar1
CommandBarName	AdxOlInspectorCommandBar1
Controls	(Controls)
Description	
Enabled	True
FolderName	*
FolderNames	(Folders)
GenerateMember	True
ItemTypes	Mail
Left	-1
Modifiers	Friend
Position	adxMsoBarFloating
Protection	adxMsoBarNoProtection
RowIndex	-1
Temporary	True
Top	-1
Visible	True

CommandBarName
A name of the command bar.

The Inspector command bar component provides the same properties as the Explorer command bar component. In the screenshot below you see the default values for the Inspector command bar.

Add a new command bar button to the command bar (see Step #5 – Adding a New Command Bar Button for details).

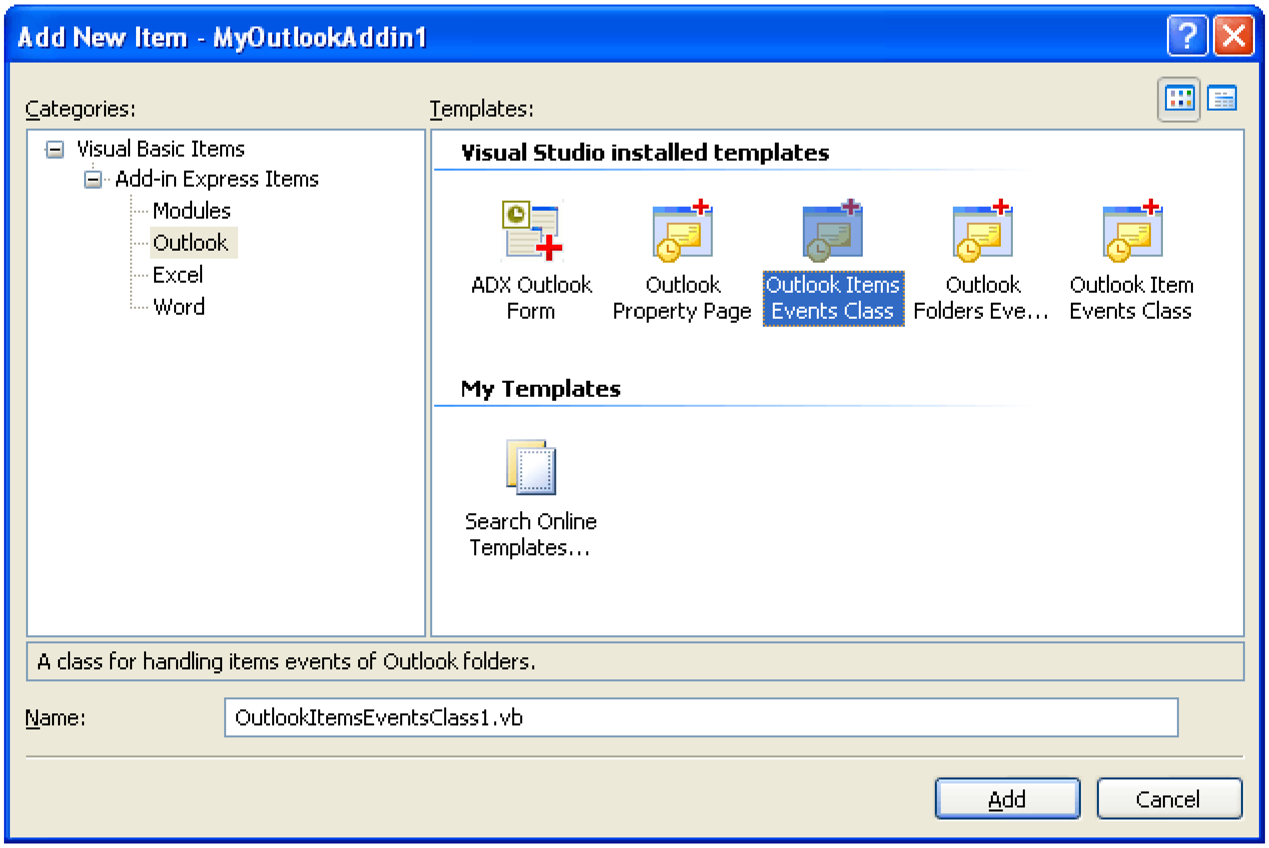
Now you can show the subject of the currently open mail item using the following code that handles the Click event of the button:

```
      Private Sub AdxCommandBarButton2_Click(ByVal sender As System.Object) _
               Handles AdxCommandBarButton2.Click
         MsgBox("The subject is:" _
            + vbCrLf _
            + Me.OutlookApp.ActiveInspector.CurrentItem.Subject)
      End Sub
```

See also Command Bars, Outlook CommandBar Visibility Rules, and COM Add-ins for Outlook – Template Characters in FolderName.

Step #9 - Handling Events of Outlook Items Object

The Outlook MAPIFolder class provides the Items collection. This collection provides the following events: ItemAdd, ItemChange, and ItemRemove. To process these events, you use the Outlook Items Events Class located in the Add-in Express Items folder in the Add New Item dialog:



A class for handling items events of Outlook folders.

Name: OutlookItemsEventsClass1.vb

This will add the OutlookItemsEventsClass1.vb class to the add-in project. You handle the ItemAdd event by entering some code into the ProcessItemAdd procedure of the class:

```
   Imports System

   'Add-in Express Outlook Items Events Class
   Public Class OutlookItemsEventsClass1
```

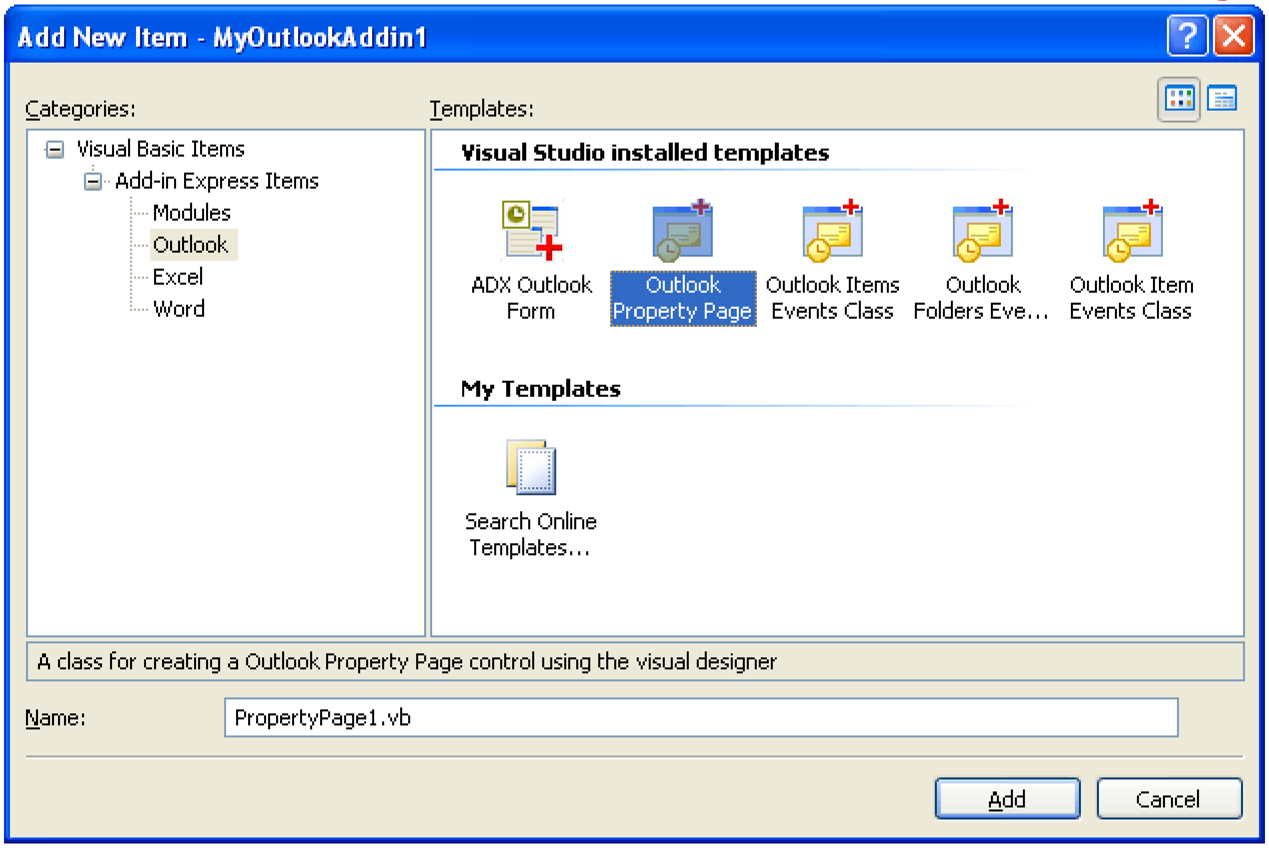
```
      Inherits AddinExpress.MSO.ADXOutlookItemsEvents

      Public Sub New(ByVal ADXModule As AddinExpress.MSO.ADXAddinModule)
         MyBase.New(ADXModule)
      End Sub

      Public Overrides Sub ProcessItemAdd(ByVal Item As Object)
         MsgBox("The item with subject '" + Item.Subject + _
            "' has been added to the Inbox folder")
      End Sub

      Public Overrides Sub ProcessItemChange(ByVal Item As Object)
         'TODO: Add some code
      End Sub

      Public Overrides Sub ProcessItemRemove()
         'TODO: Add some code
      End Sub
   End Class
```

This requires you to add the following declarations and code to the Add-in Module:

```
      Dim ItemsEvents As OutlookItemsEventsClass1 = _
          New OutlookItemsEventsClass1(Me)
   ...
      Private Sub AddinModule_AddinBeginShutdown(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Me.AddinBeginShutdown
         If ItemsEvents IsNot Nothing Then
            ItemsEvents.RemoveConnection()
            ItemsEvents = Nothing
         End If
      End Sub

      Private Sub AddinModule_AddinStartupComplete(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Me.AddinStartupComplete
         ItemsEvents.ConnectTo( _
            AddinExpress.MSO.ADXOlDefaultFolders.olFolderInbox, True)
      End Sub
```

Step #10 - Adding Folder Property Pages

Outlook allows you to add custom option pages to the Options dialog box (the Tools | Options menu) and / or to the Properties dialog box of any folder. To automate this task, Add-in Express provides the ADXOlPropertyPage component. You find it in the Add New Item dialog box.



A class for creating a Outlook Property Page control using the visual designer

Name: PropertyPage1.vb

Click the Add button to add a new property page instance, a descendant of the ADXOlPropertyPage class that implements the IPropertyPage interface:

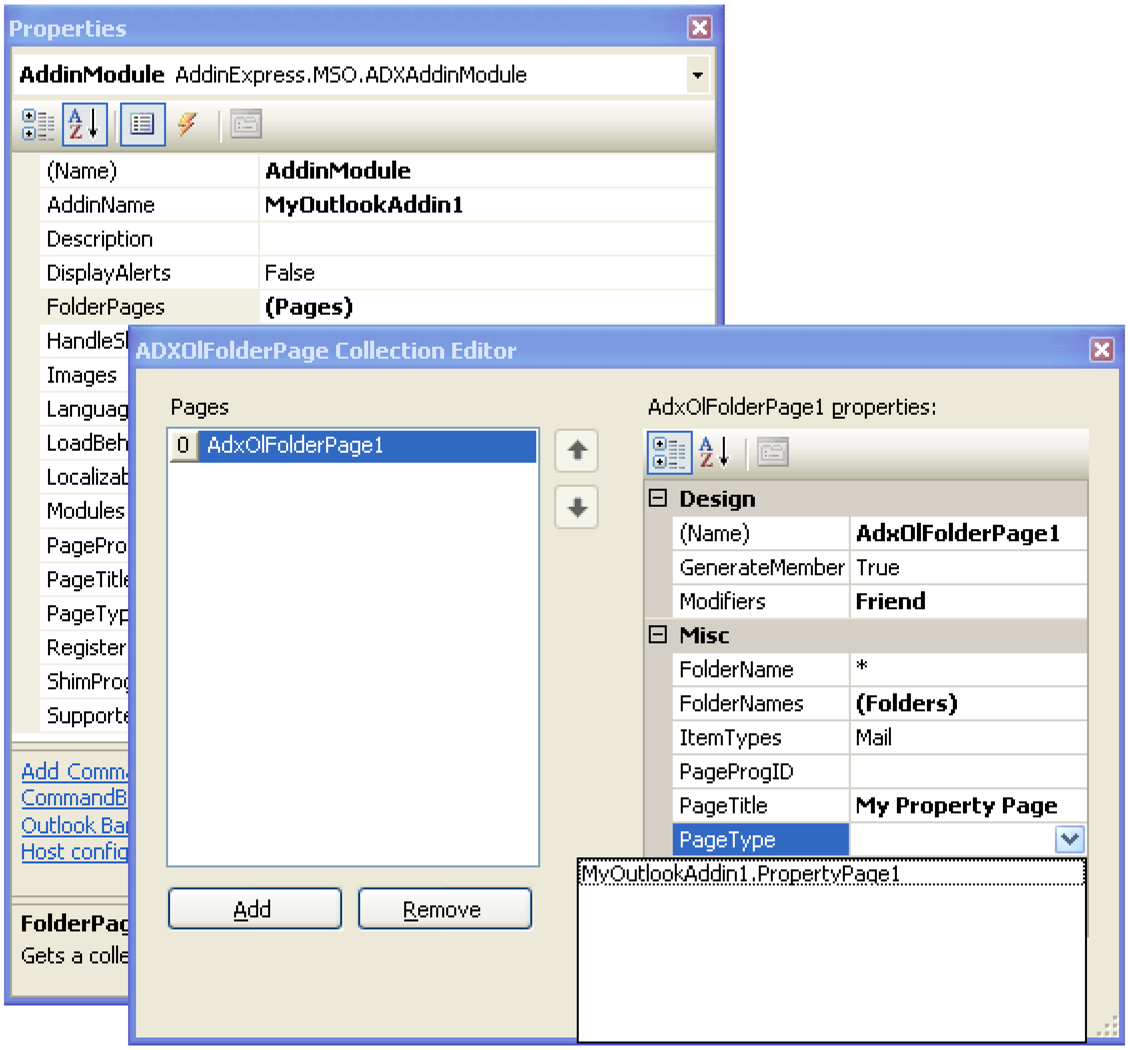
```
Imports System.Runtime.InteropServices

'Add-in Express Outlook Option Page
<GuidAttribute("D8F61EE3-B676-4FFA-9CAF-BAC46C1F0934"), _
    ProgIdAttribute("PropertyPage1.OutlookPage")> _
Public Class PropertyPage1
    Inherits AddinExpress.MSO.ADXOlPropertyPage

#Region " Component Designer generated code "
    Public Sub New()
        MyBase.New()

        'This call is required by the Component Designer
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
    End Sub

   'Clean up any resources being used
   Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
         If disposing Then
            If Not (components Is Nothing) Then
               components.Dispose()
            End If
         End If
         MyBase.Dispose(disposing)
      End Sub

       'Required by designer
       Private components As System.ComponentModel.IContainer

       'Required by designer - do not modify
       'the following method
       <System.Diagnostics.DebuggerStepThrough()> _
       Private Sub InitializeComponent()
           components = New System.ComponentModel.Container()
           '
           'PropertyPage1
           '
           Me.Name = "PropertyPage1"
           Me.Size = New System.Drawing.Size(413, 358)
       End Sub
   #End Region
   End Class
```

You can customize the page as an ordinary form: add the controls and handle their events.

To add a property page to the <FolderName> Properties dialog box of an Outlook folder(s), you do the following:

- In the AddinModule properties, run the editor of the FolderPages property,
- Click the Add button,
- Specify the folder you need in the FolderName property,
- Set the PageType property to the PropertyPage component you've added
- Specify the Title property and close the dialog box.


Properties
AddinModule AddinExpress.MSO.ADXAddinModule
ADXOlFolderPage Collection Editor
Pages
Add
Remove
AdxOlFolderPage1 properties:

(Name)	AddinModule
AddinName	MyOutlookAddin1
Description	
DisplayAlerts	False
FolderPages	(Pages)

Pages: 0 AdxOlFolderPage1

Design	
(Name)	AdxOlFolderPage1
GenerateMember	True
Modifiers	Friend
Misc	
FolderName	*
FolderNames	(Folders)
ItemTypes	Mail
PageProgID	
PageTitle	My Property Page
PageType	MyOutlookAddin1.PropertyPage1

The screenshot above shows the settings you need to display your page in the Folder Properties dialog for all Mail folders (ItemTypes = Mail).

In order to limit the property page to the Inbox folder only, change the code of the AddinStartupComplete event in the AddinModule as follows:

```
Private Sub AddinModule_AddinStartupComplete(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Me.AddinStartupComplete
    ItemsEvents.ConnectTo( _
       AddinExpress.MSO.ADXOlDefaultFolders.olFolderInbox, True)
    Dim FullPath As String = _
       GetFolderPath(Me.OutlookApp.GetNamespace("Mapi"). _
          GetDefaultFolder(Outlook.OlDefaultFolders.olFolderInbox))
    ' remove leading backslashes
    Me.FolderPages.Item(0).FolderName = _
       FullPath.Substring(2, FullPath.Length - 2)
End Sub
```

The GetFolderPath Function

See its code here: FolderPath Property Value in Outlook 2000 and XP.

In order to control the events for the folder, add a checkbox to the page and handle its CheckedChanged event as well as the Dirty, Apply, and Load events of the page as follows:

```
      ...
      Friend WithEvents CheckBox1 As System.Windows.Forms.CheckBox
      Private TrackStatusChanges As Boolean

      ...

      Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged
         If Not TrackStatusChanges Then Me.OnStatusChange()
      End Sub

      Private Sub PropertyPage1_Dirty(ByVal sender As System.Object, _
         ByVal e As AddinExpress.MSO.ADXDirtyEventArgs) Handles MyBase.Dirty
         e.Dirty = True
      End Sub

      Private Sub PropertyPage1_Apply(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles MyBase.Apply
         CType(AddinModule.CurrentInstance, _
            MyOutlookAddin1.AddinModule).IsFolderTracked = _
            Me.CheckBox1.Checked
      End Sub

      Private Sub PropertyPage1_Load(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Me.Load
         TrackStatusChanges = True
         Me.CheckBox1.Checked = _
            CType(AddinModule.CurrentInstance, _
               MyOutlookAddin1.AddinModule).IsFolderTracked
         TrackStatusChanges = False
      End Sub
   End Class
```

Finally, you add the following property to the AddinModule code:

```
      ...
      Friend Property IsFolderTracked() As Boolean
         Get
```

```
            Return ItemsEvents.IsConnected
         End Get
         Set(ByVal value As Boolean)
            If value Then
               ItemsEvents.ConnectTo(ADXOlDefaultFolders.olFolderInbox, True)
            Else
               ItemsEvents.RemoveConnection()
            End If
         End Set
      End Property
```

Adding a page to the Outlook Options dialog

To add this or other property page to the main Options dialog box, you use the PageType and PageTitle properties of the add-in module.

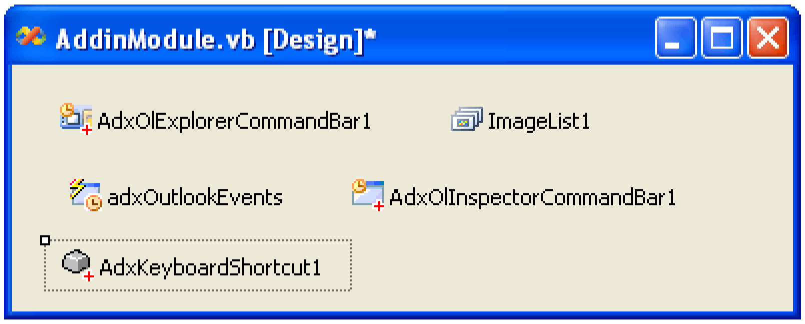
PageProgId property is obsolete

Note, whether you add a folder property page or an Outlook Options page, do not use the PageProgId property. It is obsolete; we left it for compatibility only.

See also Outlook Property Page.

Step #11 - Intercepting Keyboard Shortcut

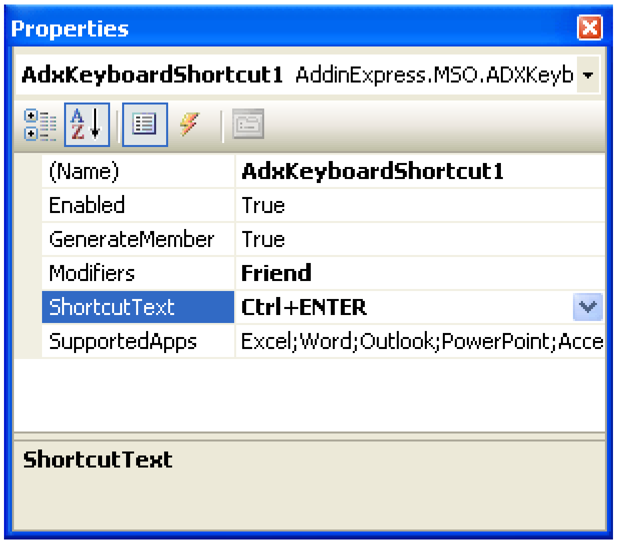
To intercept a keyboard shortcut, you add an ADXKeyboardShortcut component to the COM Add-in Module using the Add Keyboard Shortcut command of the module.



In the Properties window you select (or enter) the desired shortcut in the ShortcutText property. We chose the shortcut for the Send button in the mail Inspector's Standard command bar. It is Ctrl+Enter.

HandleShortcuts property

To use keyboard shortcuts, you need to set the HandleShortcuts property of AddinModule to true.



(Name)	AdxKeyboardShortcut1
Enabled	True
GenerateMember	True
Modifiers	Friend
ShortcutText	Ctrl+ENTER
SupportedApps	Excel;Word;Outlook;PowerPoint;Acce

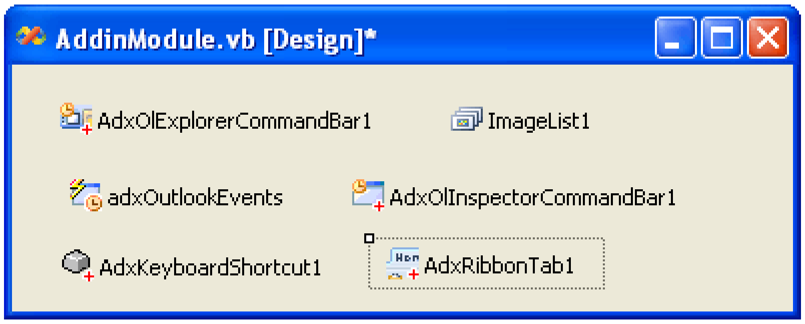
ShortcutText

Now you handle the Action event of the component:

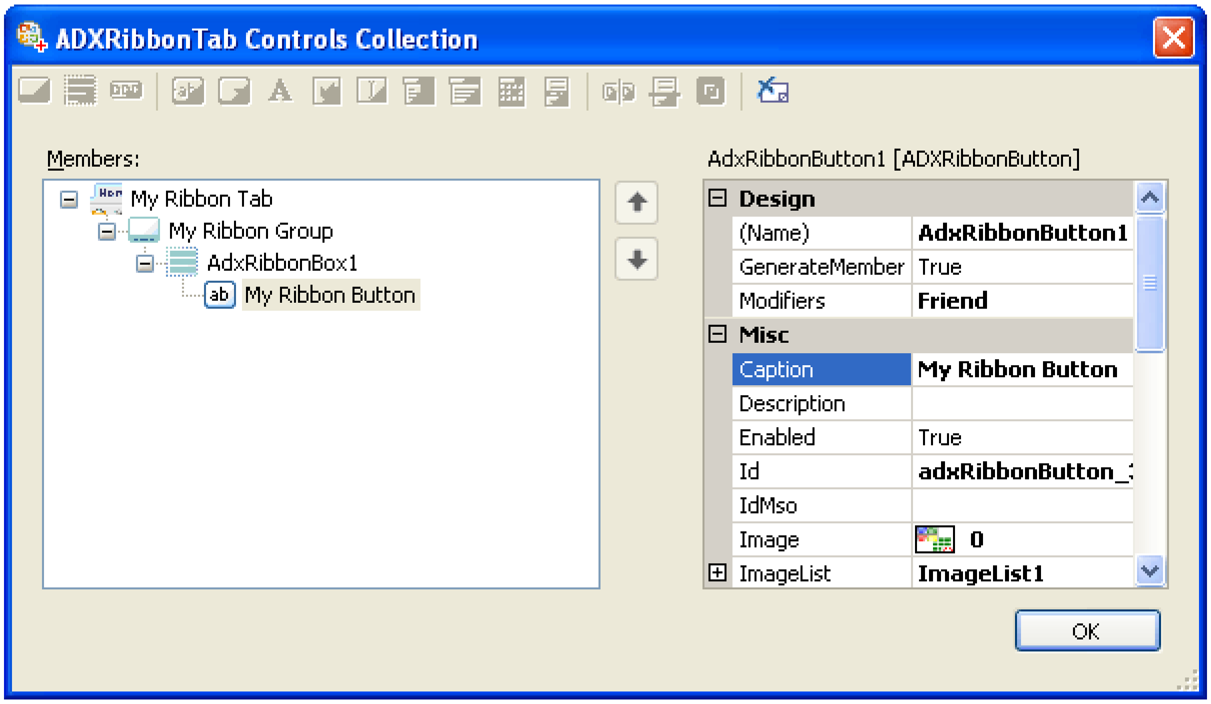
```
      Private Sub AdxKeyboardShortcut1_Action(ByVal sender As System.Object) _
         Handles AdxKeyboardShortcut1.Action
         MsgBox("You've pressed " + _
            CType(sender, AddinExpress.MSO.ADXKeyboardShortcut).ShortcutText)
      End Sub
```

Step #12 - Customizing the Outlook 2007 Ribbon User Interface

To add a new tab to the Ribbon, you use the Add Ribbon Tab command that adds an ADXRibbonTab component to the module.



