Official Guide

Add-in Express™ VCL

Getting Started

Add-in Express 2007 for VCL

Add-in Express™ 2007 for VCL

Document version **3.0**

Revised at **27-Dec-06**

Product version **3.x**

Table of Contents

Introduction

Add-in Express VCL 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 Delphi 5, 6, 7, 2005, 2006 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 – 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 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 VCL for Microsoft Outlook**

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

- **Outlook Security Manager**

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

System Requirements

Supported Delphi Versions

- Delphi 5 Architect with Update Pack 1
- Delphi 5 Enterprise with Update Pack 1
- Delphi 5 Professional with Update Pack 1
- Delphi 6 Architect with Update Pack 2
- Delphi 6 Enterprise with Update Pack 2
- Delphi 6 Professional with Update Pack 2
- Delphi 7 Architect with Update Pack 1
- Delphi 7 Enterprise with Update Pack 1
- Delphi 7 Professional with Update Pack 1
- Delphi 2005 for VCL Architect with Update Pack 3
- Delphi 2005 for VCL Enterprise with Update Pack 3
- Delphi 2005 for VCL Professional with Update Pack 3
- Delphi 2006 for VCL Architect with Update Pack 2
- Delphi 2006 for VCL Enterprise with Update Pack 2
- Delphi 2006 for VCL Professional with Update Pack 2

Note

You should have Microsoft Office 2000 Sample Automation Server Wrapper Components installed.

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

What follows below is a brief guide on activating your copy of Add-in Express.

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 setup program installs the following folders on your PC:

- Packages – Add-in Express design-time packages for Delphi 5, 6, 7, 2005, 2006

- Demo Projects – demo projects for Delphi 7
- Docs – Add-in Express documentation
- Docs \ Samples – sample documentation projects
- Sources – Add-in Express VCL source code
- Sources \ DesignTime – design-time source code.

Where is the Add-in Express 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 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 Module to the project
- Add Add-in Express components to the Add-in Express Module
- Add some business logics
- Build, register, and debug the Add-in Express project
- Deploy your project to a target PC

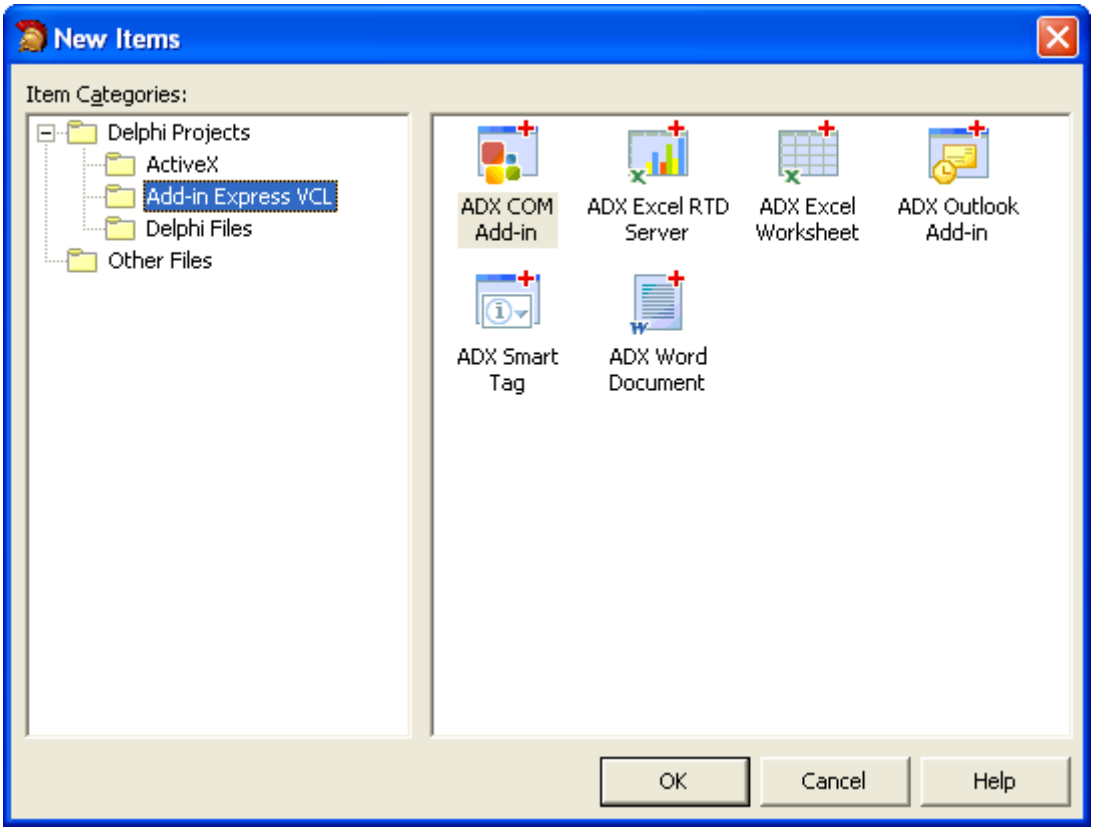
For the sake of time, in our support emails we use the ADX.VCL acronym when referring to Add-in Express .VCL.