In the Properties window, run the editor for the Controls collection of the tab. In the editor, use the toolbar buttons or context menu to add or delete Add-in Express components that form the Ribbon interface of your add-in. First, you add a Ribbon tab and change its caption to My Ribbon Tab. Then, you select the tab component, add a Ribbon group, and change its caption to My Ribbon Group. Next, you select the group, and add a button group. Finally, you select the button group and add a button. Set the button caption to My Ribbon Button. Use the ImageList and Image properties to set the icon for the button.



Design	
(Name)	AdxRibbonButton1
GenerateMember	True
Modifiers	Friend
Misc	
Caption	My Ribbon Button
Description	
Enabled	True
Id	adxRibbonButton_
IdMso	
Image	0
ImageList	ImageList1

Click OK, and, in the Properties window, find the newly added Ribbon button. Now add the event handler to the Click event of the button. Write the following code:

```
       Private Sub AdxRibbonButton1_OnClick(ByVal sender As System.Object, _
           ByVal control As AddinExpress.MSO.IRibbonControl, _
           ByVal pressed As System.Boolean) Handles AdxRibbonButton1.OnClick
           AdxCommandBarButton2_Click(Nothing)
       End Sub
```

Remember, the ADXRibbonTab Controls editor performs the XML-schema validation automatically, so from time to time you will run into the situation when you cannot add a control to some Ribbon level. It is a restriction of the Ribbon XML-schema.

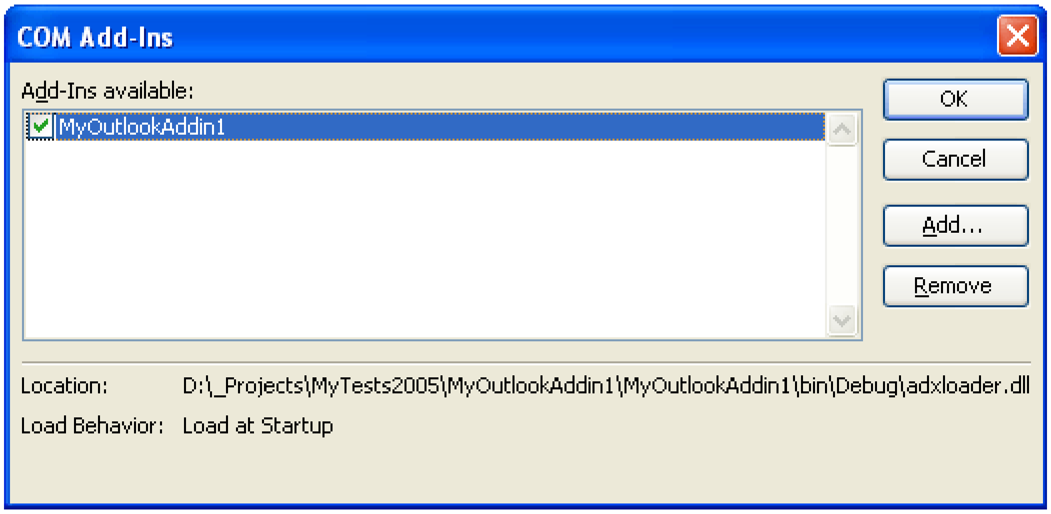
Note, unlike other Ribbon-based applications Outlook, has numerous ribbons. Please use the Ribbons property of your ADXRibbonTab components to specify the ribbons you customize with your tabs.

See also Office 2007 Ribbon Components.

Step #13 - Running the COM Add-in

Choose the Register Add-in Express Project item in the Build menu, then restart Outlook and find your option page(s), command bars, and controls.

You find your add-in in the COM Add-ins dialog:



MyOutlookAddin1

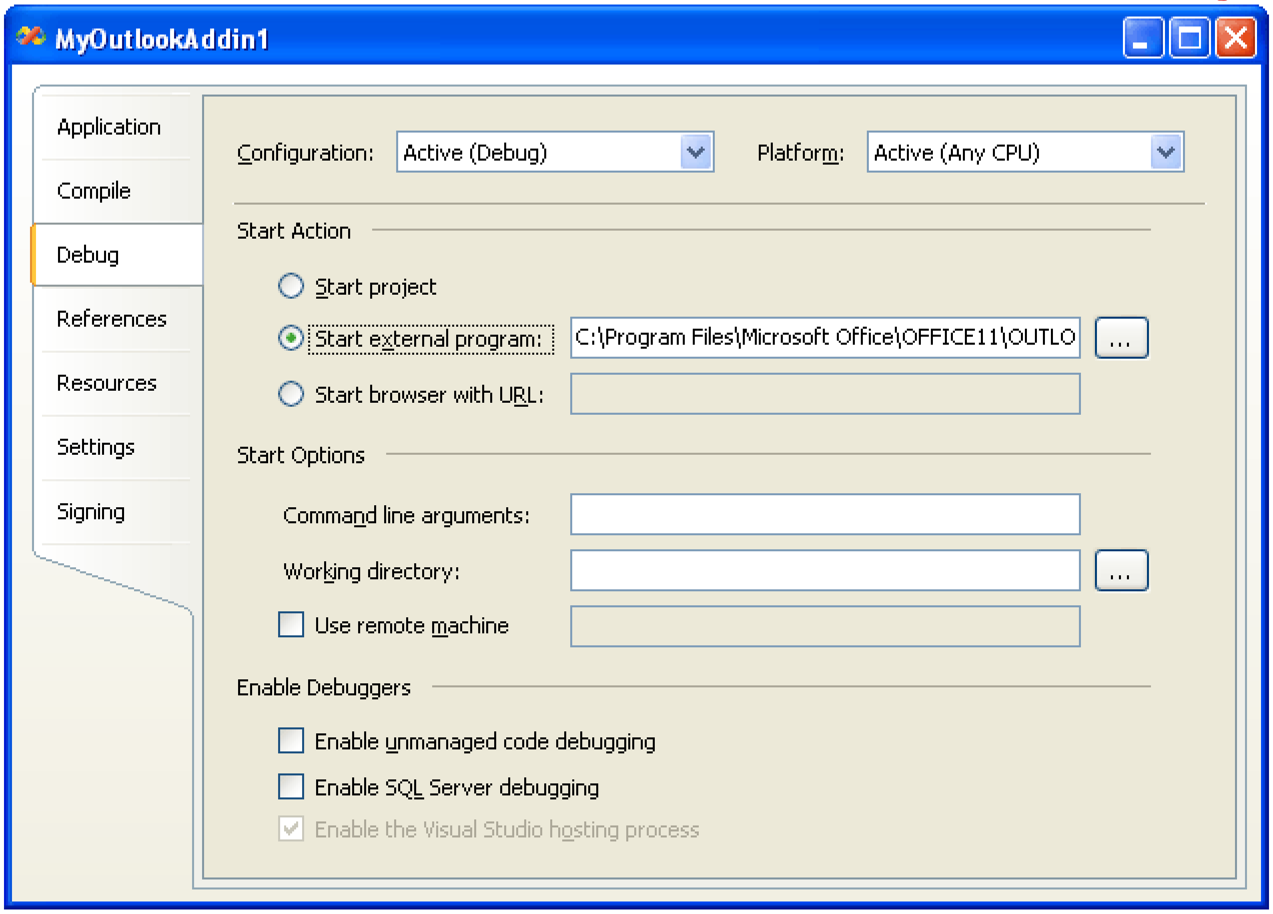
Location: D:_Projects\MyTests2005\MyOutlookAddin1\MyOutlookAddin1\bin\Debug\adxloader.dll

Load Behavior: Load at Startup

See also Add the COM Add-ins Command to a Toolbar or Menu.

Step #14 - Debugging the COM Add-in

To debug your add-in, just indicate the add-in host application as the Start Program in the Project Options window.



Configuration: Active (Debug)
Platform: Active (Any CPU)

Start Action

( ) Start project
(●) Start external program: C:\Program Files\Microsoft Office\OFFICE11\OUTLO
( ) Start browser with URL:

Start Options

Command line arguments:
Working directory:
☐ Use remote machine

Enable Debuggers

☐ Enable unmanaged code debugging
☐ Enable SQL Server debugging
✓ Enable the Visual Studio hosting process

However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

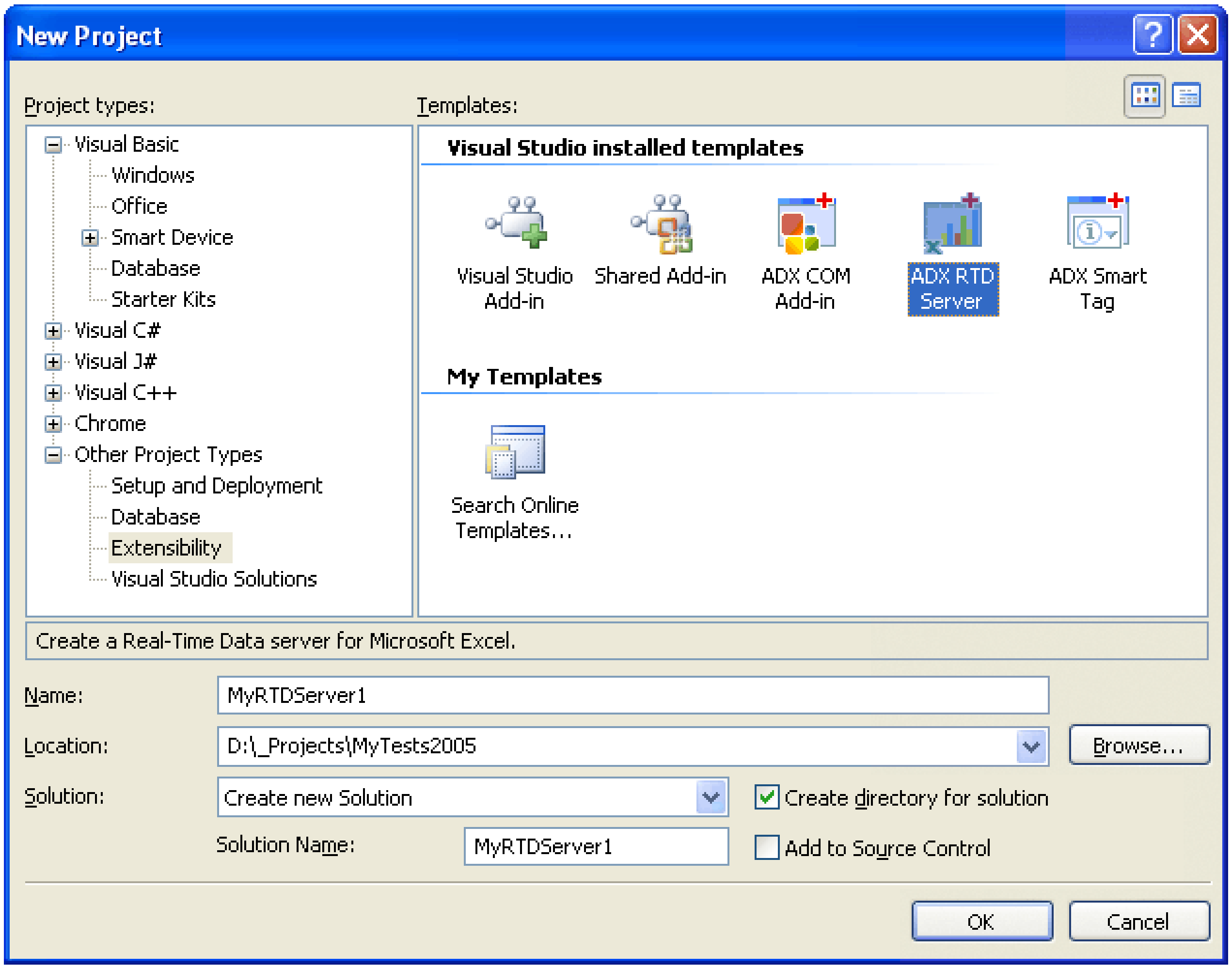
Step #15 - Deploying the COM Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in.

Your First Excel RTD Server

Step #1 - Creating a New Add-in Express RTD Server Project

Add-in Express adds the Add-in Express RTD Server project template to the Visual Studio IDE.



Create a Real-Time Data server for Microsoft Excel.

Name: MyRTDServer1

Location: D:_Projects\MyTests2005

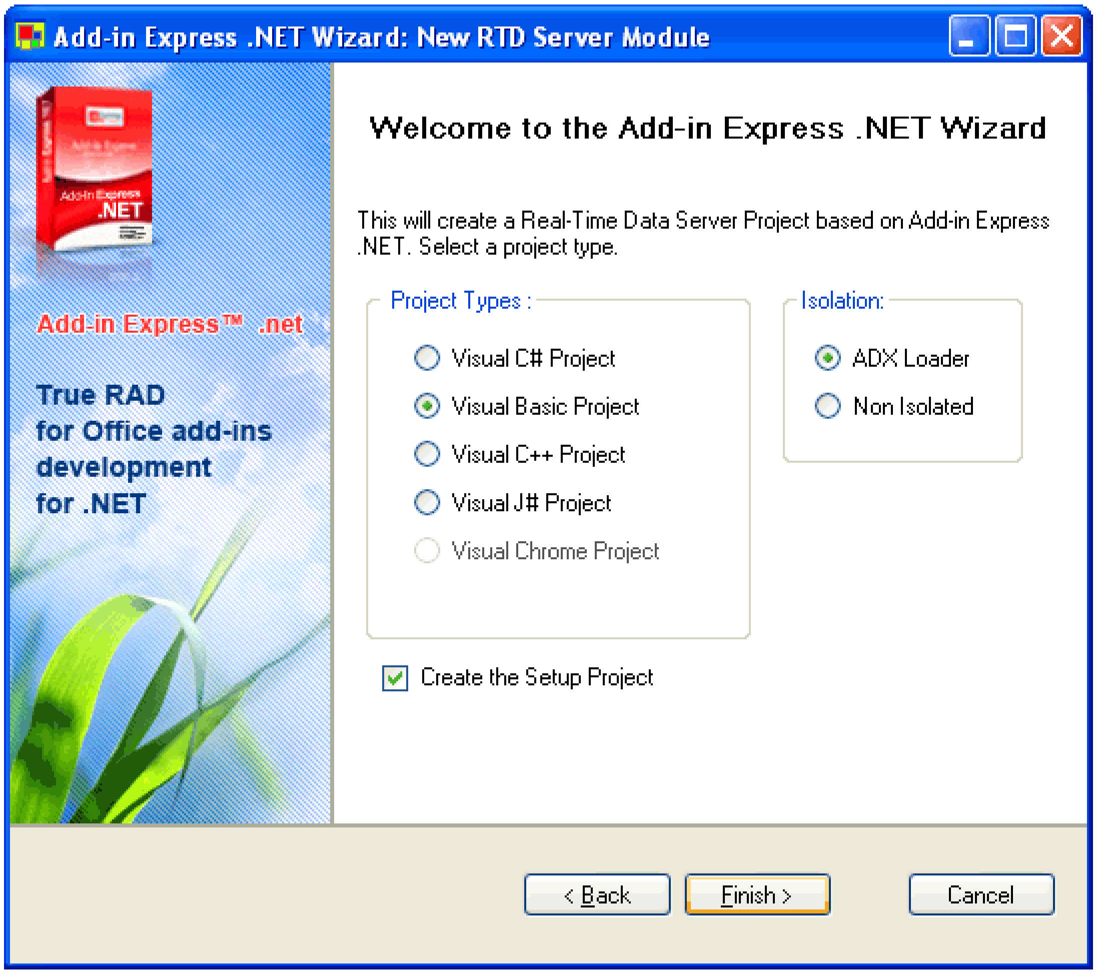
Solution: Create new Solution

Solution Name: MyRTDServer1

[✓] Create directory for solution

[ ] Add to Source Control

When you select the template and click OK, the Add-in Express RTD Server project wizard starts. In the wizard window, you choose the programming language and setup project options.


Add-in Express .NET Wizard: New RTD Server Module

Welcome to the Add-in Express .NET Wizard

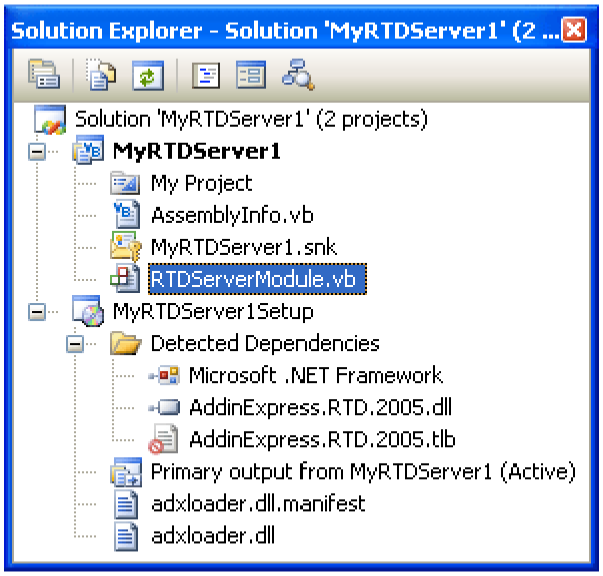
This will create a Real-Time Data Server Project based on Add-in Express .NET. Select a project type.

Project Types:
- Visual C# Project
- [✓ Visual Basic Project]
- Visual C++ Project
- Visual J# Project
- Visual Chrome Project

Isolation:
- [✓ ADX Loader]
- Non Isolated

[✓] Create the Setup Project

This VB.NET sample shows an Add-in Express RTD Server project with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.


Solution Explorer - Solution 'MyRTDServer1' (2 ...

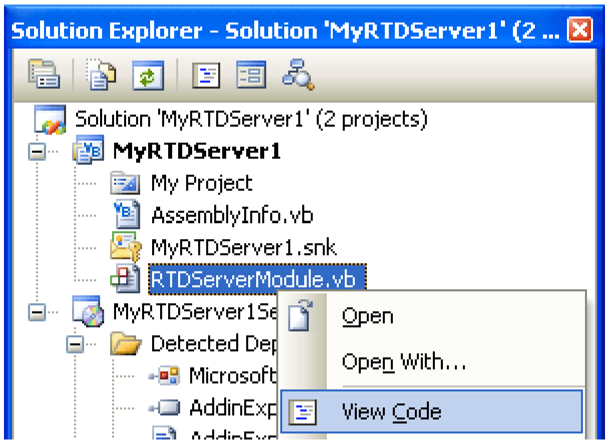
```
Solution 'MyRTDServer1' (2 projects)
  MyRTDServer1
    My Project
    AssemblyInfo.vb
    MyRTDServer1.snk
    RTDServerModule.vb
  MyRTDServer1Setup
    Detected Dependencies
      Microsoft .NET Framework
      AddinExpress.RTD.2005.dll
      AddinExpress.RTD.2005.tlb
    Primary output from MyRTDServer1 (Active)
    adxloader.dll.manifest
    adxloader.dll
```

The Add-in Express Project Wizard creates and opens the RTD Server solution in IDE. The solution includes the RTD Server project and the setup project.

The RTD Server project contains the RTDServerModule.vb (or RTDServerModule.cs) file discussed in the next step.

Step #2 - Add-in Express RTD Server Module

The RTDServerModule.vb (or RTDServerModule.cs) file is an RTD Server Module that is the core part of the RTD Server project. The module is a container for ADXRTDTopic components. It contains the RTDServerModule class, a descendant of the ADXRTDServerModule class that implements the IRtdServer interface and allows you to manage server's topics and their code. To review its source code, in the Solution Explorer window, right-click the RTDServerModule.vb (or RTDServerModule.cs) file and choose the View Code popup menu item.



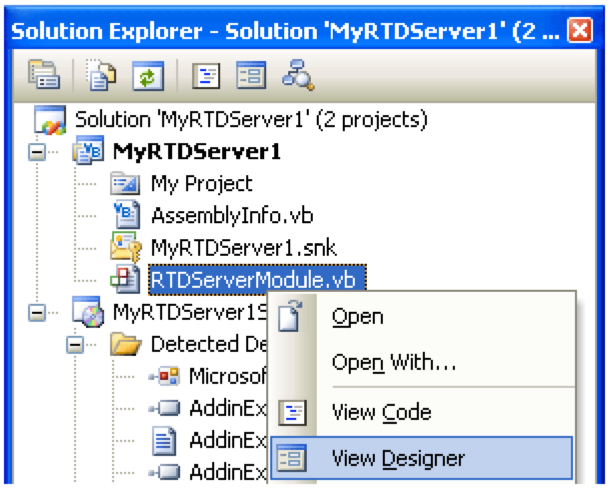
The code of RTDServerModule.vb is as follows:

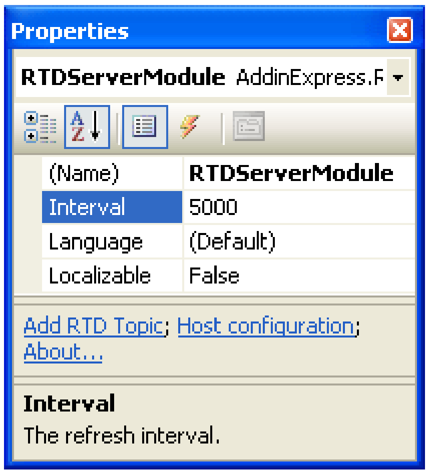
```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express RTD Server Module
<GuidAttribute("ACD23E21-2F2B-4C13-B894-6C74D8AD2EEE"), _
   ProgIdAttribute("MyRTDServer1.RTDServerModule")> _
Public Class RTDServerModule
   Inherits AddinExpress.RTD.ADXRTDServerModule

#Region " Component Designer generated code. "
   'Required by designer
   Private components As System.ComponentModel.IContainer

   'Required by designer - do not modify
   'the following method
   Private Sub InitializeComponent()
      Me.components = New System.ComponentModel.Container
   End Sub

#End Region

#Region " Add-in Express automatic code "

   'Required by Add-in Express - do not modify
   'the methods within this region

   Public Overrides Function GetContainer() _
      As System.ComponentModel.IContainer
      If components Is Nothing Then
         components = New System.ComponentModel.Container
```

```
        End If
        GetContainer = components
     End Function

     <ComRegisterFunctionAttribute()> _
     Public Shared Sub RTDServerRegister(ByVal t As Type)
        AddinExpress.RTD.ADXRTDServerModule.ADXRTDServerRegister(t)
     End Sub

     <ComUnregisterFunctionAttribute()> _
     Public Shared Sub RTDServerUnregister(ByVal t As Type)
        AddinExpress.RTD.ADXRTDServerModule.ADXRTDServerUnregister(t)
     End Sub

  #End Region

     Public Sub New()
        MyBase.New()
        'This call is required by the Component Designer
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
     End Sub
  End Class
```

Step #3 - Add-in Express RTD Server Designer

The Add-in Express RTD Server Designer allows setting RTD Server properties and adding components to the module. In the Solution Explorer window, right-click the RTDServerModule.vb (or RTDServerModule.cs) file and choose the View Designer popup menu item.

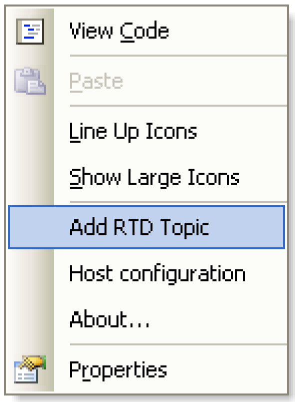



Properties
RTDServerModule AddinExpress.F
Add RTD Topic; Host configuration; About...

(Name)	RTDServerModule
Interval	5000
Language	(Default)
Localizable	False

Interval
The refresh interval.

In the Properties window, you set properties of your RTD Server (see RTD Servers).



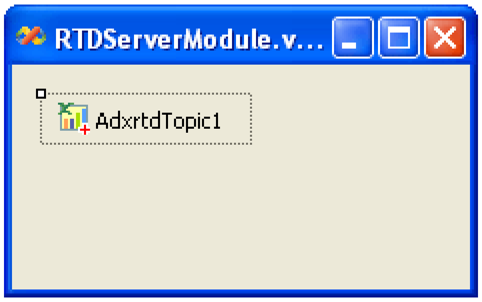
To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.

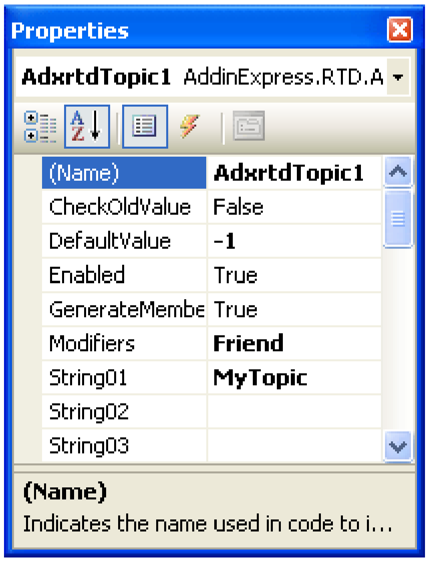
The only command available for the module adds the RTD Topic component to the module (see RTD Topic).

Step #4 - Adding and Handling a New Topic

To add a new topic to your RTD Server, you use the Add RTD Topic command that adds a new ADXRTDTopic component to the RTD Server Module (see RTD Topic).

Select the newly added component and, in the Properties window, specify the topic using the String## properties.





(Name)	AdxrtdTopic1
CheckOldValue	False
DefaultValue	-1
Enabled	True
GenerateMembe	True
Modifiers	Friend
String01	MyTopic
String02	
String03	

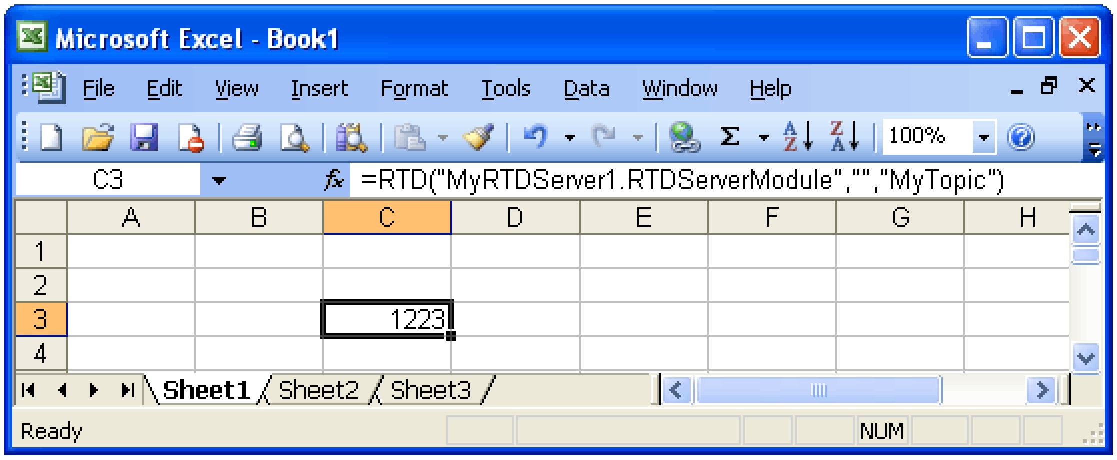
(Name)
Indicates the name used in code to i...

To handle the RefreshData event of the RTD Topic, add the RefreshData event handler and write your code to handle the event:

```
Private Function AdxrtdTopic1_RefreshData( _
   ByVal sender As System.Object) As System.Object _
      Handles AdxrtdTopic1.RefreshData
   Dim Rnd As New System.Random
   Return Rnd.Next(2000)
End Function
```

Step #5 – Running the RTD Server

Choose the Register Add-in Express Project item in the Build menu, restart Excel, and enter the RTD function to a cell.

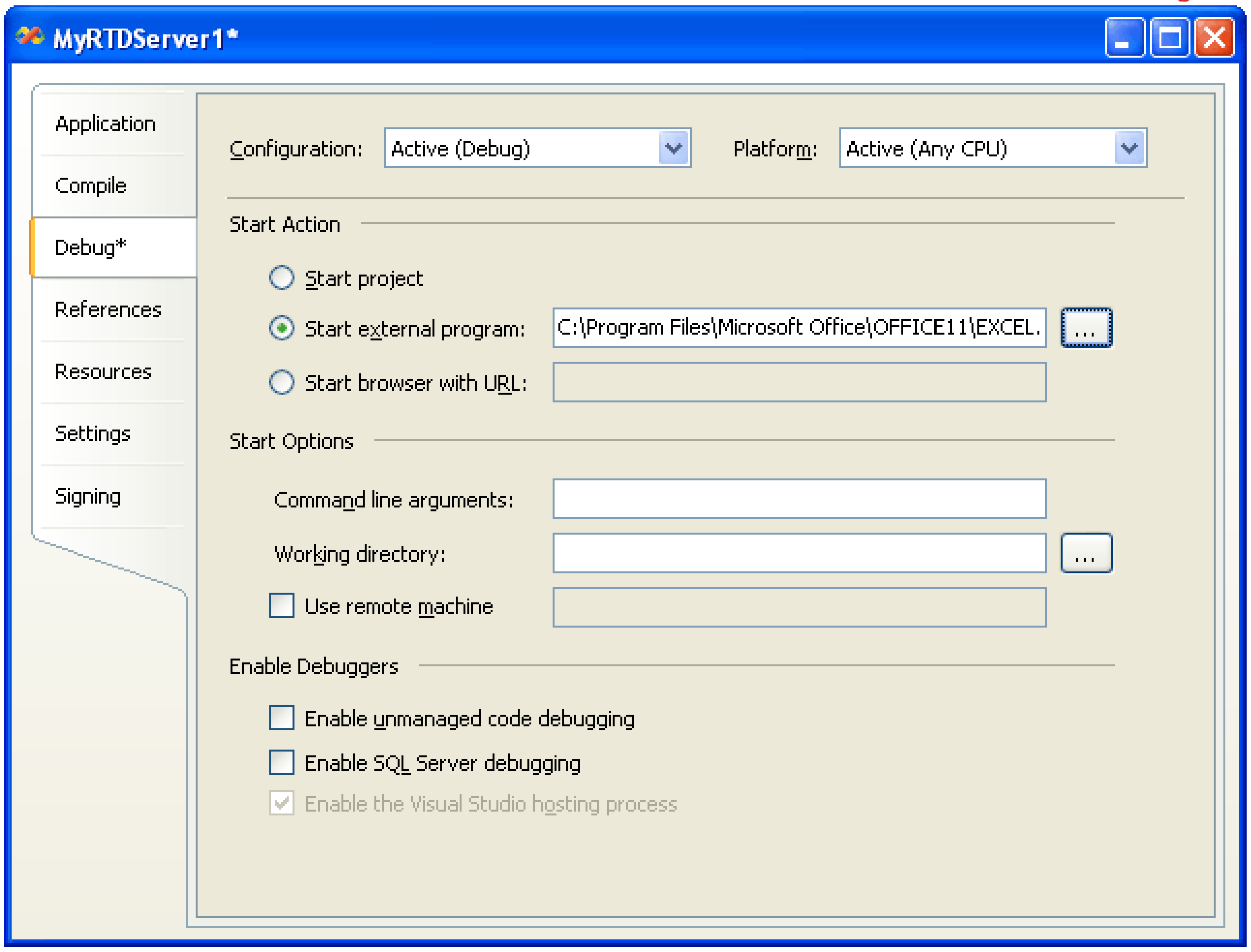


```
=RTD("MyRTDServer1.RTDServerModule","","MyTopic")
```

	A	B	C	D	E	F	G	H
1								
2								
3			1223					
4								

Step #6 – Debugging the RTD Server

To debug your RTD Server, just indicate Excel as the Start Program in the Project Options window.



Configuration: Active (Debug)
Platform: Active (Any CPU)

Start Action
- Start project
- [✓ Start external program:] C:\Program Files\Microsoft Office\OFFICE11\EXCEL.
- Start browser with URL:

Start Options
- Command line arguments:
- Working directory:
- Use remote machine

Enable Debuggers
- Enable unmanaged code debugging
- Enable SQL Server debugging
- [✓] Enable the Visual Studio hosting process

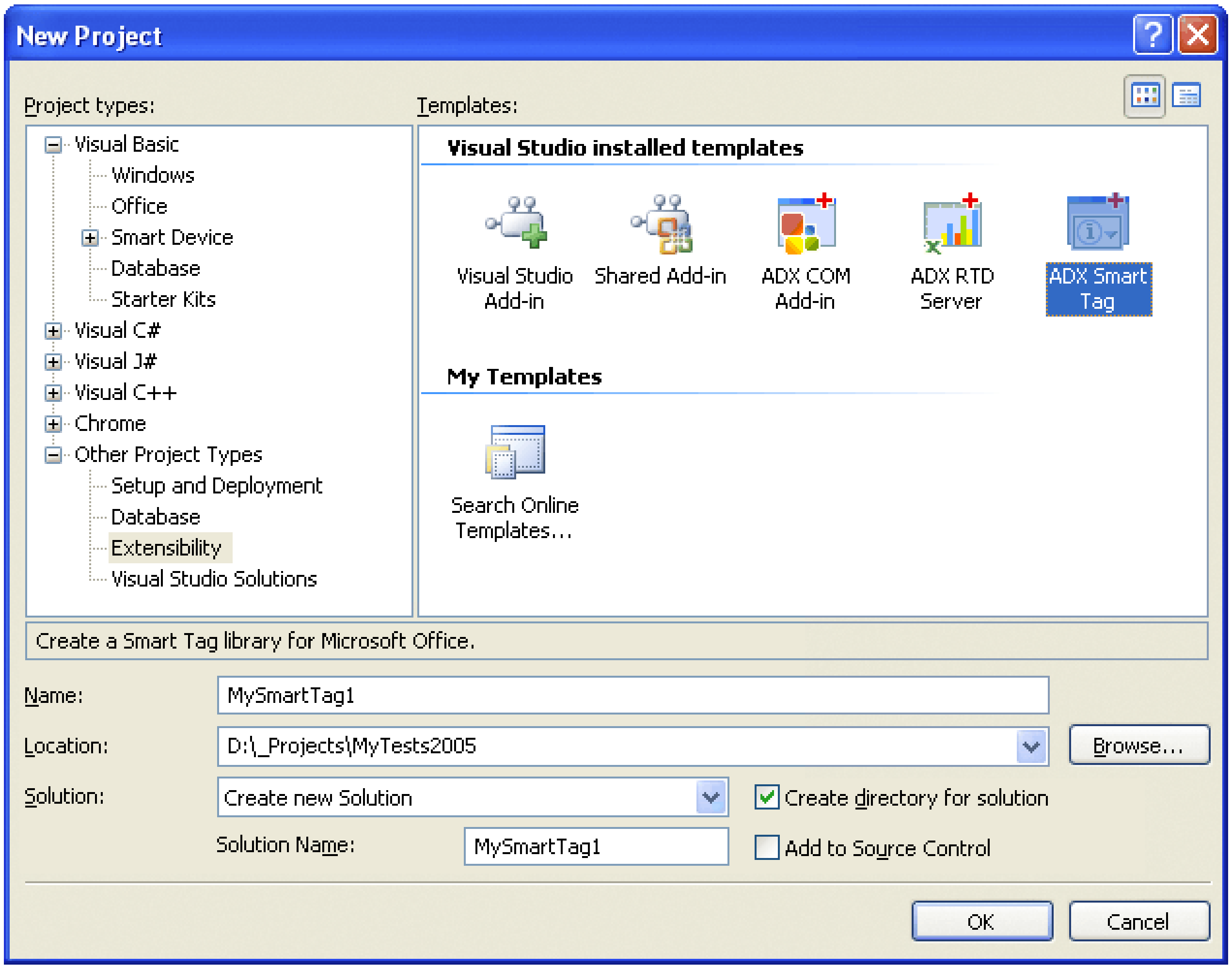
Step #7 - Deploying the RTD Server

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the RTD Server.

Your First Smart Tag

Step #1 - Creating a New Smart Tag Library Project

Add-in Express adds the Add-in Express Smart Tag project template to the Visual Studio IDE.



Create a Smart Tag library for Microsoft Office.

Name: MySmartTag1

Location: D:_Projects\MyTests2005

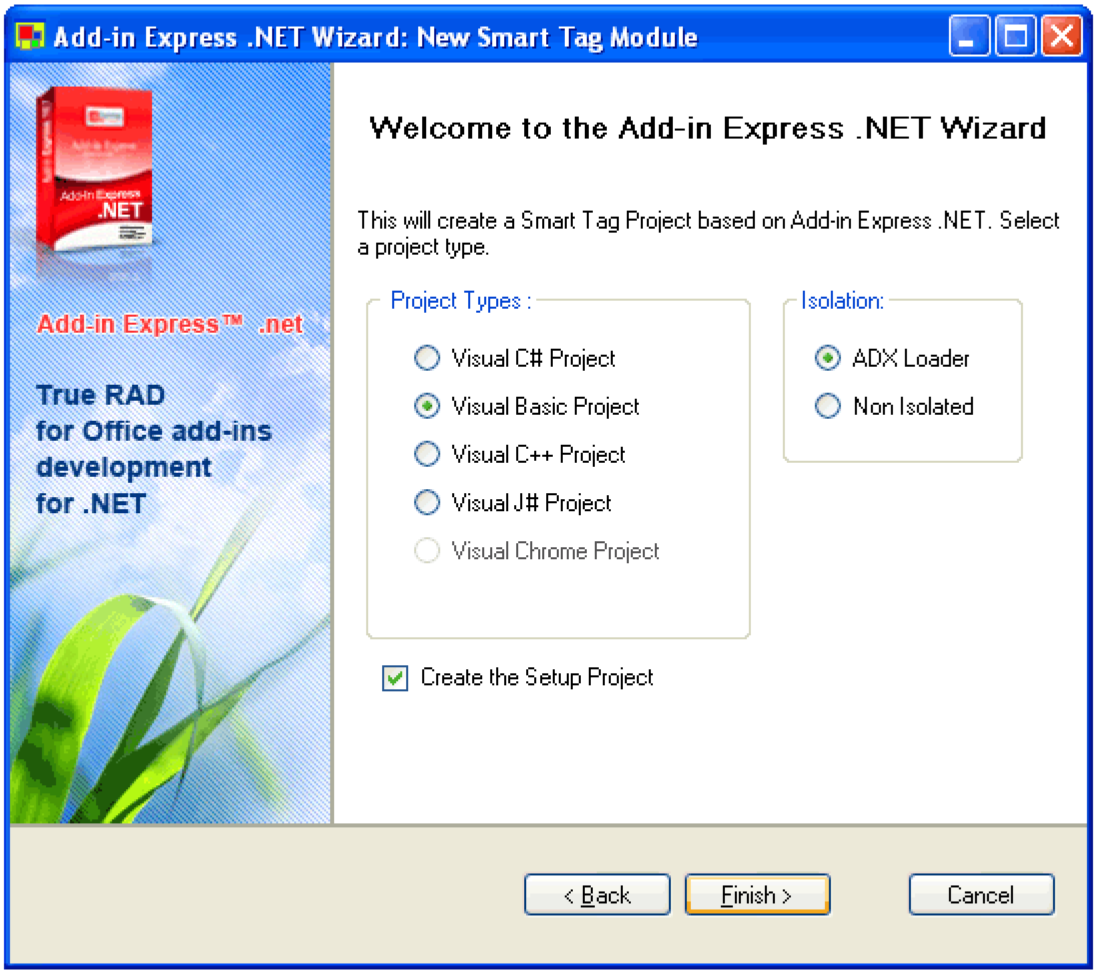
Solution: Create new Solution

[✓ Create directory for solution]

Solution Name: MySmartTag1

Add to Source Control

When you select the template and click OK, the Add-in Express Smart Tag project wizard starts. In the wizard window, you choose the programming language and setup project options.


Add-in Express .NET Wizard: New Smart Tag Module
Add-in Express™ .net
True RAD for Office add-ins development for .NET

Welcome to the Add-in Express .NET Wizard

This will create a Smart Tag Project based on Add-in Express .NET. Select a project type.

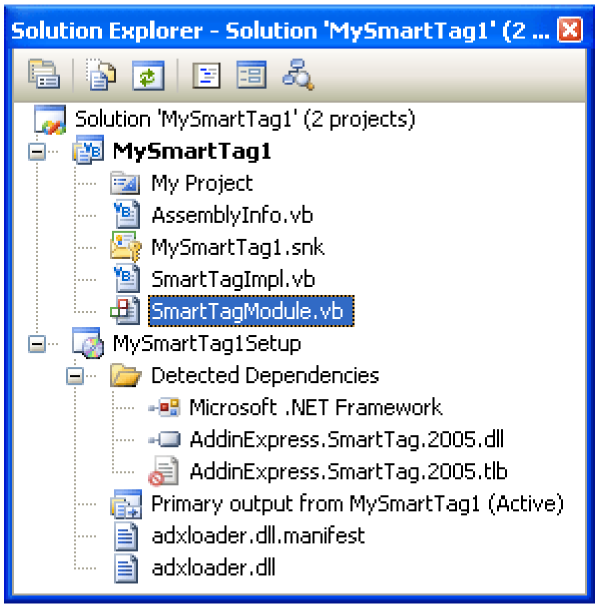
Project Types :
- Visual C# Project
- [✓ Visual Basic Project]
- Visual C++ Project
- Visual J# Project
- Visual Chrome Project

Isolation:
- [✓ ADX Loader]
- Non Isolated

[✓] Create the Setup Project

< Back | Finish > | Cancel

This VB.NET sample shows an Add-in Express SmartTag project with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.



```
Solution 'MySmartTag1' (2 projects)
  MySmartTag1
    My Project
    AssemblyInfo.vb
    MySmartTag1.snk
    SmartTagImpl.vb
    SmartTagModule.vb
  MySmartTag1Setup
    Detected Dependencies
      Microsoft .NET Framework
      AddinExpress.SmartTag.2005.dll
      AddinExpress.SmartTag.2005.tlb
    Primary output from MySmartTag1 (Active)
    adxloader.dll.manifest
    adxloader.dll
```

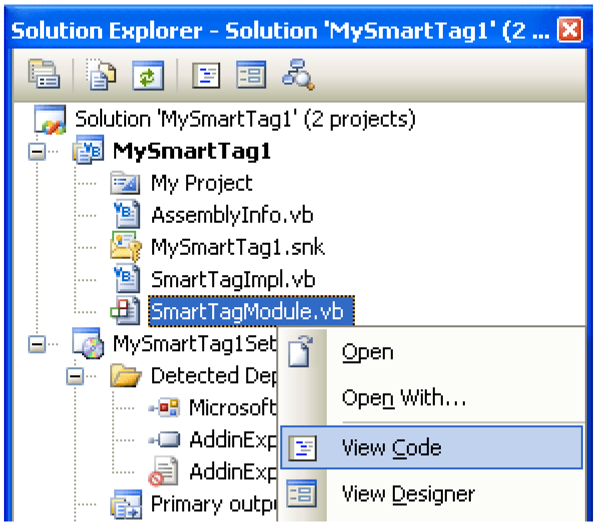
The Add-in Express Project Wizard creates and opens the Smart Tag solution in IDE. The solution includes the SmartTag project and the setup project.

Note

Do not delete the SmartTagImpl.vb file required by the Add-in Express implementation of the Smart Tag technology. Usually, you do not need to modify it.

The Smart Tag project contains the SmartTagModule.vb (or SmartTagModule.cs) file discussed in the next step.

Step #2 - Add-in Express Smart Tag Module



The SmartTagModule.vb (or SmartTagModule.cs) file is a Smart Tag Module that is the core part of the Smart Tag project. The Smart Tag module is a container for ADXSmartTag components. It contains the SmartTagModule class, a descendant of the ADXSmartTagModule class that implements the interfaces required by the Smart Tag technology and allows managing smart tags and their code. To review its source code, in the Solution Explorer window, right-click the SmartTagModule.vb (or SmartTagModule.cs) file and choose the View Code popup menu item.