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

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

Creating Add-in Express Projects

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



Whichever Add-in Express Project template you use, it starts the Add-in Express Project Wizard that allows selecting Add-in Express project options. The Add-in Express Project Wizard creates the Add-in Express project and adds an appropriate Add-in Express Module to the project (see Add-in Express Modules).

Add-in Express Modules are the core components of Add-in Express. 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 Modules

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

- COM Add-ins - TadxCOMAddinModule
- RTD Servers – TadxXLRTDServerModule

- Smart Tags - TadxSmartTagModule
- Excel Automation Add-ins - TadxCOMAddinModule
- Excel Workbooks - TadxExcelSheetModule
- Word Documents - TadxWordDocumentModule

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.

TadxCOMAddinModule represents a COM Add-in in any Office application. To add another add-in to your assembly, add another TadxCOMAddinModule 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 See add-ins, you specify the Options page and Folder Property pages (see Outlook Property Page). Look the following chapters for the Add-in Express components you add onto the TadxCOMAddinModule: 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.

TadxXLRTDServerModule 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.

TadxSmartTagModule 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 provides developing Excel Automation Add-ins using the COM Add-in Module that has the XLAutomationAddin Boolean property. Set the property to true, add a method to the add-in module's type library, and write the method's code.

Excel Workbooks

Sometimes you need to automate a given Excel workbook (template). You can do it with TadxExcelSheetModule that represents one worksheet of the workbook. For the module to recognize the workbook, you need to fill the following properties: Document, Worksheet, PropertyID, and PropertyValue. When you fill the PropertyID and PropertyValue properties, the design-time code of the module creates the property in the workbook and specifies its value.

A typical scenario of the module usage includes creating the workbook and designing it with MS Forms controls. Accordingly, in the IDE, you set up the PropertyID and PropertyValue properties, add Add-in Express MSForms control components to the module and bind them to the MS Forms controls on the worksheet. The module provides a full set of events available for the Excel Workbook class.

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

Note

We regard TadxExcelSheetModule and TadxWordDocumentModule as deprecated and do not recommend using them.

Word Documents

To automate a given Word document, you use the TadxWordDocumentModule. You can do it with TadxExcelSheetModule that represents a document. For the module to recognize the document, you need to fill the following properties: Document, PropertyID, and PropertyValue. When you fill the PropertyID and PropertyValue properties, the design-time code of the module creates the property in the document and specifies its value.

A typical scenario of the module usage includes creating the document and designing it with MS Forms controls. Accordingly, in the IDE, you set up PropertyID and PropertyValue properties, add Add-in Express MSForms control components to the module and bind them to the MS Forms controls on the document. The module provides a full set of events available for the Word Document class.

For the Add-in Express components available for the module see the following chapters: Command Bars, Command Bar Controls, Built-in Control Connector, MSForms Controls, and Application-level Events. The module provides the full set of events available for Word document.

Note

We regard TadxExcelSheetModule and TadxWordDocumentModule 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

You find all the Add-in Express components in the Add-in Express category on the Tool Palette.

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 TadxRibbonTab, TadxRibbonOfficeMenu and TadxRibbonQAT (Quick Access Toolbar) 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.

TadxRibbonTab and TadxRibbonOfficeMenu 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 providing the appropriate window in the Add-in Express project wizard and equipping the COM Add-in module with the TaskPanes property. Use the Add-in Express COM Add-in project wizard to add a task pane(s) to your project. Add your reaction to the OnTaskPaneXXX event series of the COM Add-in module and the OnDockPositionStateChange and OnVisibleStateChange 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 TadxCommandBar, TadxOlExplorerCommandBar, ad TadxOlInspectorCommandBar components.

The main property of any command bar 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.

What follows below is a list of controls available for Add-in Express command bars:

- TadxCommandBarButton
- TadxCommandBarComboBox
- TadxCommandBarEdit
- TadxCommandBarPopup
- TadxCommandBarDropDownList
- TadxCommandBarControl (you use this item to add built-in controls to your command bars)
- TadxCommandBarAdvancedControl (reserved for future use).

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 TadxBuiltInControl onto TadxCOMAddinModule. 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 TadxCOMAddinModule, 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 TadxCOMAddinModule. 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 TadxOlBarShortcut and/or TadxOlBarGroup 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 TadxOlBarShortcut 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 VCL forms into the Outlook Explorer and Inspector windows and replace folder views with custom VCL forms.