The smart tag module contains the following code:

```
Imports System.Runtime.InteropServices
Imports System.ComponentModel

'Add-in Express Smart Tag Module
<GuidAttribute("6FFEFD7D-FD3C-47EE-AE67-A84DE4AD4E7A"),
    ProgIdAttribute("MySmartTag1.SmartTagModule")> _
Public Class SmartTagModule
    Inherits AddinExpress.SmartTag.ADXSmartTagModule

#Region " Component Designer generated code. "
   'Required by designer
   Private components As System.ComponentModel.IContainer

   'Required by designer - do not modify
   'the following method
```

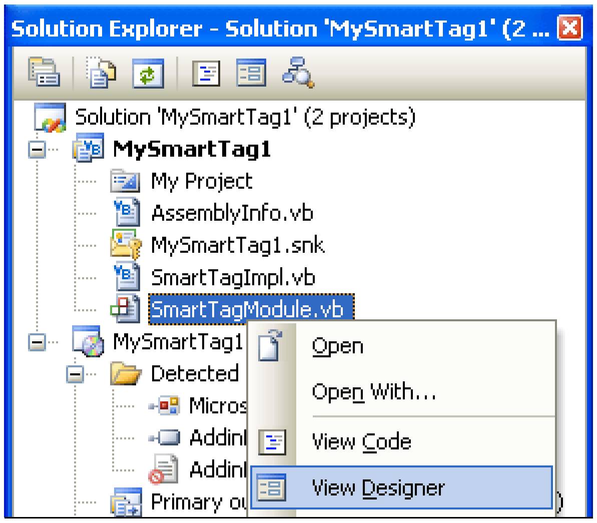
```
    Private Sub InitializeComponent()
       Me.components = New System.ComponentModel.Container
       '
       'SmartTagModule
       '
       Me.NamespaceURI = "mysmarttag1/smarttagmodule"
       Me.LibraryName = New AddinExpress.SmartTag.ADXLocalizedList
       Me.LibraryName.Add(0, "MySmartTag1")
       Me.Description = New AddinExpress.SmartTag.ADXLocalizedList
       Me.Description.Add(0, "This is a description")

    End Sub

 #End Region

 #Region " Add-in Express automatic code "

    'Required by Add-in Express - do not modify
    'the methods within this region

    Public Overrides Function GetContainer() As
       System.ComponentModel.IContainer
       If components Is Nothing Then
          components = New System.ComponentModel.Container
       End If
       GetContainer = components
    End Function

    <ComRegisterFunctionAttribute()> _
    Public Shared Sub SmartTagRegister(ByVal t As Type)
       AddinExpress.SmartTag.ADXSmartTagModule.ADXSmartTagRegister(t, _
          System.Type.GetType("MySmartTag1.SmartTagRecognizerImpl"), _
          System.Type.GetType("MySmartTag1.SmartTagActionImpl"))
    End Sub

    <ComUnregisterFunctionAttribute()> _
    Public Shared Sub SmartTagUnregister(ByVal t As Type)
       AddinExpress.SmartTag.ADXSmartTagModule.ADXSmartTagUnregister(t, _
          System.Type.GetType("MySmartTag1.SmartTagRecognizerImpl"), _
          System.Type.GetType("MySmartTag1.SmartTagActionImpl"))
    End Sub

 #End Region

    Public Sub New()
       MyBase.New()
```

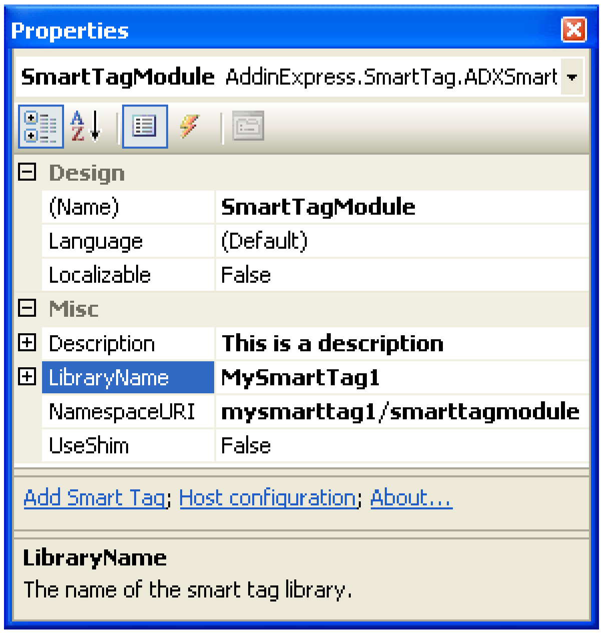
```
        'This call is required by the Component Designer
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub
End Class
```

Step #3 - Add-in Express Smart Tag Designer

The Add-in Express Smart Tag Designer allows setting Smart Tag properties and adding components to the module.

In the Solution Explorer window, right-click the SmartTagModule.vb (or SmartTagModule.cs) file and choose the View Designer popup menu item.

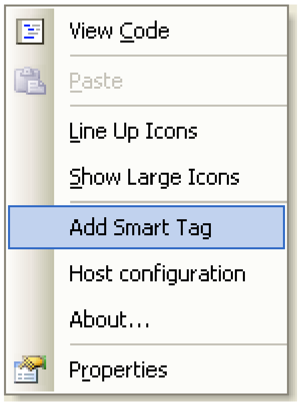



Properties
SmartTagModule AddinExpress.SmartTag.ADXSmart
Add Smart Tag; Host configuration; About...

Design	
(Name)	SmartTagModule
Language	(Default)
Localizable	False
Misc	
Description	This is a description
LibraryName	MySmartTag1
NamespaceURI	mysmarttag1/smarttagmodule
UseShim	False

LibraryName
The name of the smart tag library.

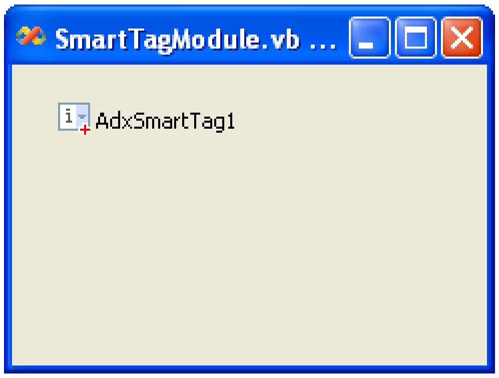
In the Properties window, you set properties of your Smart Tag module (see Smart Tags).



To add an Add-in Express Component to the module, you use an appropriate command in the Properties window, or you can right-click the designer surface and choose the same command in the context menu.

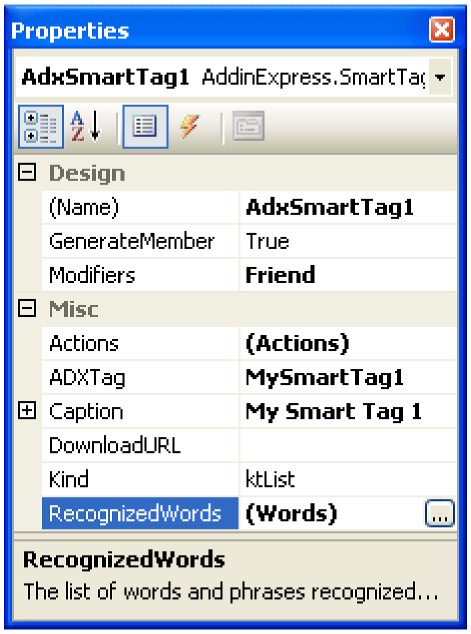
The only command available for the module adds the Smart Tag component to the module.

Step #4 - Adding a New Smart Tag



To add a new Smart Tag to your library, use the Add Smart Tag command that adds a new ADXSmartTag component to the Smart Tag Module (see Smart Tag).

Select the newly added component and, in the Properties window, specify the caption for the added smart tag in the Caption property. The value of this property will become a caption of the smart tag context menu. Also, specify the phrase(s) recognizable by the smart tag in the RecognizedWords string collection.

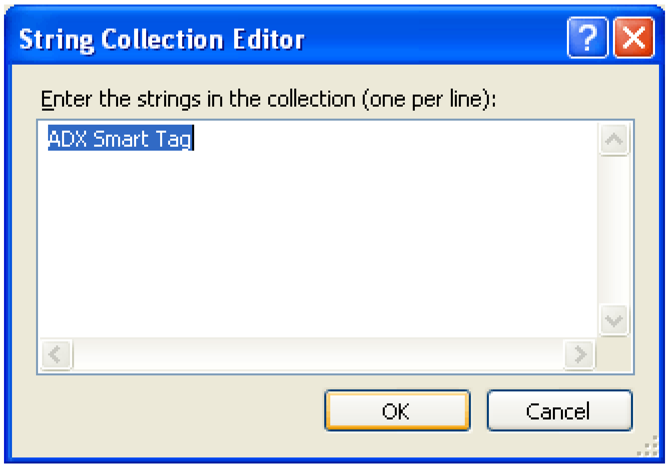

Properties

AdxSmartTag1 AddinExpress.SmartTag

Design	
(Name)	AdxSmartTag1
GenerateMember	True
Modifiers	Friend
Misc	
Actions	(Actions)
ADXTag	MySmartTag1
Caption	My Smart Tag 1
DownloadURL	
Kind	ktList
RecognizedWords	(Words)

RecognizedWords
The list of words and phrases recognized...

Say, in this sample Smart Tag, the words are as follows:

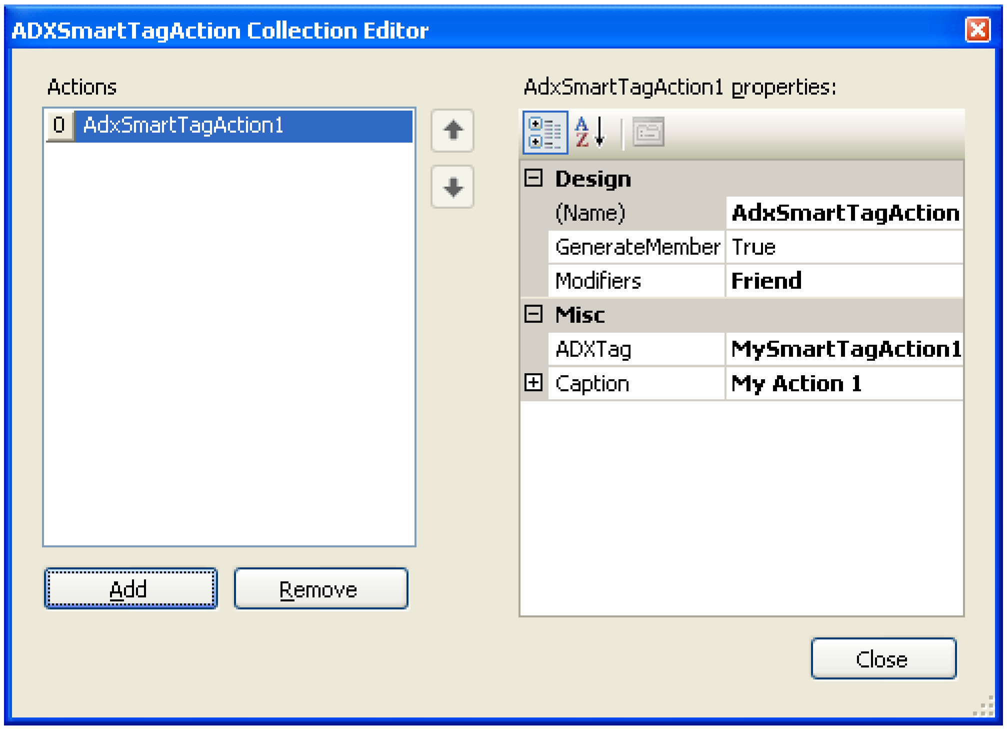

String Collection Editor

Enter the strings in the collection (one per line):

```
ADX Smart Tag
```

Step #5 - Adding and Handling Smart Tag Actions

Now you add smart tag actions to your smart tag context menu. To add a new smart tag action, add an item to the Actions collection of the smart tag.


ADXSmartTagAction Collection Editor
Actions
0 AdxSmartTagAction1
Add
Remove
AdxSmartTagAction1 properties:
Close

Design	
(Name)	AdxSmartTagAction
GenerateMember	True
Modifiers	Friend
Misc	
ADXTag	MySmartTagAction1
Caption	My Action 1

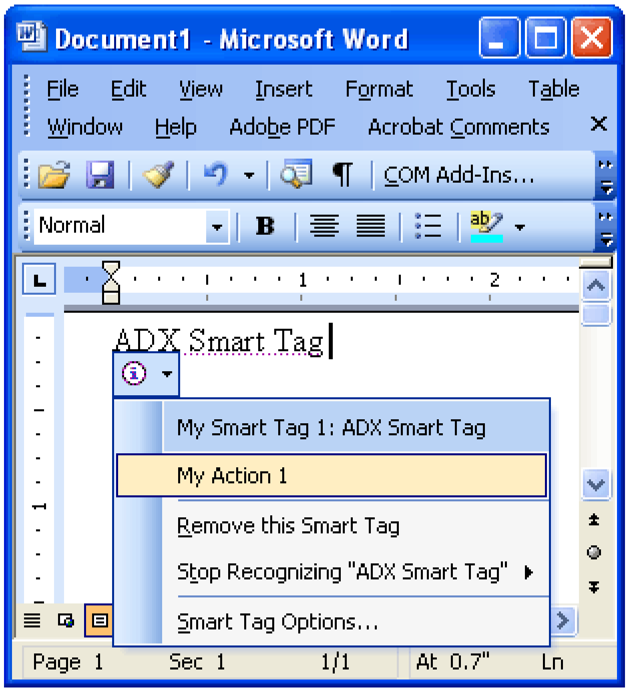
The value of the action's Caption property will become the caption of the appropriate item in the smart tag context menu.

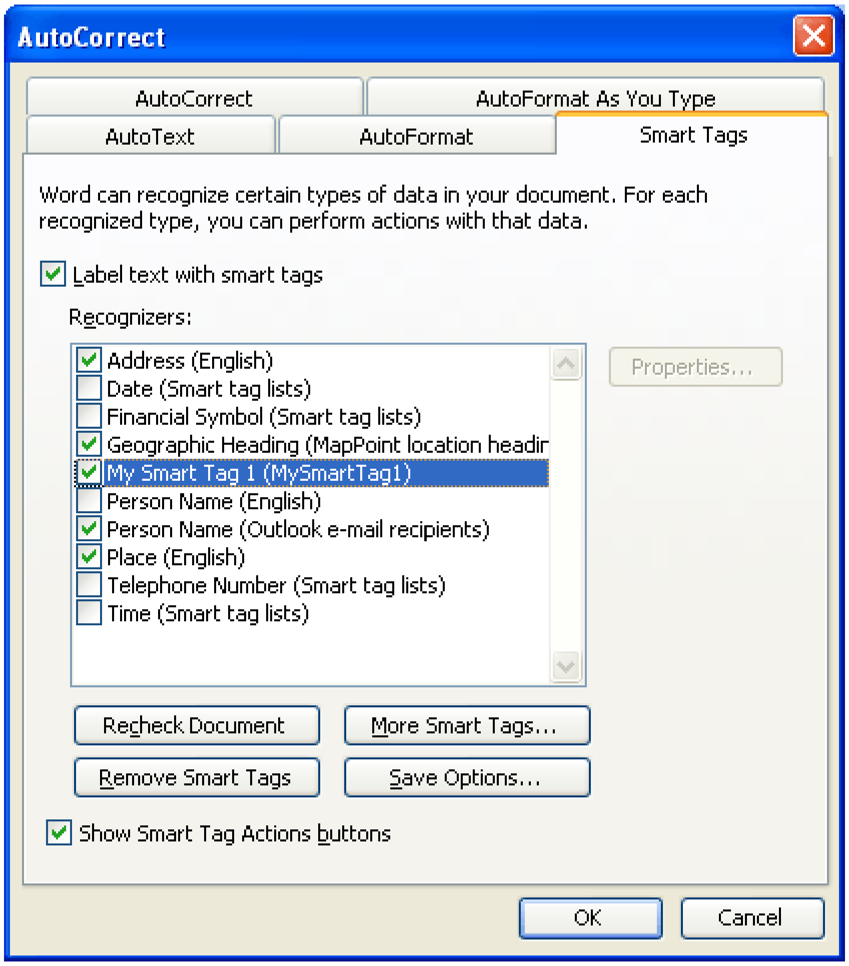
To handle the Click event of the action, close the Actions collection editor, and, in the Properties window, select the added action. Then add the Click event handler and write your code to handle the event:

```
Private Sub AdxSmartTagAction1_Click(ByVal sender As System.Object, _
   ByVal e As AddinExpress.SmartTag.ADXSmartTagActionEventArgs) _
      Handles AdxSmartTagAction1.Click
   MsgBox("Recognized text is '" + e.Text + "'!")
End Sub
```

Step #6 - Running Your Smart Tag

Choose the Register Add-in Express Project item in the Build menu. Restart Word, put the words recognizable by your smart tag into a document, and see if the smart tag works.


Document1 - Microsoft Word
File Edit View Insert Format Tools Table
Window Help Adobe PDF Acrobat Comments
COM Add-Ins...
Normal
ADX Smart Tag
My Smart Tag 1: ADX Smart Tag
My Action 1
Remove this Smart Tag
Stop Recognizing "ADX Smart Tag"
Smart Tag Options...
Page 1 Sec 1 1/1 At 0.7" Ln



Word can recognize certain types of data in your document. For each recognized type, you can perform actions with that data.

[✓] Label text with smart tags

Recognizers:

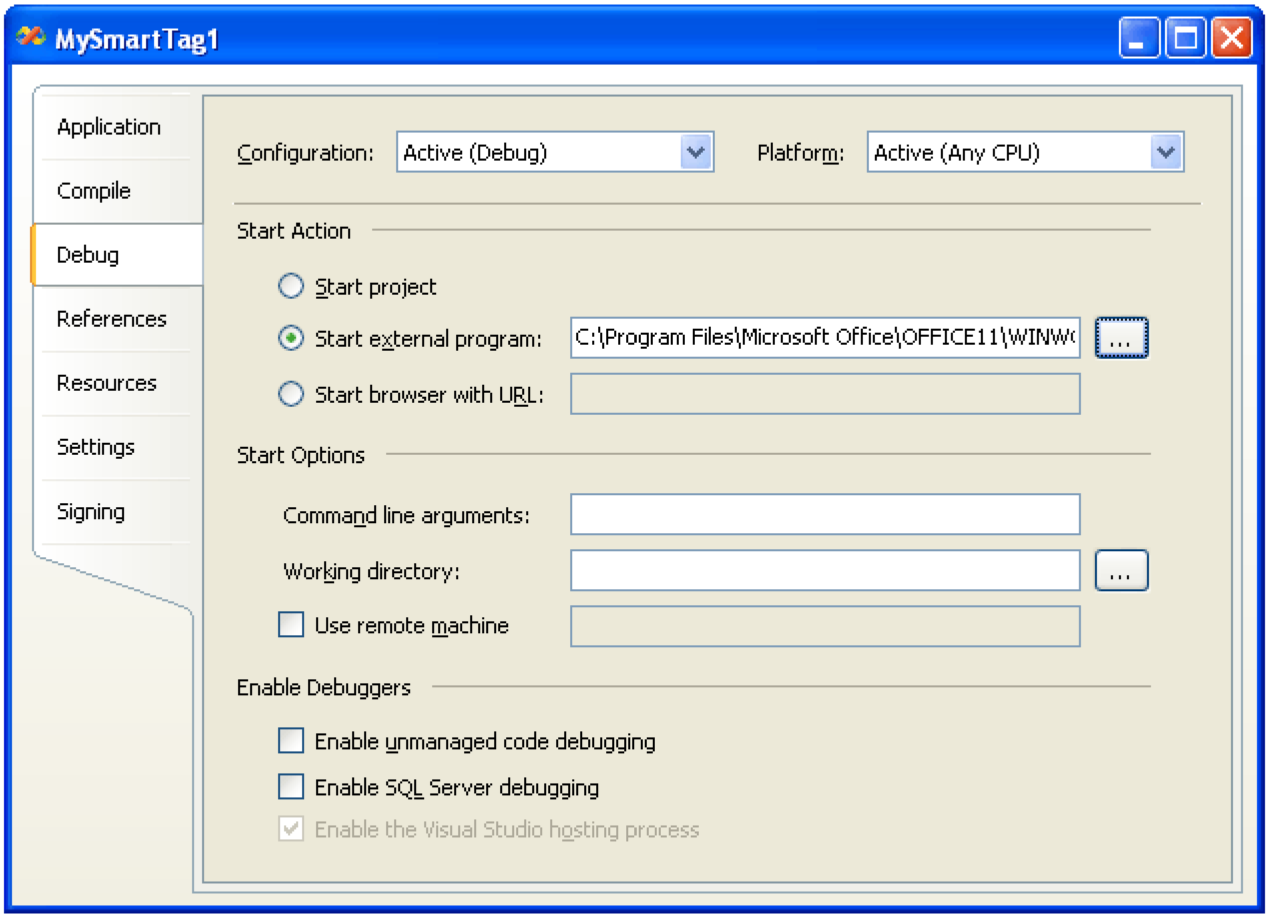
- [✓] Address (English)
- [ ] Date (Smart tag lists)
- [ ] Financial Symbol (Smart tag lists)
- [✓] Geographic Heading (MapPoint location headir
- [✓] My Smart Tag 1 (MySmartTag1)
- [ ] Person Name (English)
- [✓] Person Name (Outlook e-mail recipients)
- [✓] Place (English)
- [ ] Telephone Number (Smart tag lists)
- [ ] Time (Smart tag lists)

[✓] Show Smart Tag Actions buttons

In Office 2003-2003, choose the Tools | AutoCorrect menu item and find your smart tag on the Smart Tags tab. In Office2007, the path to this dialog is as follows: Office button | Word Options | Add-ins | Mange | Manage Smart Tags | Go.

Step #7 – Debugging the Smart Tag

To debug your Smart Tag, just indicate the add-in host application as the Start Program in the Project Options window.



Configuration: Active (Debug)
Platform: Active (Any CPU)

Start Action

Start project
[✓ Start external program:] C:\Program Files\Microsoft Office\OFFICE11\WINW(
Start browser with URL:

Start Options

Command line arguments:
Working directory:
Use remote machine

Enable Debuggers

Enable unmanaged code debugging
Enable SQL Server debugging
[✓ Enable the Visual Studio hosting process]

However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

Step #8 - Deploying the Smart Tag

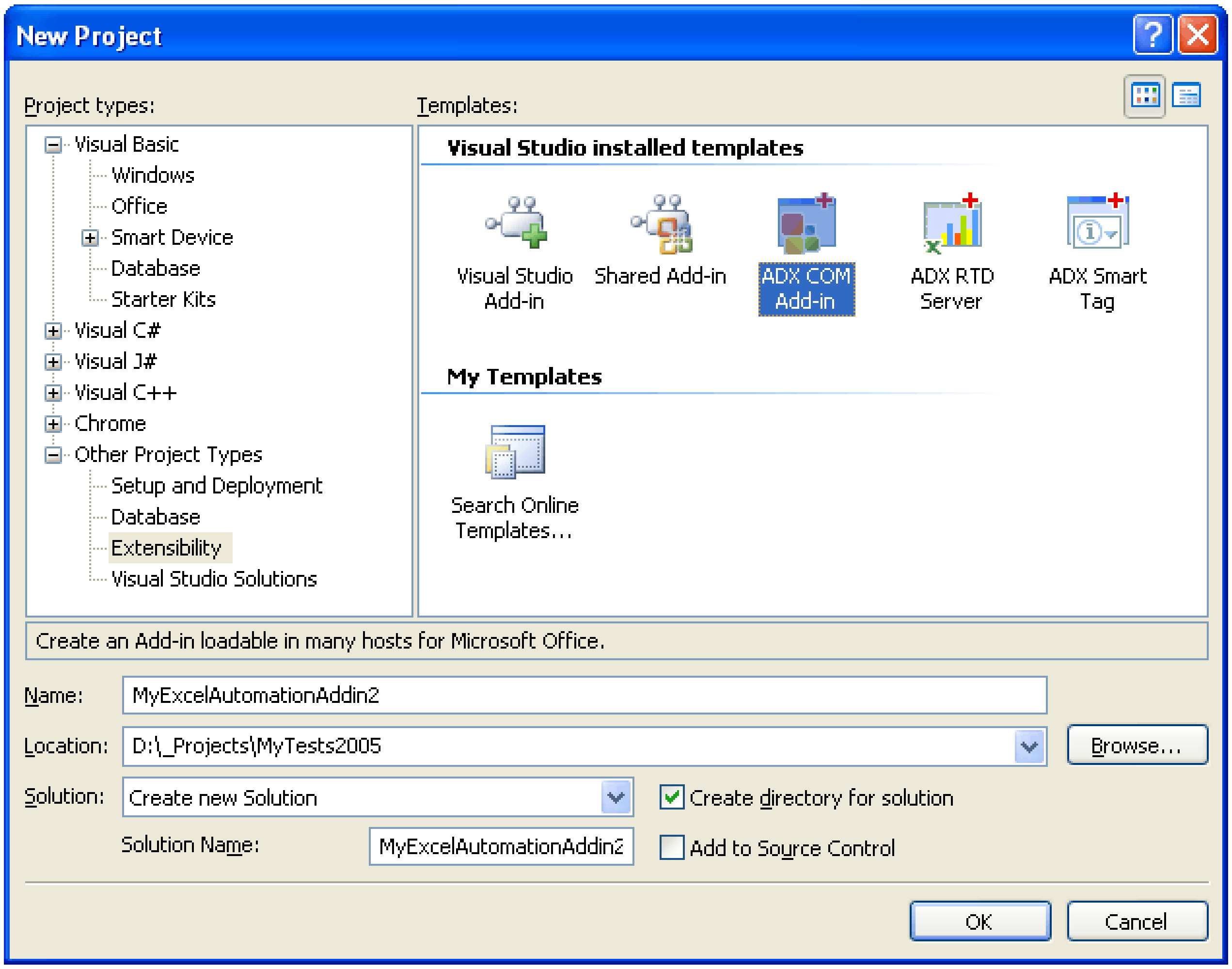
Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the Smart Tag.

Your First Excel Automation Add-in

The fact is that Excel Automation Add-ins do not differ from COM Add-ins except for the registration in the registry. That's why Add-n Express bases Excel Automation Add-in projects on Add-in Express COM Add-in projects.

Step #1 - Creating a New COM Add-in Project

Add-in Express adds the Add-in Express COM Add-in project template to the Visual Studio IDE.



Create an Add-in loadable in many hosts for Microsoft Office.

Name: MyExcelAutomationAddin2

Location: D:_Projects\MyTests2005

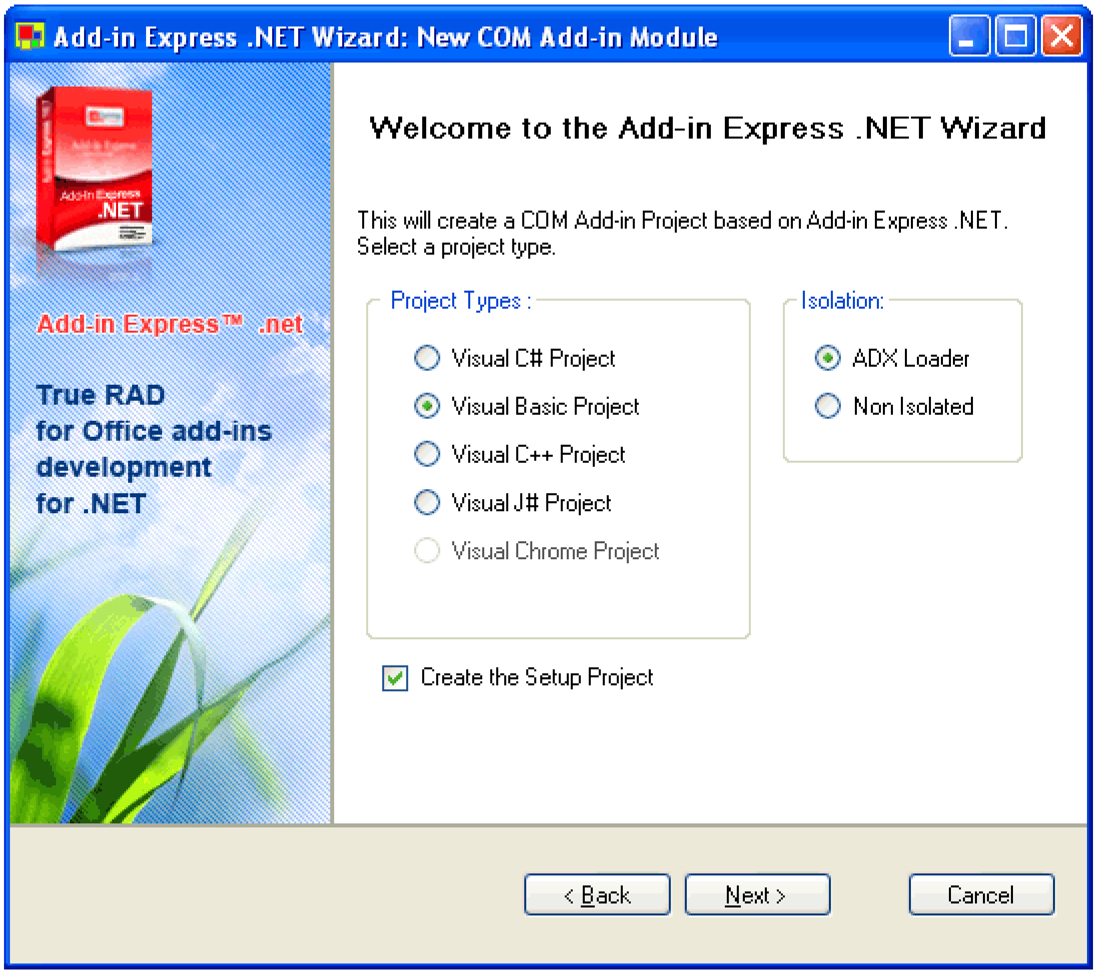
Solution: Create new Solution

[✓] Create directory for solution

Solution Name: MyExcelAutomationAddin2

[ ] Add to Source Control

When you select the template and click OK, the Add-in Express COM Add-in project wizard starts. In the wizard windows, you choose the programming language, setup project options, and supported applications of your add-in.


Add-in Express .NET Wizard: New COM Add-in Module

Welcome to the Add-in Express .NET Wizard

This will create a COM Add-in Project based on Add-in Express .NET. Select a project type.

Project Types :

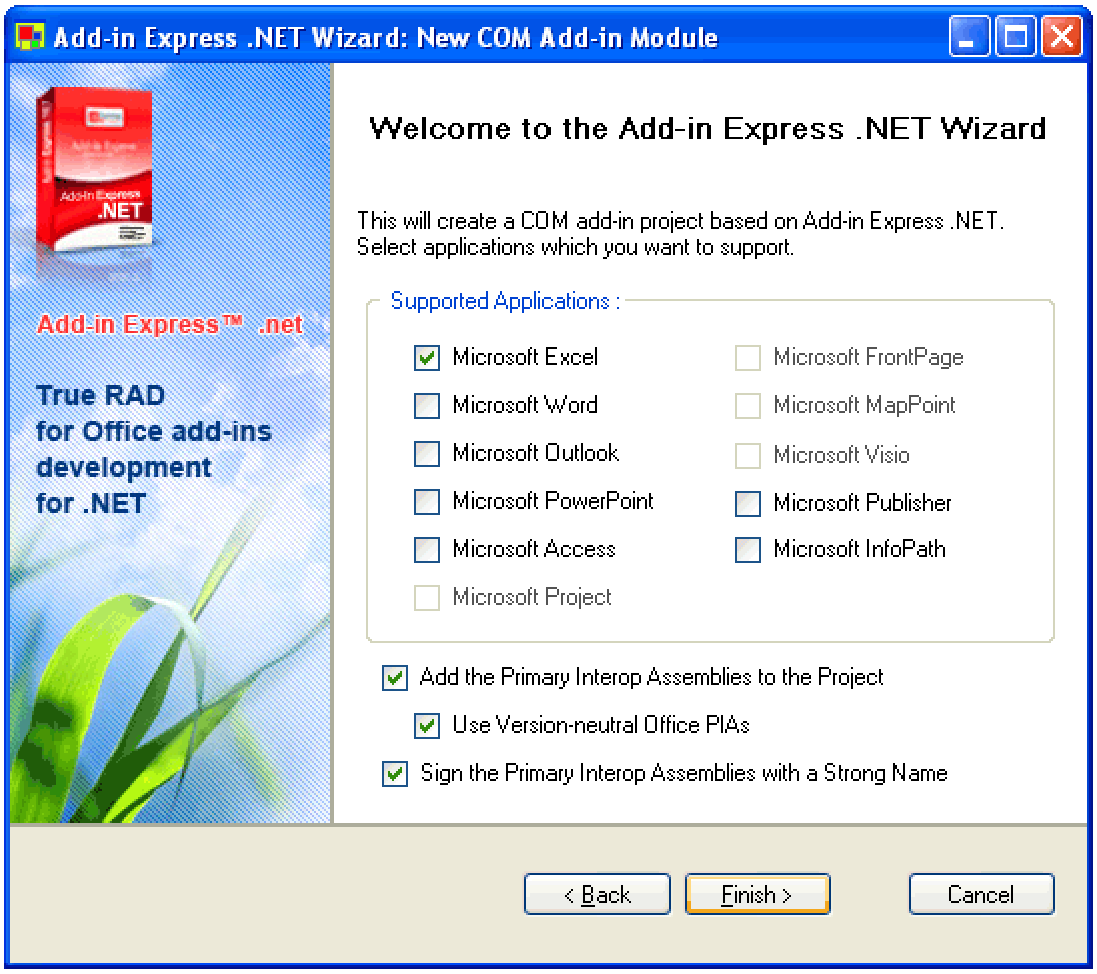
- ( ) Visual C# Project
- (●) Visual Basic Project
- ( ) Visual C++ Project
- ( ) Visual J# Project
- ( ) Visual Chrome Project

Isolation:

- (●) ADX Loader
- ( ) Non Isolated

[✓] Create the Setup Project

This VB.NET sample shows an Add-in Express COM Add-in project implementing a COM add-in for Excel with the Add-in Express Loader as a shim. To understand shims and the Add-in Express Loader, see Deploying Add-in Express Projects.


Add-in Express .NET Wizard: New COM Add-in Module

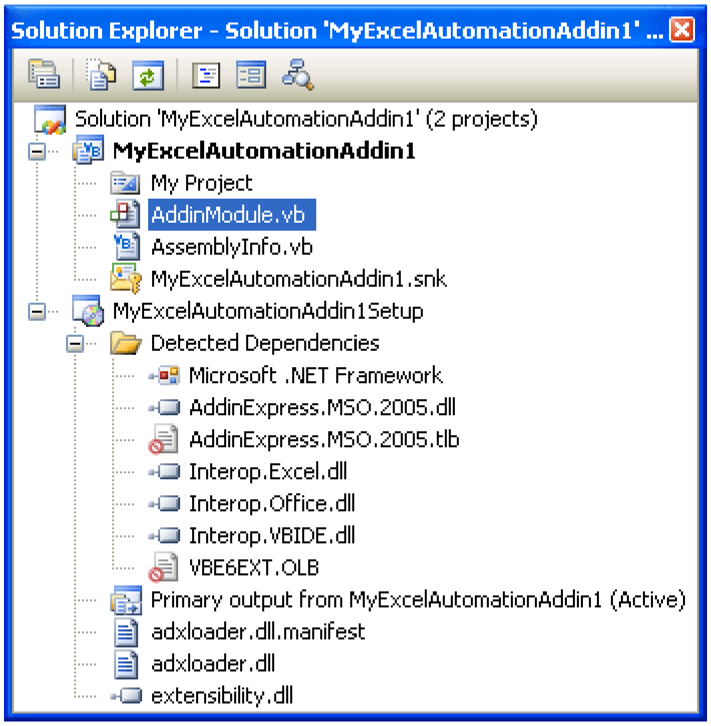
Welcome to the Add-in Express .NET Wizard

This will create a COM add-in project based on Add-in Express .NET.
Select applications which you want to support.

Supported Applications :

- [✓] Microsoft Excel
- [ ] Microsoft Word
- [ ] Microsoft Outlook
- [ ] Microsoft PowerPoint
- [ ] Microsoft Access
- [ ] Microsoft Project
- [ ] Microsoft FrontPage
- [ ] Microsoft MapPoint
- [ ] Microsoft Visio
- [ ] Microsoft Publisher
- [ ] Microsoft InfoPath

- [✓] Add the Primary Interop Assemblies to the Project
- [✓] Use Version-neutral Office PIAs
- [✓] Sign the Primary Interop Assemblies with a Strong Name


Solution Explorer - Solution 'MyExcelAutomationAddin1' ...

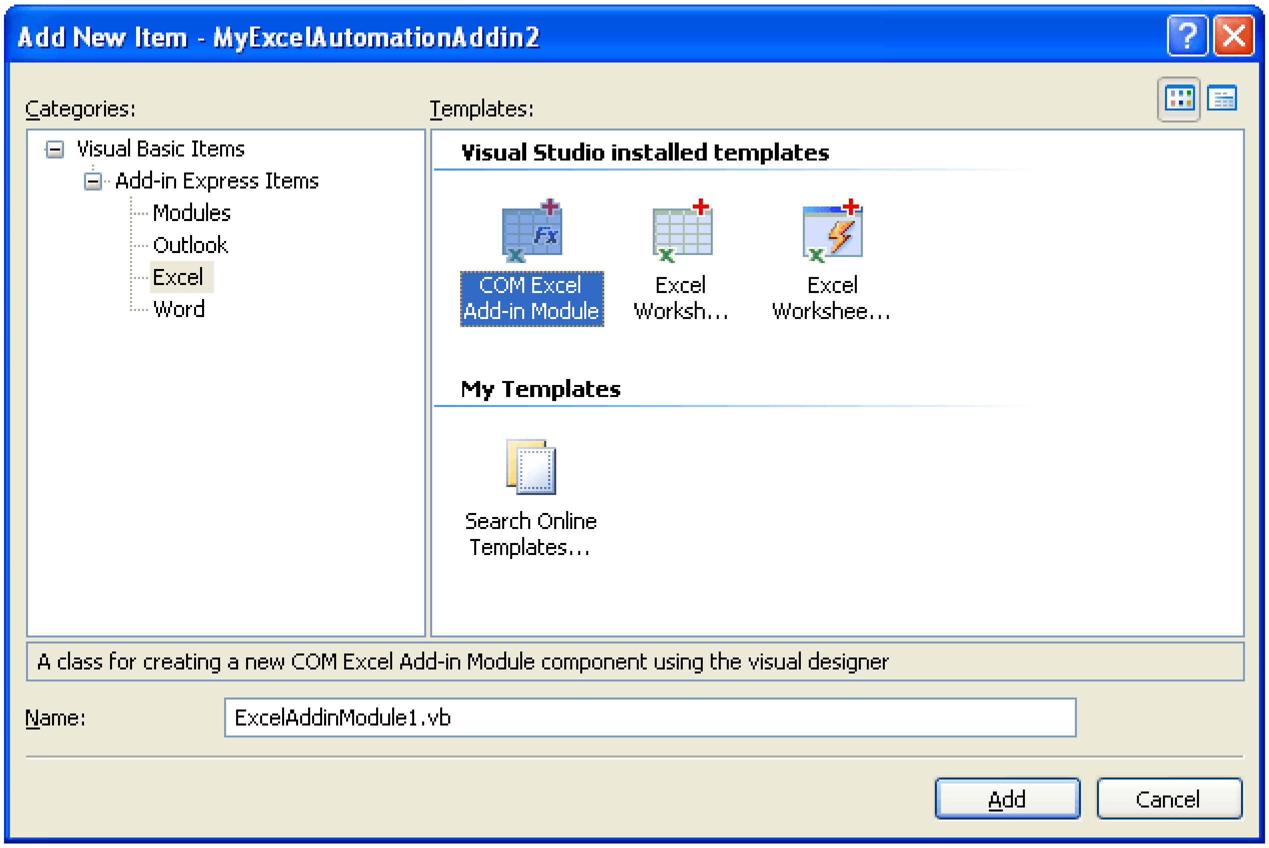
```
Solution 'MyExcelAutomationAddin1' (2 projects)
  MyExcelAutomationAddin1
    My Project
    AddinModule.vb
    AssemblyInfo.vb
    MyExcelAutomationAddin1.snk
  MyExcelAutomationAddin1Setup
    Detected Dependencies
      Microsoft .NET Framework
      AddinExpress.MSO.2005.dll
      AddinExpress.MSO.2005.tlb
      Interop.Excel.dll
      Interop.Office.dll
      Interop.VBIDE.dll
      VBE6EXT.OLB
    Primary output from MyExcelAutomationAddin1 (Active)
    adxloader.dll.manifest
    adxloader.dll
    extensibility.dll
```

The Add-in Express Project Wizard creates and opens the COM Add-in solution in IDE. The solution includes the COM Add-in project and the setup project.

The COM Add-in project contains the AddinModule.vb (or AddinModule1.cs) file discussed in Your First Microsoft Office COM Add-in.

Step #2 – Adding a New COM Excel Add-in Module

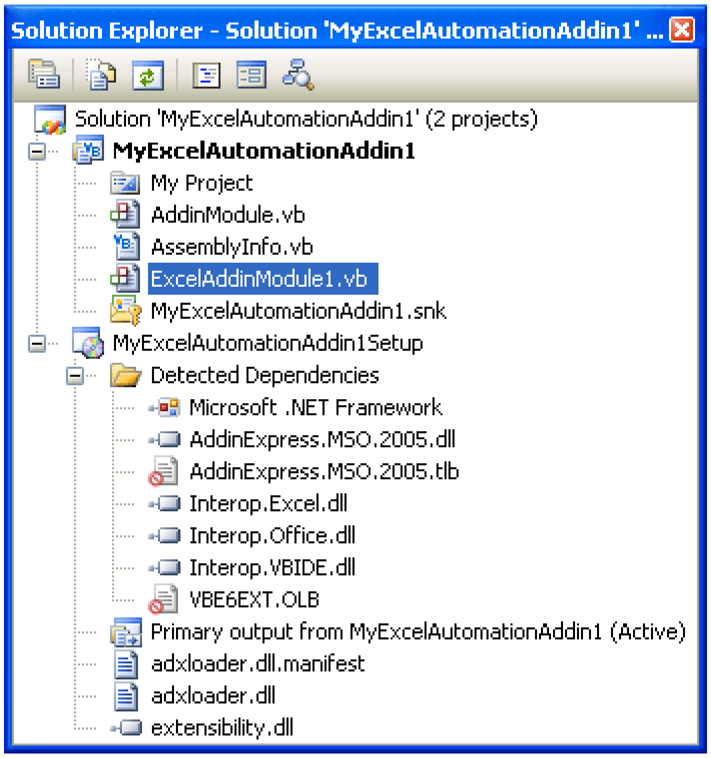
In order to add Excel user-defined functions to the COM Add-in, you add the COM Excel Add-in Module to the Add-in Express COM Add-in project using the Add New Item dialog.



A class for creating a new COM Excel Add-in Module component using the visual designer

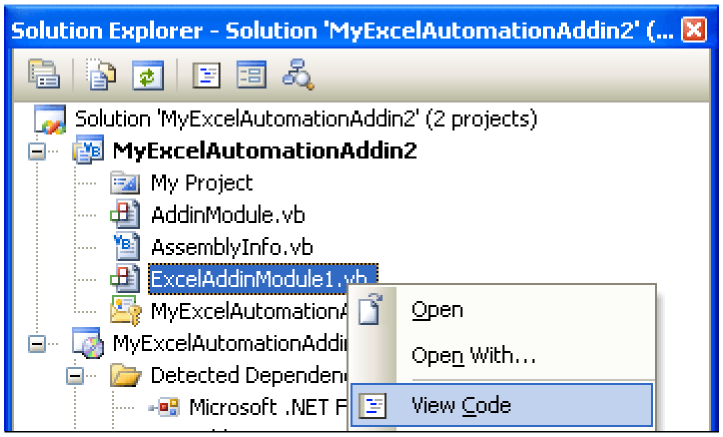
Name: ExcelAddinModule1.vb

This adds the ExcelAddinModule1.vb (or ExcelAddinModule1.cs) file to your COM Add-in project.


Solution Explorer - Solution 'MyExcelAutomationAddin1' ...
Solution 'MyExcelAutomationAddin1' (2 projects)
MyExcelAutomationAddin1
My Project
AddinModule.vb
AssemblyInfo.vb
ExcelAddinModule1.vb
MyExcelAutomationAddin1.snk
MyExcelAutomationAddin1Setup
Detected Dependencies
Microsoft .NET Framework
AddinExpress.MSO.2005.dll
AddinExpress.MSO.2005.tlb
Interop.Excel.dll
Interop.Office.dll
Interop.VBIDE.dll
VBE6EXT.OLB
Primary output from MyExcelAutomationAddin1 (Active)
adxloader.dll.manifest
adxloader.dll
extensibility.dll

Step #3- Writing a User-Defined Function

In the Solution Explorer window, right-click the ExcelAddinModule.vb (or ExcelAddinModule.cs) file and choose the View Code item in the context menu.



The module contains the following code:

```
Imports System.Runtime.InteropServices

'Add-in Express Excel Add-in Module
<GuidAttribute("287D044F-D233-47E6-BB48-35999635BAD3"), _
```

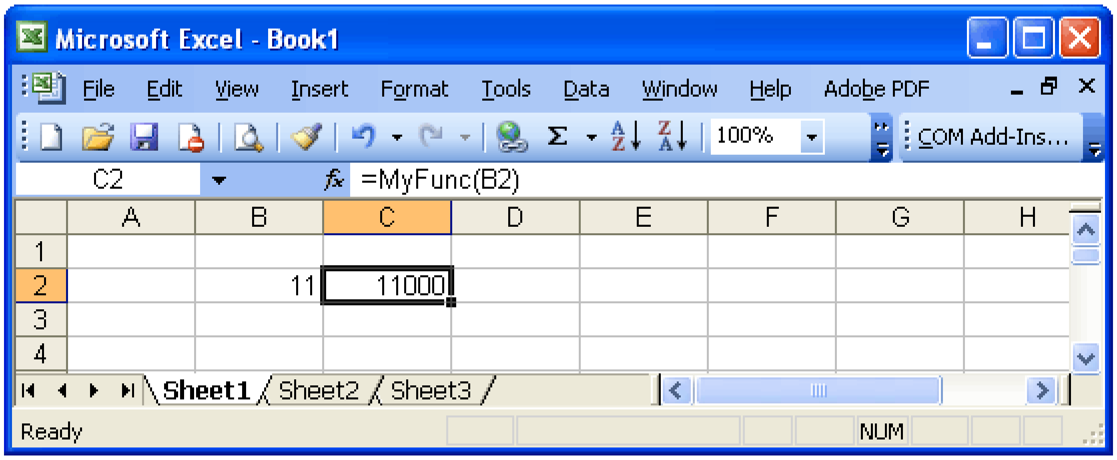
```
ProgIdAttribute("MyExcelAutomationAddin2.ExcelAddinModule1"), _
   ClassInterface(ClassInterfaceType.AutoDual)> _
Public Class ExcelAddinModule1
   Inherits AddinExpress.MSO.ADXExcelAddinModule

#Region " Add-in Express automatic code "

   <ComRegisterFunctionAttribute()> _
   Public Shared Sub AddinRegister(ByVal t As Type)
      AddinExpress.MSO.ADXExcelAddinModule.ADXExcelAddinRegister(t)
   End Sub

   <ComUnregisterFunctionAttribute()> _
   Public Shared Sub AddinUnregister(ByVal t As Type)
      AddinExpress.MSO.ADXExcelAddinModule.ADXExcelAddinUnregister(t)
   End Sub

#End Region

   Public Sub New()
      MyBase.New()
   End Sub
End Class
```

Just add a new function to the module. Say, the following one:

```
   Public Function MyFunc(ByVal Range As Object) As Object
      MyFunc = CType(Range, Excel.Range).Value * 1000
   End Function
```

Step #4 - Running the Excel Automation Add-in

Choose the Register Add-in Express Project item in the Build menu, restart Excel, and check if your add-in works.

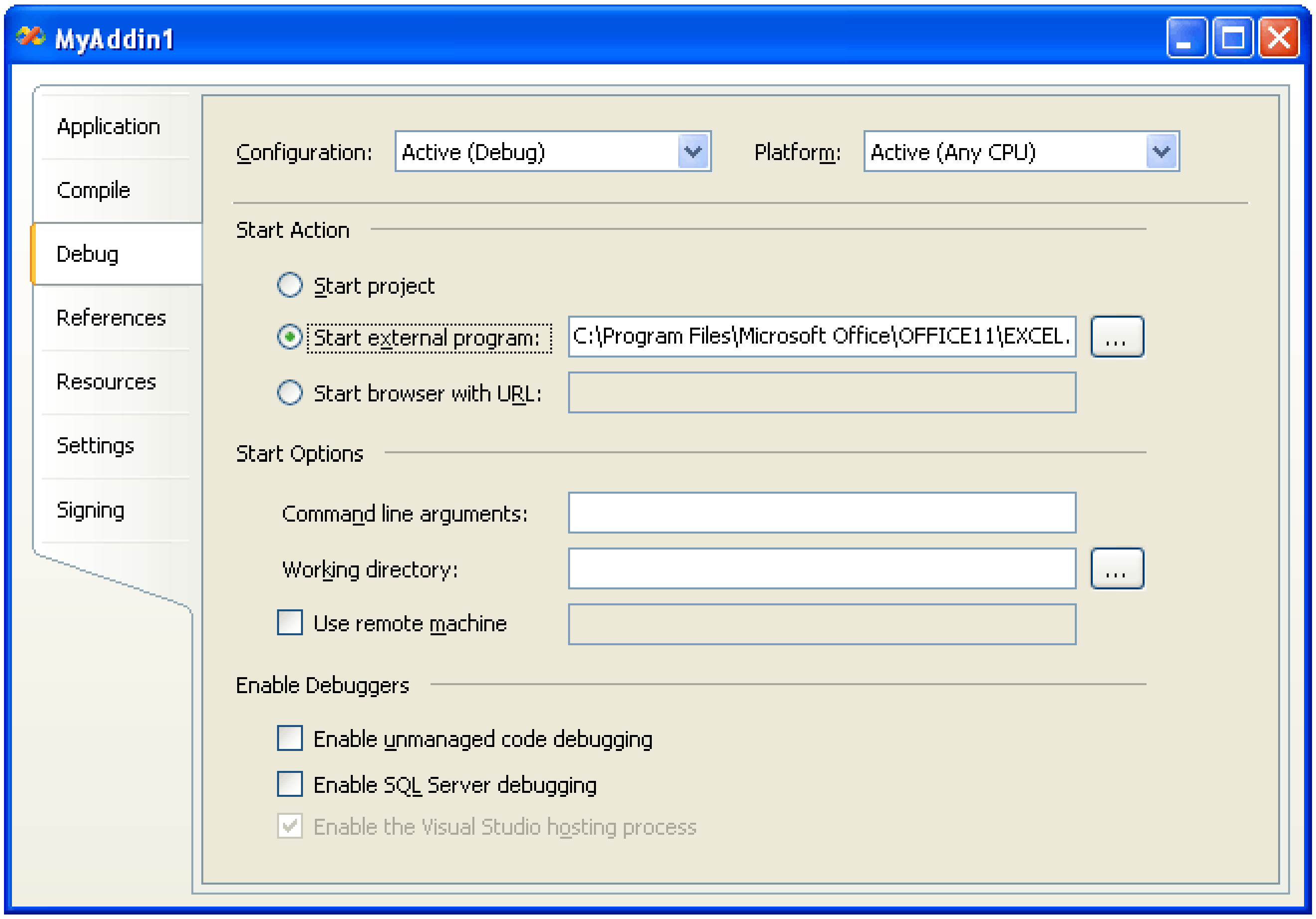


C2 =MyFunc(B2)

	A	B	C	D	E	F	G	H
1								
2		11	11000					
3								
4								

Step #5 - Debugging the Excel Automation Add-in

To debug your add-in, just indicate the add-in host application as the Start Program in the Project Options window.



Configuration: Active (Debug)
Platform: Active (Any CPU)

Start Action

( ) Start project
(●) Start external program: C:\Program Files\Microsoft Office\OFFICE11\EXCEL.
( ) Start browser with URL:

Start Options

Command line arguments:
Working directory:
☐ Use remote machine

Enable Debuggers

☐ Enable unmanaged code debugging
☐ Enable SQL Server debugging
☑ Enable the Visual Studio hosting process

However, there is a problem here. When debugging an add-in or a smart tag on Visual Studio 2003 and Office XP you cannot see your add-in or smart tag on the COM add-ins or AutoCorrect dialog box. This is a “feature” of Office XP "added" by MS people.

Step #6 - Deploying the Excel Automation Add-in

Just built the setup project, copy all setup files to the target PC and run the setup.exe file to install the add-in.

Deploying Add-in Express Projects

All Office applications are unmanaged. And all Add-in Express .NET based add-ins are managed class libraries. Therefore, there must be some software located between Office applications and your add-ins. Otherwise, Office applications will not know of your .NET add-ins and other Office extensions. That software is called a shim. Shims are unmanaged DLLs that isolate your add-ins in a separate application domain.

Generally, (but see the table below) to deploy managed COM add-ins you will need to incorporate a shim into your add-in project. When you install your add-in, the registry settings for the add-in will point to the shim. And the shim will be the first DLL examined by the host application when it runs the add-in (the same applies to smart tags).

You have two isolation options when creating Add-in Express .NET projects. You can see them in the Add-in Express Project Wizard window. Below we show you the shim features table.

	MSCOREE.DLL	Add-in Express Loader
Is the add-in isolated in a separate application domain?	No	Yes
Is the source code available?	No	No
Does the shim require adding to the setup project?	No	Yes (added automatically if created by the Add-in Express Project Wizard)
Add-in Express DLL that provides you with ready-to-use custom actions	AddinExpress.Install.dll AddinExpress.Install.2005.dll	adxloader.dll
Can the shim be signed?	No	Yes
Does the shim allow updating add-in DLLs when add-in is running?	No	Yes (see Deploying – Shadow Copy)
Is the shim configurable?	No	Yes (by changing the manifest file)
Can regsvr32 be run against the shim?	No	Yes
Does the shim allow many add-ins in one assembly?	Yes	Yes

Setup Projects for Add-in Express Solutions

Add-in Express provides the Add-in Express Loader (adxloader.dll), which is a compiled shim not bound to any certain Add-in Express project. Instead, the loader uses the adxloader.dll.manifest file containing a list of .NET assemblies that must be registered. The loader's files (adxloader.dll and adxloader.dll.manifest) must always be located in the Loader subdirectory of the main Add-in Express Project directory. When an

Add-in Express based project is being built, the loader files are copied to the project's output directory. You can sign the loader with a digital signature and create trusted COM extensions for Office. For Add-in Express loader-based setup projects, the loader provides ready-to-use Instal, Ununstal, and Roolback custom actions.

If, in the project wizard, you choose the Loader and automatic setup project generation, the Add-in Express .NET wizard does the following:

- Creates Add-in Express and setup projects.
- In the Add-in Express project's folder, creates the Loader folder and places adxloader.dll in it.
- Generates adxloader.dll.manifest and places it into the Loader folder.
- Adds the Add-in Express project's Primary Output to the setup project.
- Adds the adxloader.dll and adxloader.dll.manifest files to the setup project.
- Generates custom actions that reference adxLoader.dll.

Add-in Express Loader requires the manifest file adxloader.dll.manifest to be located in the same folder. Below, you see the contents of a sample manifest file.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<assemblyIdentity name="MyAddin1, PublicKeyToken=4590b35563ddbf3d" />
<loaderSettings generateLogFile="true" shadowCopyEnabled="true"
   priviledges ="administrator" />
</configuration>
```

To add another Add-in Express based assembly to the manifest file, you add another <assemblyIdentity> node.

The manifest file allows generating the log file containing useful information about errors on the add-n loading stage. The log file is located here: My Docyments\Add-in Express\adxloader.log The manifest file also allows you to disable the Shadow Copy feature of Add-in Express Loader, which is enabled by default (see Deploying – Shadow Copy). The **privileges** attribute accepts the "user" string value indicating that the Add-in Express based setup projects can be run with non-administrator privileges. Please, note, all other string values will require administrator privileges to install your project. Also, you should be aware that the value of this attribute is controlled by the RegisterForAllUsers property value of the add-in and RTD module.

Also, you can run regsvr32 against the adxloader.dll. Provided that the correct manifest file is located in the same folder, this will register all Add-in Express projects listed in the manifest file.

Creating Setup Projects Manually

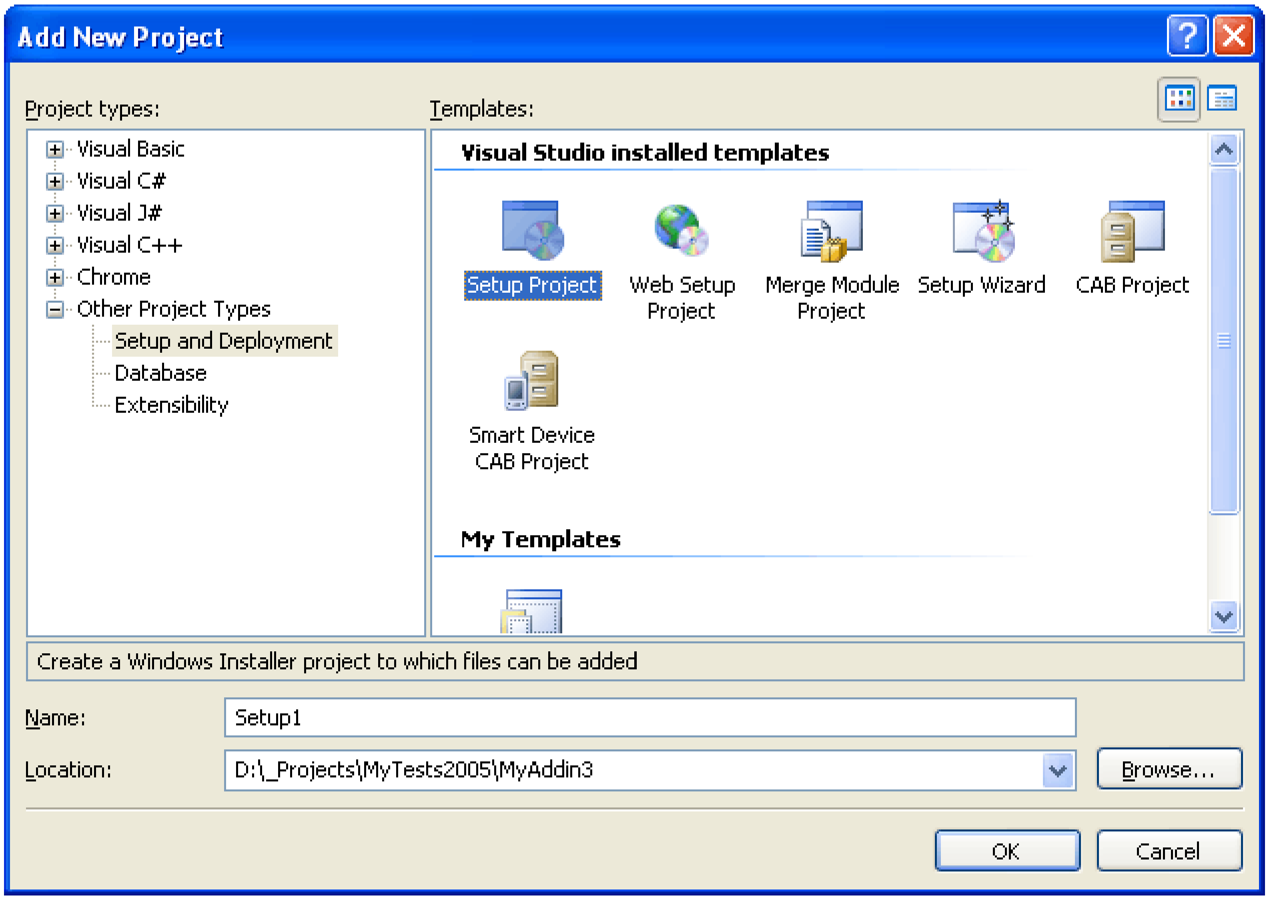
We don't recommend that you create setup project manually because you always can create them with the Add-in Express Project Wizard when starting a new project. Nevertheless, if you need to create a setup project manually, use the following step-by-step instructions.

Add-in Express Loader

To create the setup project manually, please follow the steps below.

Add a New Setup Project

Right-click the solution item and choose Add | New Project.

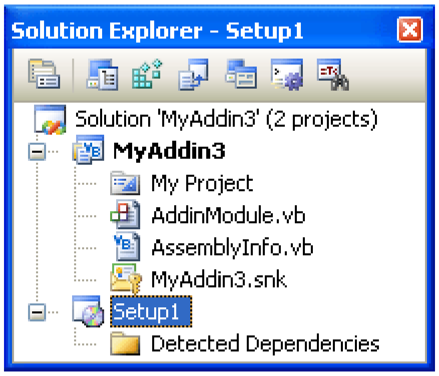


Create a Windows Installer project to which files can be added

Name: Setup1

Location: D:_Projects\MyTests2005\MyAddin3

In the Add New Project dialog, select the Setup Project item and click OK. This will add the setup project to your solution.

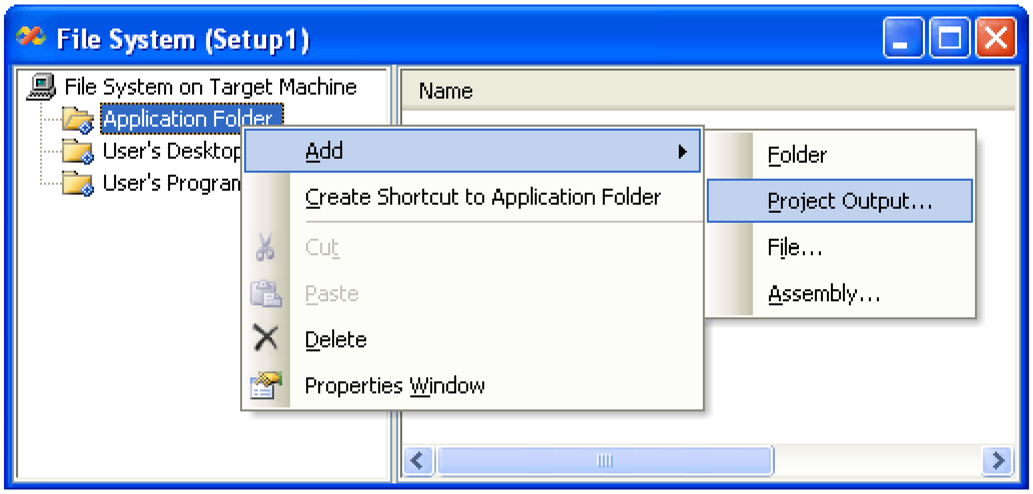


File System Editor

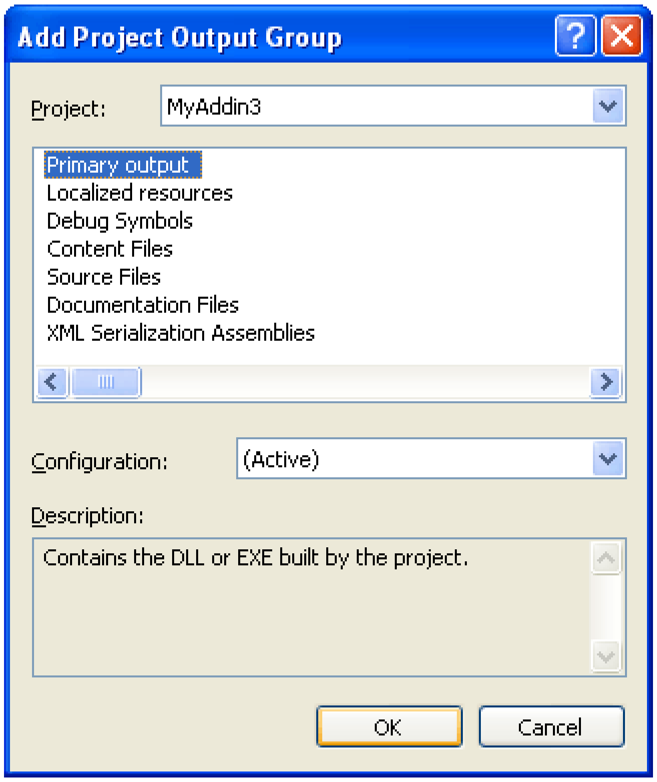
Right-click the setup project item (Setup1 in the screenshot) and choose View | File System.

Primary Output

Right-click the Application Folder item and choose Add | Project Output

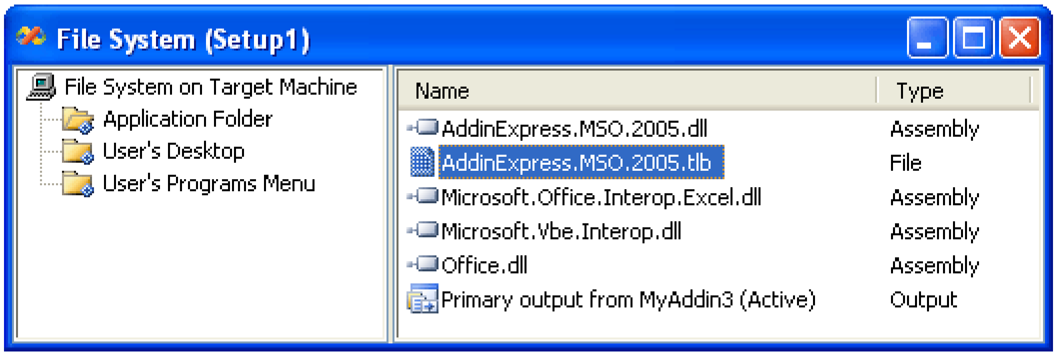


In the Add Project Output Group dialog, select the Primary Output Item of your Add-in/RTD Server/Smart Tag project and click OK.

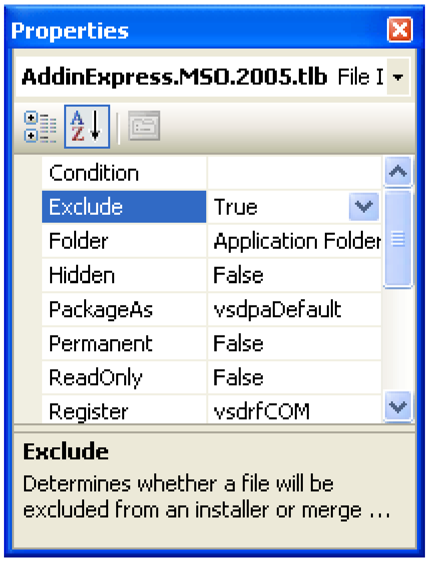


Project: MyAddin3
Configuration: (Active)
Description: Contains the DLL or EXE built by the project.

This adds the following entries to the Application Folder of your setup project.



Name	Type
AddinExpress.MSO.2005.dll	Assembly
AddinExpress.MSO.2005.tlb	File
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin3 (Active)	Output



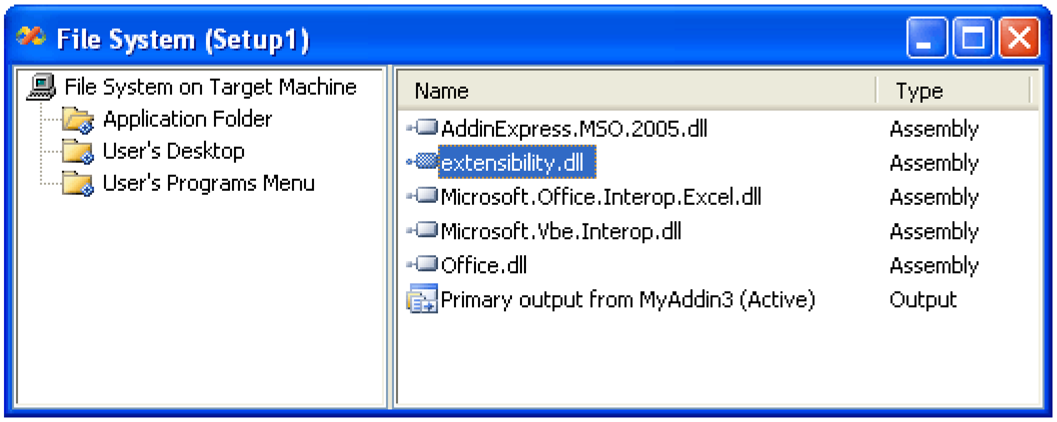
Condition	
Exclude	True
Folder	Application Folder
Hidden	False
PackageAs	vsdpaDefault
Permanent	False
ReadOnly	False
Register	vsdrfCOM

Exclude
Determines whether a file will be excluded from an installer or merge ...

Select AddinExpress.MSO.2005.tlb (or AddinExpress.MSO.2003.tlb in Visual Studio 2003) and, in the Properties window, set the Exclude property to True. If you use version-neutral PIAs, please exclude the VB6EXT.OLB file in the same way.

Extensibility.dll

Add the Extensibility.dll assembly to the Application Folder if it doesn't exist in the Detected Dependencies section of the setup project



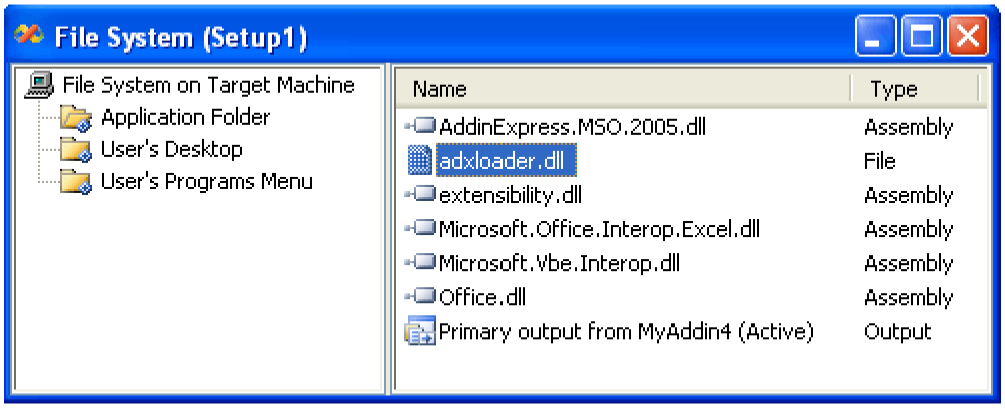
Name	Type
AddinExpress.MSO.2005.dll	Assembly
extensibility.dll	Assembly
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin3 (Active)	Output

Project-depended Resources

Now you add all resources (e.g. assemblies, dlls or any resources) required for your project.

Add-in Express Loader

Copy the adxloader.dll file from the 'Redistributables' subfolder of the Add-in Express installation directory to the 'Loader' subfolder of the add-in project directory. Then add the adxloader.dll file to the 'Application Folder' of the setup project.



Name	Type
AddinExpress.MSO.2005.dll	Assembly
adxloader.dll	File
extensibility.dll	Assembly
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin4 (Active)	Output

Add-in Express Loader Manifest

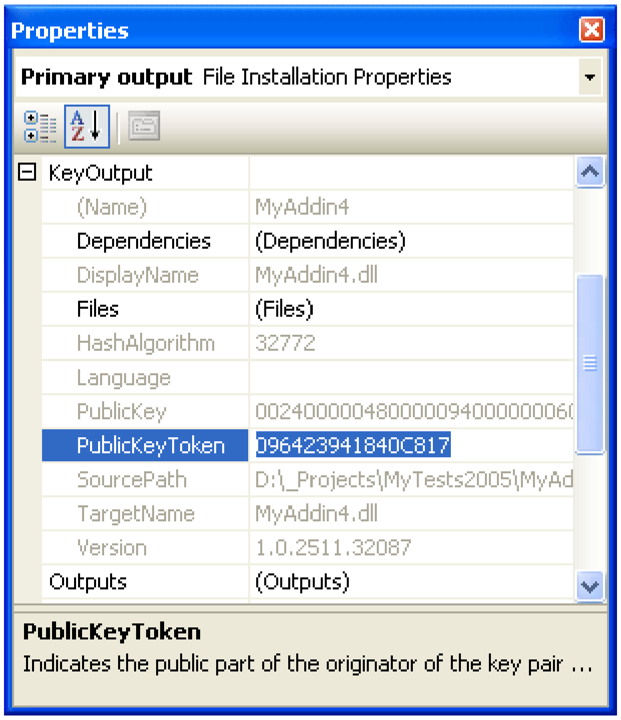
Create a new text file, say in Notepad, and add the following text into it:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<assemblyIdentity name="<the name of the add-in assembly>,
 PublicKeyToken=<public key token of the add-in assembly>" />
</configuration>
```

A sample Add-in Express Loader manifest file can look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<assemblyIdentity name="MyAddin4, PublicKeyToken=096423941840C817" />
</configuration>
```

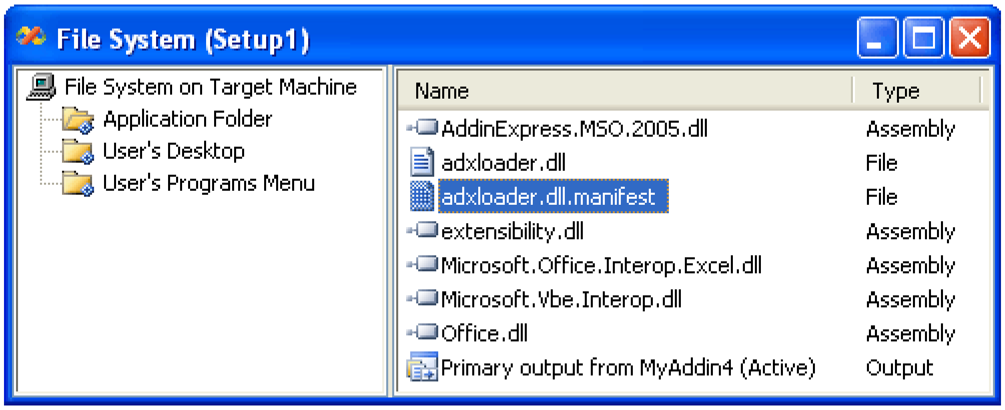
The Name attribute of the assemblyIdentity node includes the name of you add-in assembly (without '.dll') and PublicKeyToken of your add-in assembly. The latter can be found in the setup project (that must be already built) - click on your add-in primary output in the setup projects and, in the Properties window, expand the KeyOutput property and see the PublicKeyToken property value.


Properties
Primary output File Installation Properties

KeyOutput	
(Name)	MyAddin4
Dependencies	(Dependencies)
DisplayName	MyAddin4.dll
Files	(Files)
HashAlgorithm	32772
Language	
PublicKey	002400000480000094000000060
PublicKeyToken	096423941840C817
SourcePath	D:_Projects\MyTests2005\MyAd
TargetName	MyAddin4.dll
Version	1.0.2511.32087
Outputs	(Outputs)

PublicKeyToken
Indicates the public part of the originator of the key pair …

Save the text file under ‘adxloader.dll.manifest’ name to the ‘Loader’ subfolder of the add-in project directory, and then add it to the ‘Application Folder’ of the setup project.

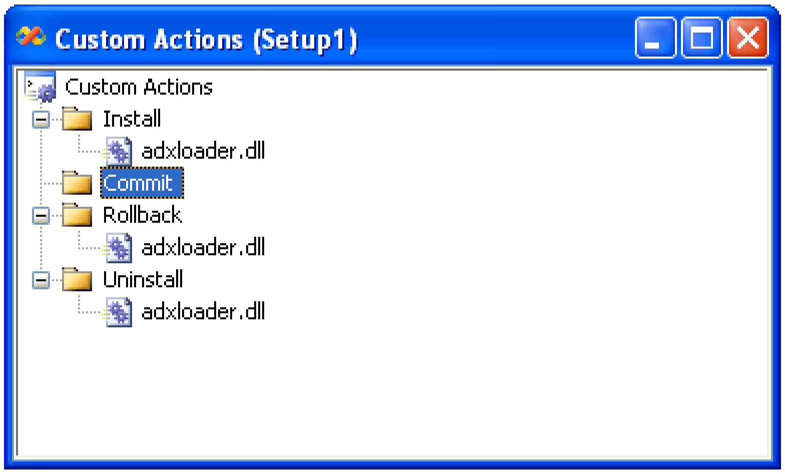

File System (Setup1)

File System on Target Machine
- Application Folder
- User's Desktop
- User's Programs Menu

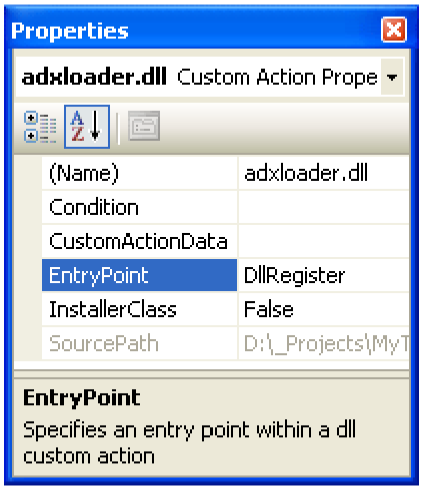
Name	Type
AddinExpress.MSO.2005.dll	Assembly
adxloader.dll	File
adxloader.dll.manifest	File
extensibility.dll	Assembly
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin4 (Active)	Output

Custom Actions

Open the Custom Actions editor and add a new action to the Install, Rollback, Uninstall sections. Use the adxloader.dll file as an item for the custom actions.



EntryPoint



(Name)	adxloader.dll
Condition	
CustomActionData	
EntryPoint	DllRegister
InstallerClass	False
SourcePath	D:_Projects\MyT

EntryPoint
Specifies an entry point within a dll custom action

Add the following parameter to the EntryPoint property of the following custom actions:

- **Install**

```
DllRegister
```

- **Rollback**

```
DllUnregister
```

- **Uninstall**

```
DllUnregister
```

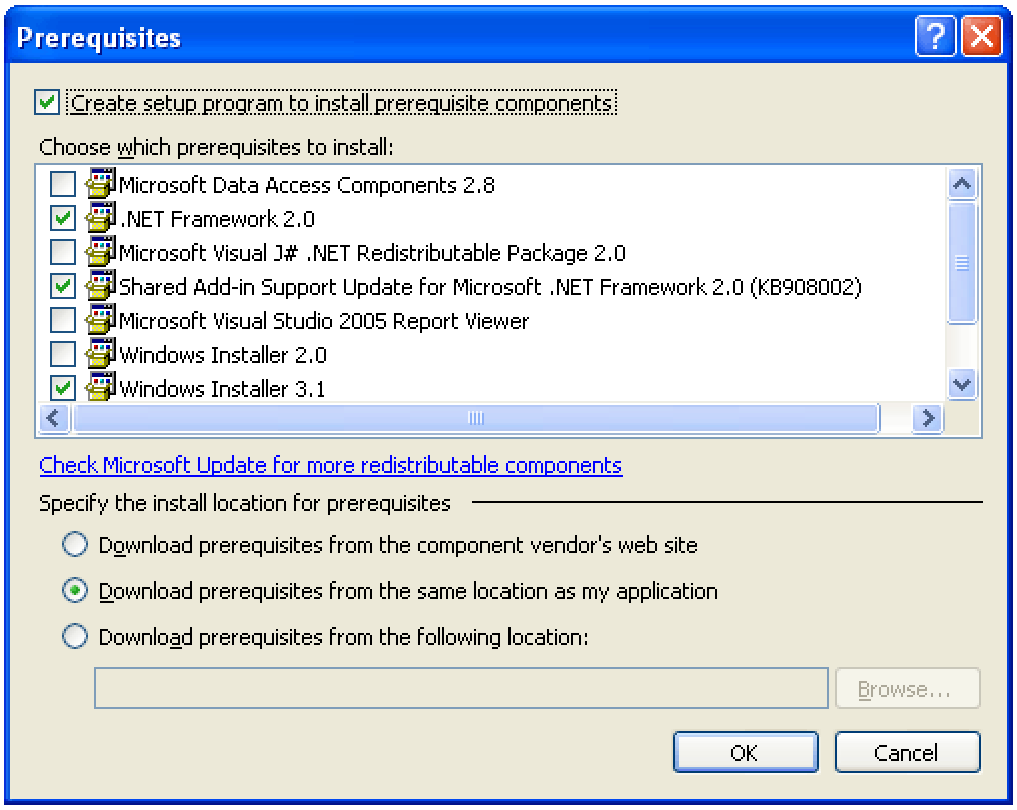
Dependencies

Right click on the Detected Dependencies secton of the setup project and choose the Refresh Dependencies option. Also, exclude all dependencies that are not required for your setup.

VS 2005 only

Right click on the setup project and open the Properties dialog.

Click on the Prerequisites button and, in the Prerequisites dialog, check all prerequisites you need.



Prerequisites

[✓] Create setup program to install prerequisite components

Choose which prerequisites to install:

- [ ] Microsoft Data Access Components 2.8
- [✓] .NET Framework 2.0
- [ ] Microsoft Visual J# .NET Redistributable Package 2.0
- [✓] Shared Add-in Support Update for Microsoft .NET Framework 2.0 (KB908002)
- [ ] Microsoft Visual Studio 2005 Report Viewer
- [ ] Windows Installer 2.0
- [✓] Windows Installer 3.1

Check Microsoft Update for more redistributable components

Specify the install location for prerequisites

- ○ Download prerequisites from the component vendor's web site
- ● Download prerequisites from the same location as my application
- ○ Download prerequisites from the following location:

You can choose the 'Download prerequisites from the same location as my application' option to distribute all prerequisites with the add-in installation package.

Deploy

Rebuild the setup project. Copy all setup files to the target PC and run the setup.exe file to install the add-in.

MSCOREE.DLL

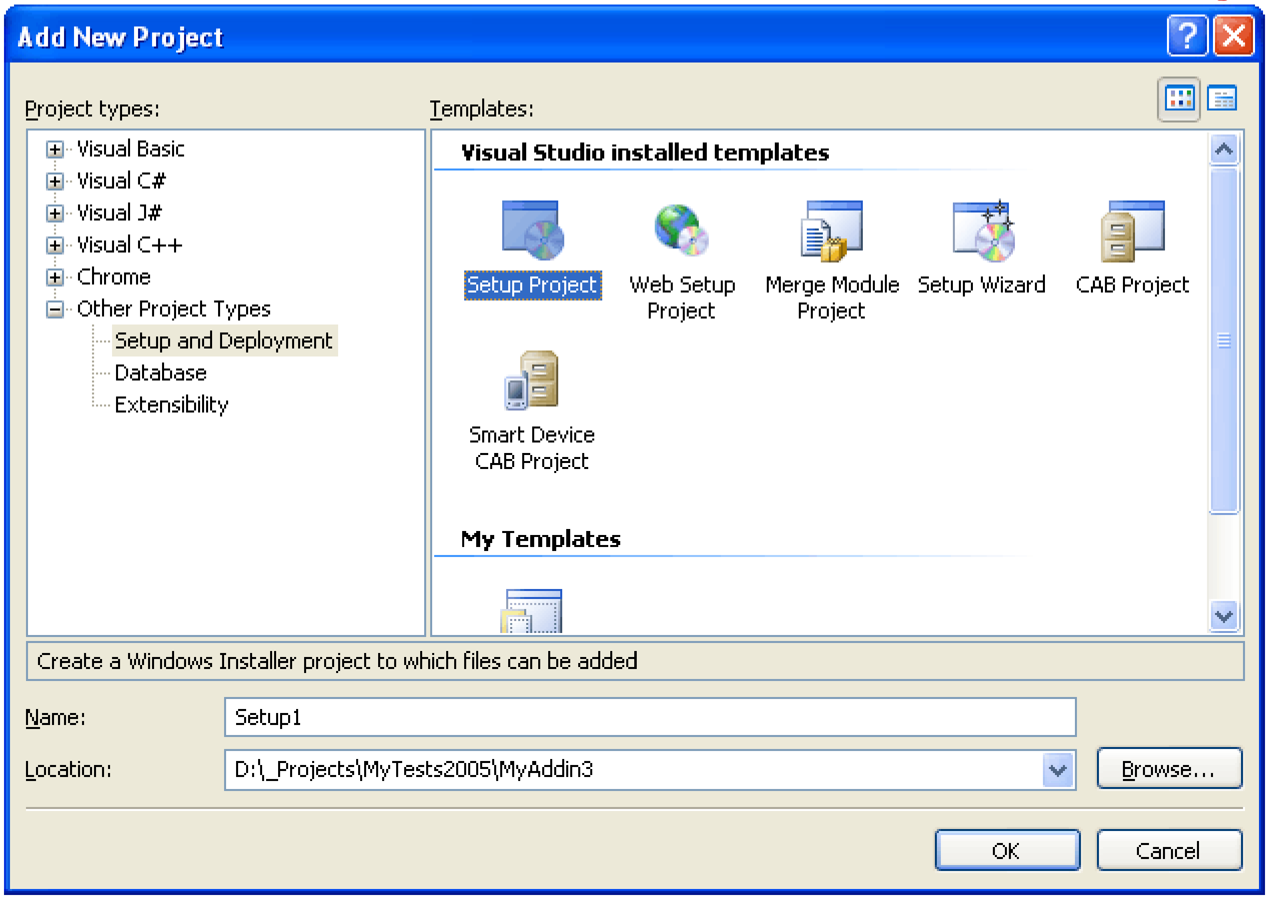
Typically, you use this shim when your add-in doesn't have any special requirements to security. The main advantage of using this shim is its simplicity. However, you cannot sign it, and this is its main drawback. If the host application security is set to Medium, High or Very High (Tools | Macro | Security), unsigned add-ins will not start. This shim is included in the .NET Framework. That is, you don't need to include it into your setup project.

If you choose the Create the Setup Project option of the Add-in Express Project Wizard, the wizard will create the setup project automatically.

To create setup project manually, please follow the steps below.

Add a New Setup Project

Right-click the solution item and choose Add | New Project.



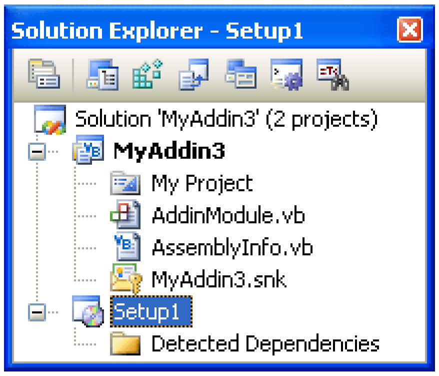
Create a Windows Installer project to which files can be added

Name: Setup1

Location: D:_Projects\MyTests2005\MyAddin3

In the Add New Project dialog, select the Setup Project item and click OK. This will add the setup project to your solution.

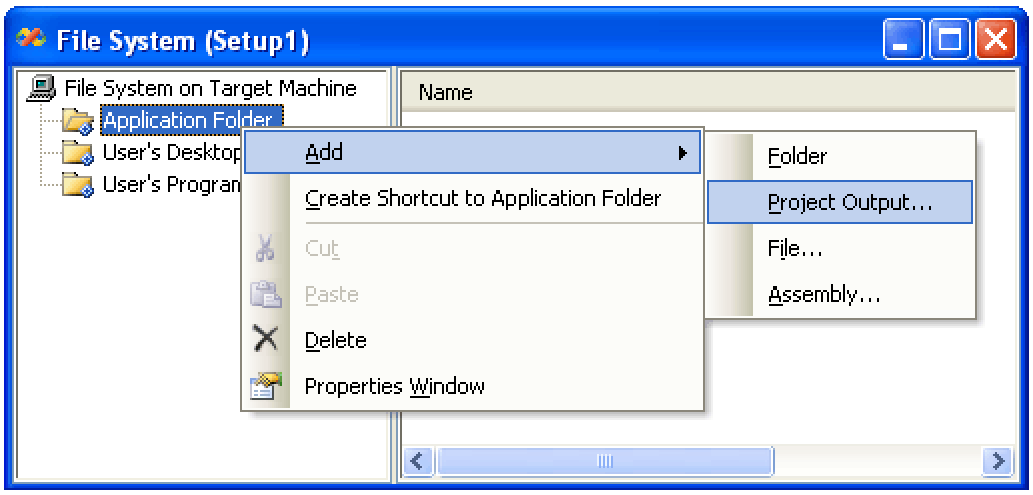
File System Editor

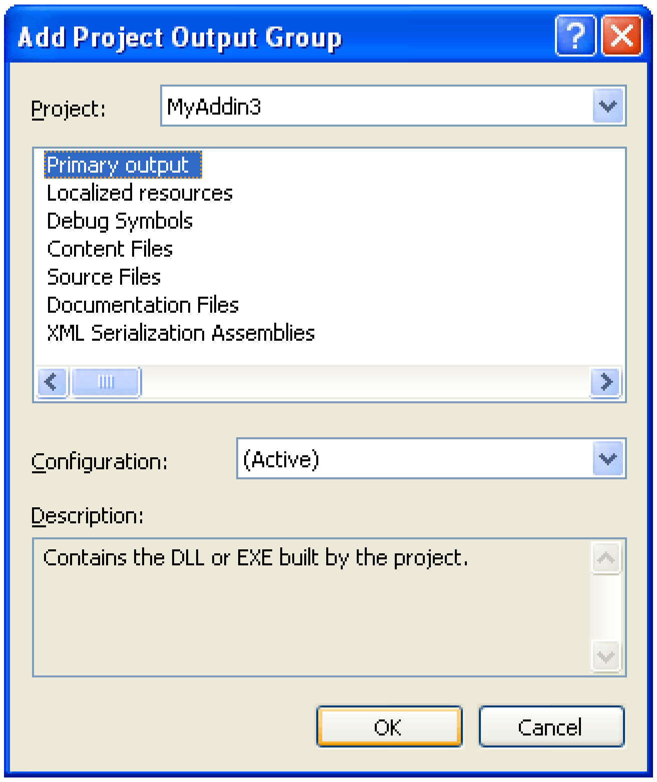


Right-click the setup project item (Setup1 in the screenshot) and choose View | File System.

Primary Output

Right-click the Application Folder item and choose Add | Project Output

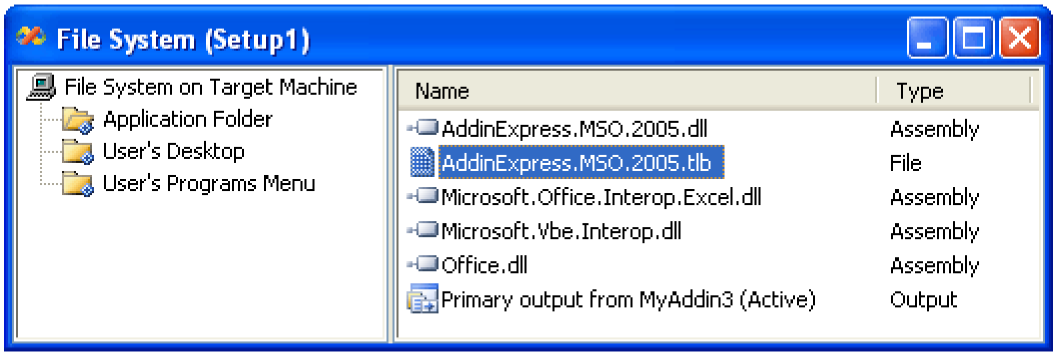




Contains the DLL or EXE built by the project.

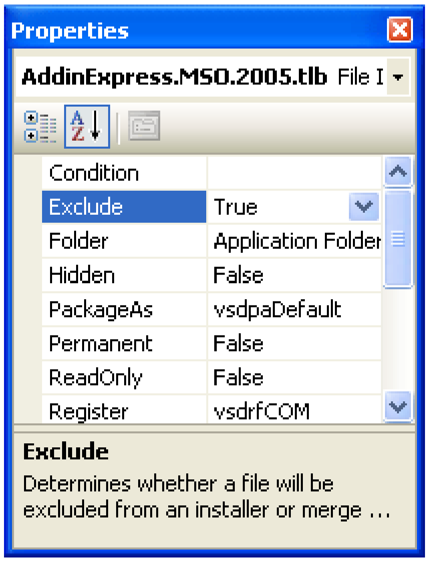
In the Add Project Output Group dialog, select the Primary Output Item of your Add-in/RTD Server/Smart Tag project and click OK.

This adds the following entries to the Application Folder of your setup project.



Name	Type
AddinExpress.MSO.2005.dll	Assembly
AddinExpress.MSO.2005.tlb	File
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin3 (Active)	Output

Select AddinExpress.MSO.2005.tlb (or AddinExpress.MSO.2003.tlb in Visual Studio 2003) and, in the Properties window, set the Exclude property to True. If you use version-neutral PIAs, please exclude the VB6EXT.OLB file in the same way.

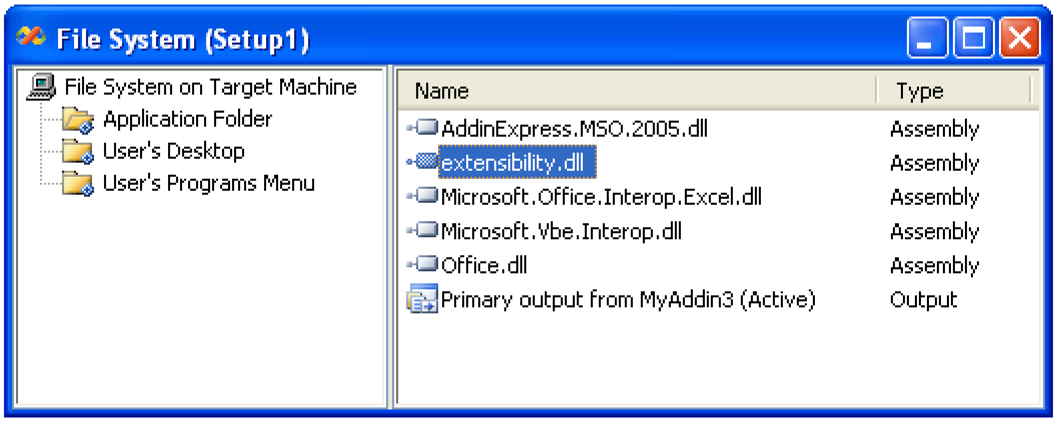


Condition	
Exclude	True
Folder	Application Folder
Hidden	False
PackageAs	vsdpaDefault
Permanent	False
ReadOnly	False
Register	vsdrfCOM

Exclude
Determines whether a file will be excluded from an installer or merge ...

Extensibility.dll

Add the Extensibility.dll assembly to the Application Folder if it doesn't exist in the Detected Dependencies section of the setup project



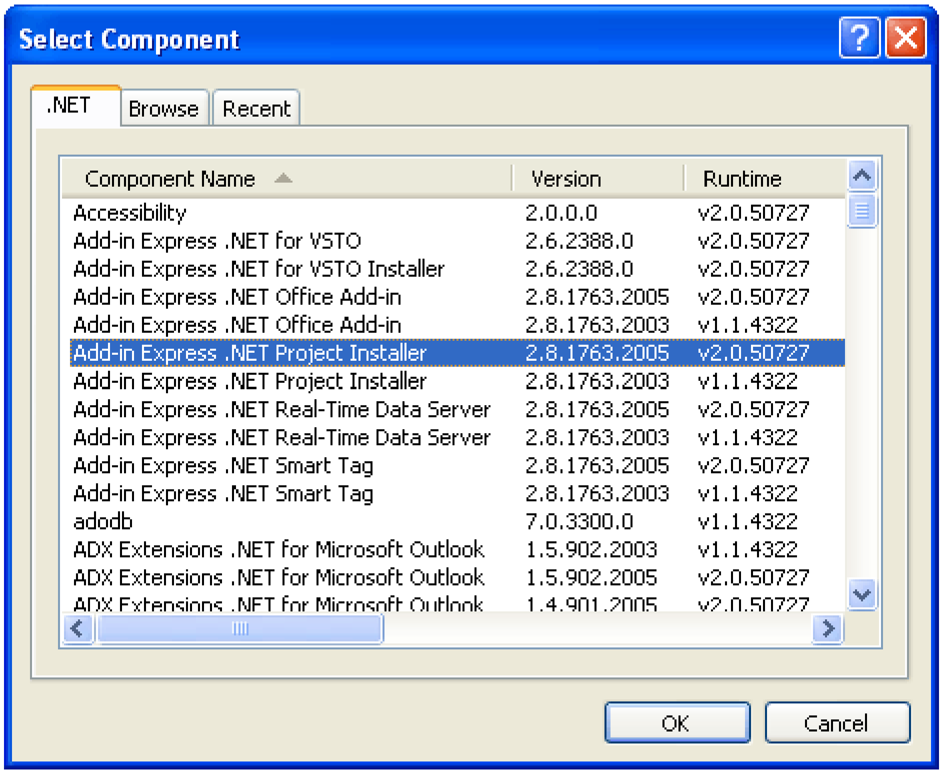
Name	Type
AddinExpress.MSO.2005.dll	Assembly
extensibility.dll	Assembly
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin3 (Active)	Output

Project-depended Resources

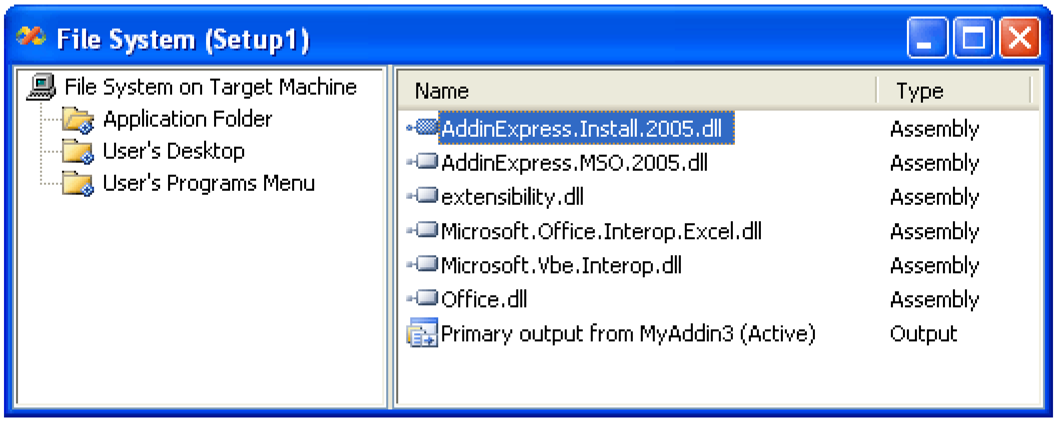
Now you add all resources (e.g. assemblies, dlls or any resources) required for your project.

AddinExpress.Install.2005.dll / AddinExpress.Install.dll

Add the AddinExpress.Install2005.dll / AddinExpress.Install.dll (for VS 2003) assembly to the Application Folder.

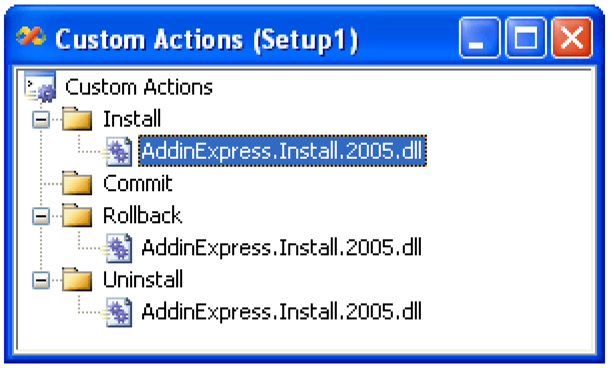


Component Name	Version	Runtime
Accessibility	2.0.0.0	v2.0.50727
Add-in Express .NET for VSTO	2.6.2388.0	v2.0.50727
Add-in Express .NET for VSTO Installer	2.6.2388.0	v2.0.50727
Add-in Express .NET Office Add-in	2.8.1763.2005	v2.0.50727
Add-in Express .NET Office Add-in	2.8.1763.2003	v1.1.4322
Add-in Express .NET Project Installer	2.8.1763.2005	v2.0.50727
Add-in Express .NET Project Installer	2.8.1763.2003	v1.1.4322
Add-in Express .NET Real-Time Data Server	2.8.1763.2005	v2.0.50727
Add-in Express .NET Real-Time Data Server	2.8.1763.2003	v1.1.4322
Add-in Express .NET Smart Tag	2.8.1763.2005	v2.0.50727
Add-in Express .NET Smart Tag	2.8.1763.2003	v1.1.4322
adodb	7.0.3300.0	v1.1.4322
ADX Extensions .NET for Microsoft Outlook	1.5.902.2003	v1.1.4322
ADX Extensions .NET for Microsoft Outlook	1.5.902.2005	v2.0.50727
ADX Extensions .NET for Microsoft Outlook	1.4.901.2005	v2.0.50727



Name	Type
AddinExpress.Install.2005.dll	Assembly
AddinExpress.MSO.2005.dll	Assembly
extensibility.dll	Assembly
Microsoft.Office.Interop.Excel.dll	Assembly
Microsoft.Vbe.Interop.dll	Assembly
Office.dll	Assembly
Primary output from MyAddin3 (Active)	Output

Custom Actions


Custom Actions (Setup1)
Custom Actions
Install
AddinExpress.Install.2005.dll
Commit
Rollback
AddinExpress.Install.2005.dll
Uninstall
AddinExpress.Install.2005.dll

Open the Custom Actions editor and add a new action to the Install, Rollback, Uninstall sections. Use the AddinExpress.Install.dll / AddinExpress.Install.2005.dll (for VS 2005) assembly as an item for the custom actions.

CustomActionData

Add the following parameter to the CustomActionData property of all the custom actions mentioned above depending on your project type.

- **For COM Add-ins**

```
/Addin="[TARGETDIR]\<the add-in assembly name>.dll"
```

- **For RTD Servers**

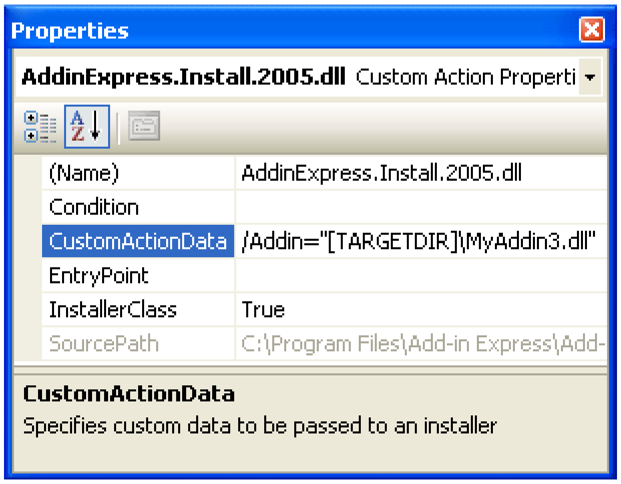
```
/RTD="[TARGETDIR]\<the RTD server assembly name>.dll"
```

- **For Smart Tags**

```
/SmartTag="[TARGETDIR]\<the smart tag assembly name>.dll"
```

- **For Excel Automation Add-ins**

```
/Addin="[TARGETDIR]\<the add-in assembly name>.dll"
```



(Name)	AddinExpress.Install.2005.dll
Condition	
CustomActionData	/Addin="[TARGETDIR]\MyAddin3.dll"
EntryPoint	
InstallerClass	True
SourcePath	C:\Program Files\Add-in Express\Add-

CustomActionData
Specifies custom data to be passed to an installer

Dependencies

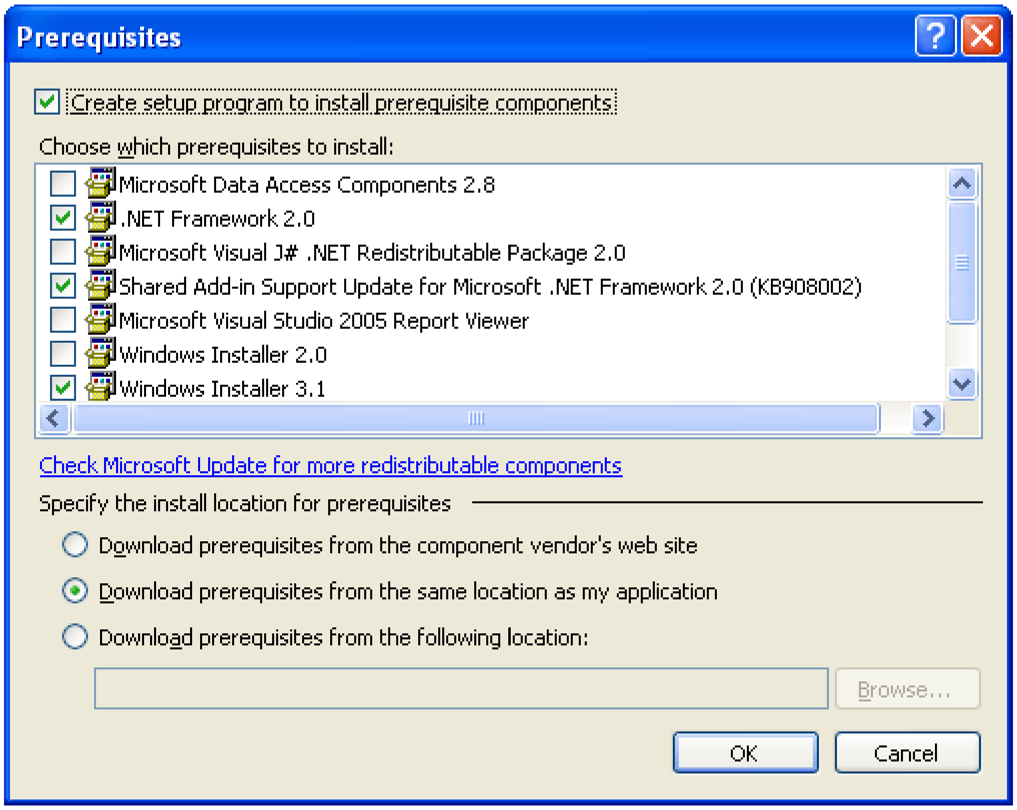
Right click on the Detected Dependencies secton of the setup project and choose the Refresh Dependencies option. Also, exclude all dependencies that are not required for your setup.

VS 2005 only

Right click on the setup project and open the Properties dialog.

Click on the Prerequisites button and, in the Prerequisites dialog, check all prerequisites you need.

You can choose the ‘Download prerequisites from the same location as my application’ option to distribute all prerequisites with the add-in installation package.



Prerequisites

[✓] Create setup program to install prerequisite components

Choose which prerequisites to install:

- [ ] Microsoft Data Access Components 2.8
- [✓] .NET Framework 2.0
- [ ] Microsoft Visual J# .NET Redistributable Package 2.0
- [✓] Shared Add-in Support Update for Microsoft .NET Framework 2.0 (KB908002)
- [ ] Microsoft Visual Studio 2005 Report Viewer
- [ ] Windows Installer 2.0
- [✓] Windows Installer 3.1

Check Microsoft Update for more redistributable components

Specify the install location for prerequisites

- ( ) Download prerequisites from the component vendor's web site
- [✓ Download prerequisites from the same location as my application]
- ( ) Download prerequisites from the following location:

Deploy

Rebuild the setup project. Copy all setup files to the target PC and run the setup.exe file to install the add-in.

Add-in Express Tips and Notes

You might have an impression that creating add-ins is a very simple task. Please don't get too enthusiastic. Sure, Add-in Express makes embedding your code into Office applications very simple, but you should write the applied code yourself, and we guess it would be something more intricate than a single call of MessageBox.

Here we describe some important issues you will encounter when developing your add-ins.

Terminology

In this document, on our site, and in all our texts we use the terminology suggested by Microsoft for all toolbars, their controls, and for all interfaces of the Office Type Library. For example:

- Command bar is a toolbar, a menu bar, or a context menu.
- Command bar control is one of the following: a button, an edit box, a combo box, or a pop-up.
- Pop-up can stand for a pop-up menu, a pop-up button on a command bar or a submenu on a menu bar.

Add-in Express uses interfaces from the Office Type Library. We do not describe them here. Please refer to the VBA help and to application type libraries.

.NET Framework 1.1 and 2.0 Installed on the Development PC

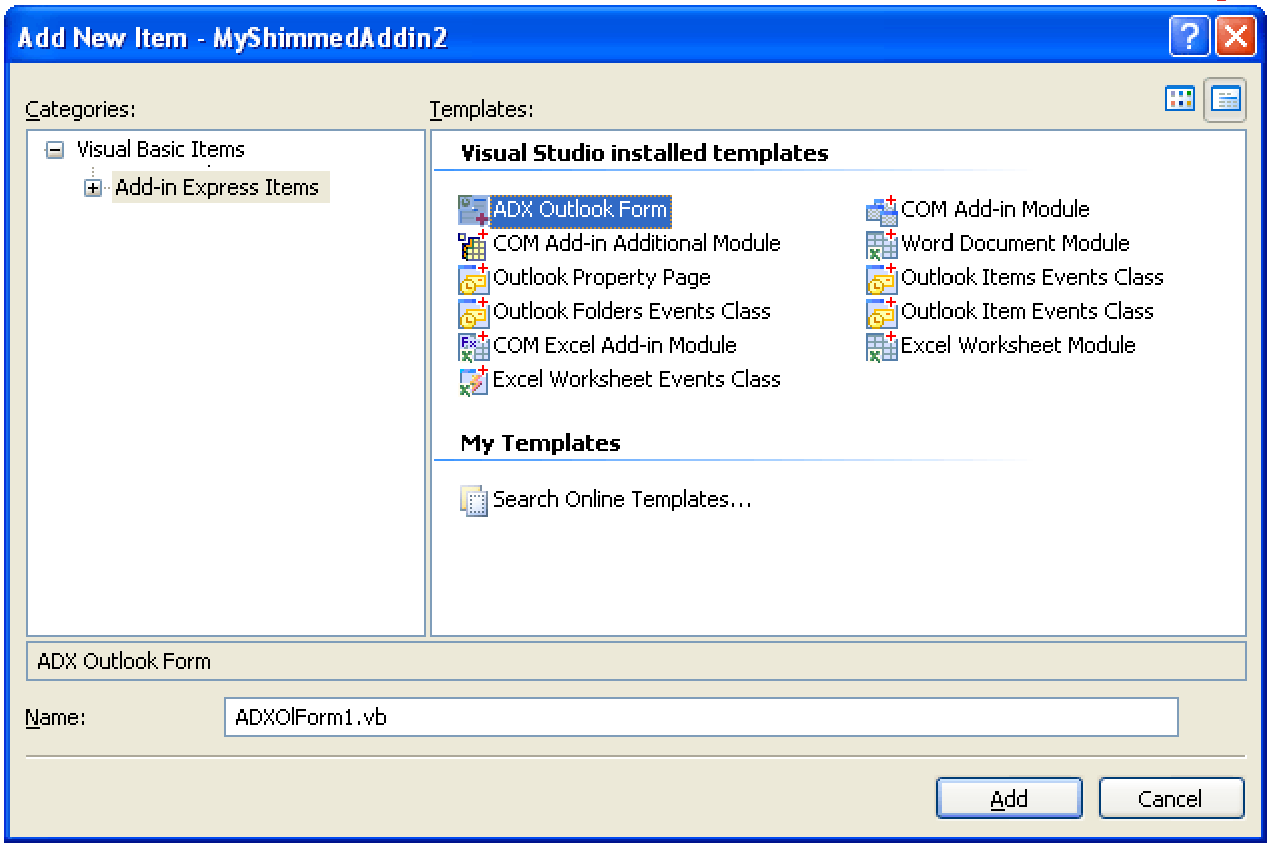
In a mixed .NET Framework environment, you may need to use the Host Configuration command of an Add-in Express Module. This command [creates and] changes the configuration file for your host application. The examples of configuration file names are outlook.exe.config and excel.exe.config. The file is located in C:\Program Files\Microsoft Office\OFFICE11.

Getting Help on COM Objects, Properties and Methods

To get assistance with host applications' objects, their properties and methods as well as help info, use the Object Browser. Go to the VBA environment (in the host application, choose menu Tools / Macro / Visual Basic Editor or just press Alt+F11), press F2, select the host application (also Office and MSForms) in the topmost combo and/or specify a search string in the search combo.

Add New Item Dialog

Add-in Express .NET installs the following items to the Add New Item dialog (right-click you project item in the Solution Explorer and choose the Add New Item menu).



- Add-in Express Outlook Form – a form designed for embedding into the Outlook Explorer and Inspector windows. To use this functionality, you will have to install an appropriate Add-in Express .NET package. See the Packages & Pricing page for details.
- COM Add-in Additional Module – it is an additional add-in module. Really. It will supplement your main module in case it has grown up too big in size.
- Outlook Property Page – the form designed for extending Outlook Options and Folder Properties dialogs with custom pages. See Outlook Property Page and Your First Microsoft Outlook COM Add-in.
- Outlook Folders Event Class – provides easy access to the events of the Folders class of Outlook.
- COM Excel Add-in Module – this module implements the functionality of the Excel Automation Add-in. You may need it only if you want to add new functions that can be used in Excel formulas.
- Excel Worksheet Event Class – provides easy access to the events of the Worksheet class.
- COM Add-in Module – the core of any Add-in Express COM add-in. See COM Add-ins.
- Word Document Module – allows handling events of any MS Forms controls placed on a specified Word document. See Word Documents.
- Outlook Items Event Class – provides easy access to the events of the Items class of Outlook.
- Outlook Item Event Class – provides easy access to the events of the MailItem, TaskItem, ContactItem, etc classes of Outlook.

- Excel Worksheet Module – allows handling events of any MS Forms controls placed on a specified Excel worksheet. See Excel Workbooks.

Add the COM Add-ins Command to a Toolbar or Menu

To add the COM Add-ins command to a toolbar or menu you do the following:

- Open the host application (Outlook, Excel, Word, etc)
- On the Tools menu, click Customize.
- Click the Commands tab.
- In the Categories list, click the Tools category.
- In the Commands list, click COM Add-Ins and drag it to a toolbar or menu of your choice.

How to Get Access to the Add-in Host Applications

The Add-in Module provides the HostApplication property that returns the Application object (of the Object type) of the host application the add-in is currently running in. For your convenience, the Add-in Express Project Wizard adds host-related properties to the Add-in module, such as OutlookApp and ExcelApp. To identify the host application, you can also use the HostName property of the module.

The previous example is intended for demonstration purposes only. We created it to let you get acquainted with the general principles of Add-in Express. In your future add-ins, you may want to get access to objects of host applications. Add-in Express allows accessing them through the HostApplication property of the add-in module. For example, you can access an active Excel worksheet by calling HostApplication.ActiveSheet or you can access an active explorer by calling HostApplication.ActiveExplorer.

Unfortunately, we can't describe here in detail how to work with all objects of MS Office applications. The reason is simple - there are too many of them (you can count yourself: Excel, Word, Outlook, PowerPoint, Access, Project, FrontPage, MapPoint, Visio, Publisher), while we have limited staff. So we recommend you to use the VBA online help of Office applications.

What is ProgId?

ProgID – Program Identifier. This is a textual name that represents a server object. It consists of the project name and the class name, like MyServer.MyClass.

You find it in ProgIDAttribute of an Add-in Express Module. For instance:

```
...
'Add-in Express Add-in Module
<GuidAttribute("43F48D82-7C6F-4705-96BB-03859E881E2C"), _
    ProgIdAttribute("MyAddin1.AddinModule")> _
```

```
   Public Class AddinModule
       Inherits AddinExpress.MSO.ADXAddinModule
   ...
```

Note

We found the definition of ProgId in [The COM / DCOM Glossary](). On that page, you can find other COM-related terms and their definitions.

Registry Entries

COM Add-ins registry entries are located in the following registry branches:

```
   HKEY_CURRENT_USER\Software\Microsoft\Office\<OfficeApplication>\AddIns\<Add-
in ProgID>
   HKEY_CLASSES_ROOT\CLSID\<Add-in Express Project GUID>
```

ControlTag vs. Tag Property

Add-in Express identifies all its controls (command bar controls) through the use of the ControlTag property (the Tag property of the CommandBarControl interface). The value of this property is generated automatically and you don't need to change it. For your own needs, use the Tag property instead.

Pop-ups

According to the Microsoft's terminology, the term "pop-up" can be used for several controls: pop-up menu, pop-up button, and submenu. With Add-in Express you can create your own pop-up as an element of your controls command bar collection and add to it any control via the Controls property.

But pop-ups have a feature that is very annoying: if an edit box or a combo box is added to a pop-up, their events are fired very oddly. Don't regard this bug as that of Add-in Express. It seems to be intended by MS.

Edits and Combo Boxes and the Change Event

The Change event appears only when the value was changed and the focus was shifted. This is also not our bug but MS guys' "trick".

Built-in Controls and Command Bars

You can connect an ADXCommandBar instance to any built-in command bar. For example, you can add your own controls to the "Standard" command bar or remove some controls from it. To do this just add to the add-in module a new ADXCommandBar instance and specify the name of the built-in command bar you need via the CommandBarName property.

Also, you can add any built-in controls to your own command bars. To do this just add an ADXCommandBarControl instance to the ADXCommandBar.Controls collection and specify the Id of the built-in control you need via the Id property.

CommandBar.SupportedApps

Use this property to specify if the command bar will appear in some or all host applications supported by the add-in.

Outlook CommandBar Visibility Rules

Add-in Express displays the Explorer command bar for every folder, which name **AND** type correspond to the values of FolderName, FolderNames, and ItemTypes properties. For the Inspector toolbar, the same rule applies to the folder in which an Outlook Item is opened or created.

COM Add-ins for Outlook - Template Characters in FolderName

Regardless of the fact that the default value of the FolderName property is '*' (asterisk), which means "every folder", the current version doesn't support template characters in the FolderName(s) property value. Moreover, this is the only use of the asterisk recognizable in the current version.

Removing Custom Command Bars and Controls

Add-in Express removes custom command bars and controls while add-in is uninstalled. However, this doesn’t apply to Outlook and Access add-ins. You should set the Temporary property of custom command bars (and controls) to true to notify the host application that it can remove them itself. If you need to remove a toolbar or button yourself, use the Tools | Customize dialog.

Deploying - Shadow Copy

The Add-in Express Loader uses the ShadowCopy-related properties and methods of the AppDomain class. When you run your Add-in Express add-in, the host application loads the Add-in Express Loader DLL referenced in the registry. The Add-in Express Loader DLL does the following:

- It finds your add-in DLLs in the DLL Cache. If there are no add-in DLLs in the cache it copies all .NET DLLs to the cache (including dependencies). The cache folder is located in c:\Documents and Settings\<user name>\Local Settings\Application Data\assembly\dl<number>. If all add-in DLLs (including dependencies) already exist in the cache, it compares their versions. If the versions are not the same, it copies new DLLs to the cache.
- It loads the add-in DLLs from the cache.

You can see how the add-in versioning influences the add-in loading.

This approach (it is built into .NET, as you see) allows you to replace add-in DLLs when the add-in is loaded. The disadvantage is numerous files located in the cache. As far as we know, MS doesn't provide a solution for this problem. You may think you can remove these files in an add-in's uninstall custom action. But you remove the files from the current profile only.

My Add-in Is Always Disconnected

If your add-in fires exceptions at the startup, the host application can block the add-in and move it to the Disabled Items list. To find the list, in the host application, go to "Help", then "About". At the bottom of the About dialog, there is the Disabled Items button. Check it to see if the add-in is listed there (if so, select it and click the enable button).

No RTD Servers in EXE

Add-in Express currently supports RTD Servers in DLLs only.

Update Speed for an RTD Server

Microsoft limits the minimal interval between updates to 2 seconds. There is a way to change this minimum value but Microsoft doesn't recommend doing this.

FolderPath Property Value in Outlook 2000 and XP

The function returns the same value as the MAPIFolder.FolderPath property available in Outlook 2003+.

```
Private Function GetFolderPath(ByVal MAPIFolder As _
   Outlook.MAPIFolder) As String

   If MAPIFolder Is Nothing Then Return ""

   Dim CurrentPath As String = ""
   Do
      CurrentPath = "\" + MAPIFolder.Name + CurrentPath
```

```
                Dim Obj As Object = MAPIFolder.Parent
                'The parent of a root folder is of the Outlook.Namespace type
                If Not TypeOf Obj Is Outlook.MAPIFolder Then Exit Do
                MAPIFolder = CType(Obj, Outlook.MAPIFolder)
            Loop

            CurrentPath = "\" + CurrentPath
            Return CurrentPath
        End Function
```

Installing Your Solutions on Vista

If you install you add-in or RTD server for all users on the PC (RegisterForAllUsers = True), we strongly recommend installing your add-ins on Vista using the Setup.exe file that is automatically generated by the setup project. Note, in this case, we don't recommend using the .msi file generated by the same setup project. Nevertheless, you still need to supply both files (as well as other files generated by the setup project) to the end user.

But if you install your add-in or RTD server for the current, say non-admin, user only (RegisterForAllUsers = False) in order to avoid the Remove / Repair dialog during the second and successive installations on the same PC, we recommend using the .msi file to install the add-in.

Note, Smart Tags and Excel Automation add-ins are installed using the profile of the user that runs the installer. So, you have to use the .msi file too.

Also, if you don't use the Add-in Express Loader as a shim, you must always use the setup.exe file.

Final Note

If your questions are not answered here, please see the HOWTOs section on www.add-in-express.com. We are constantly updating these pages.