The Outlook Forms Manager component is available for TadxCOMAddinModule only. It is the core component of the Add-in Express Extensions VCL for Microsoft Outlook which is a plug-in for Add-in Express VCL. You find the detailed information on this product at http://www.add-in-express.com/outlook-extension/. Once again, you don't have this component if you don't have this product bought as part of any Add-in Express product packages.

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 in the Add-in Express Outlook Add-in project wizard.

Note, the FolderName, FolderNames, and ItemTypes properties of the Add-in Express Outlook folder pages work in the same way as those of Outlook-specific command-bars.

Specify reactions required by your business logics in the Apply event handler. In the page controls' event handlers, use the UpdatePropertyPageSite method to mark the page as Dirty.

Smart Tag

The Kind property of the TadxSmartTag 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

Add-in Express provides event components for all Office applications on the Tool Palette: Just add appropriate Add-in Express event components to the module, and use their event handlers to respond to the host application's events.

MSForms Controls

Add-in Express provides MS Forms control components on the Tool Palette These components are to be used on the TadxExcelSheetModule and TadxWordDocumentModule. Add 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 TadxExcelSheetModule and TadxWordDocumentModule as deprecated and do not recommend using them.

Your First Microsoft Office COM Add-in

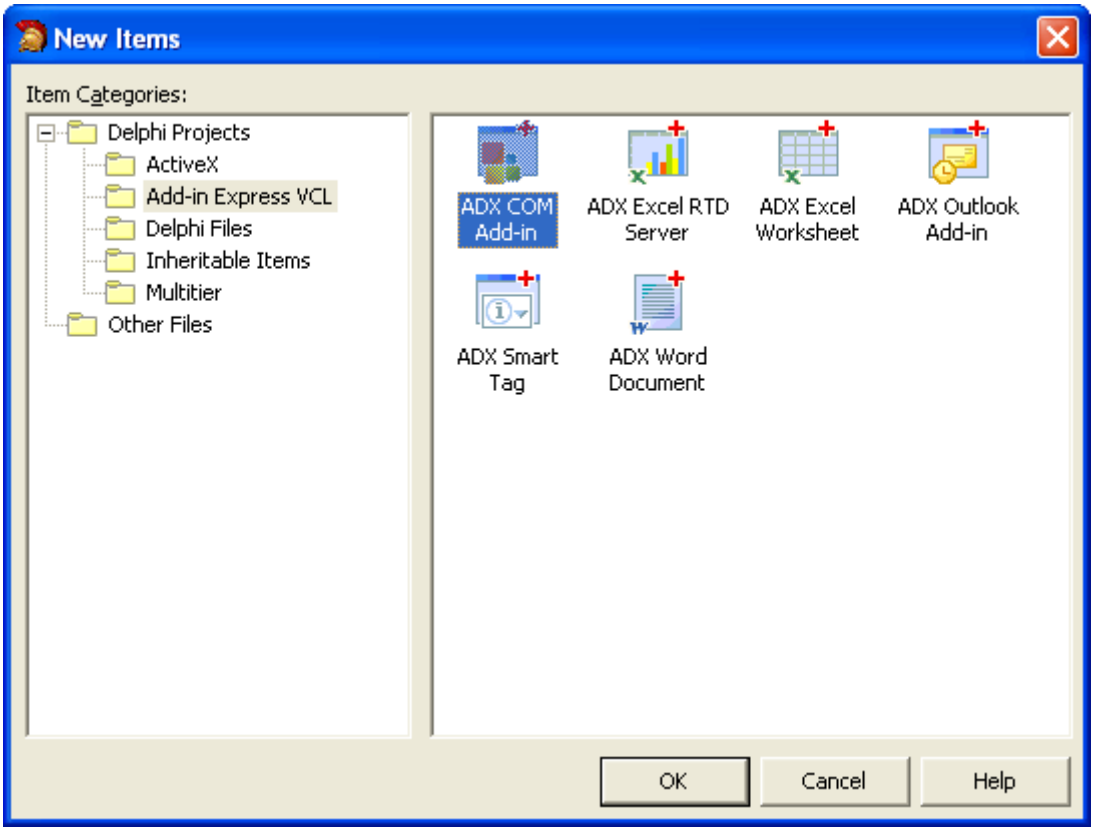
This chapter highlights almost every aspect of creating COM Add-ins for Microsoft Office applications. You find this sample in the Docs \ Samples folder of the Add-in Express install folder. This sample shows an Add-in Express COM Add-in project implementing a COM add-in for Excel and Word.

Outlook and Add-in Express

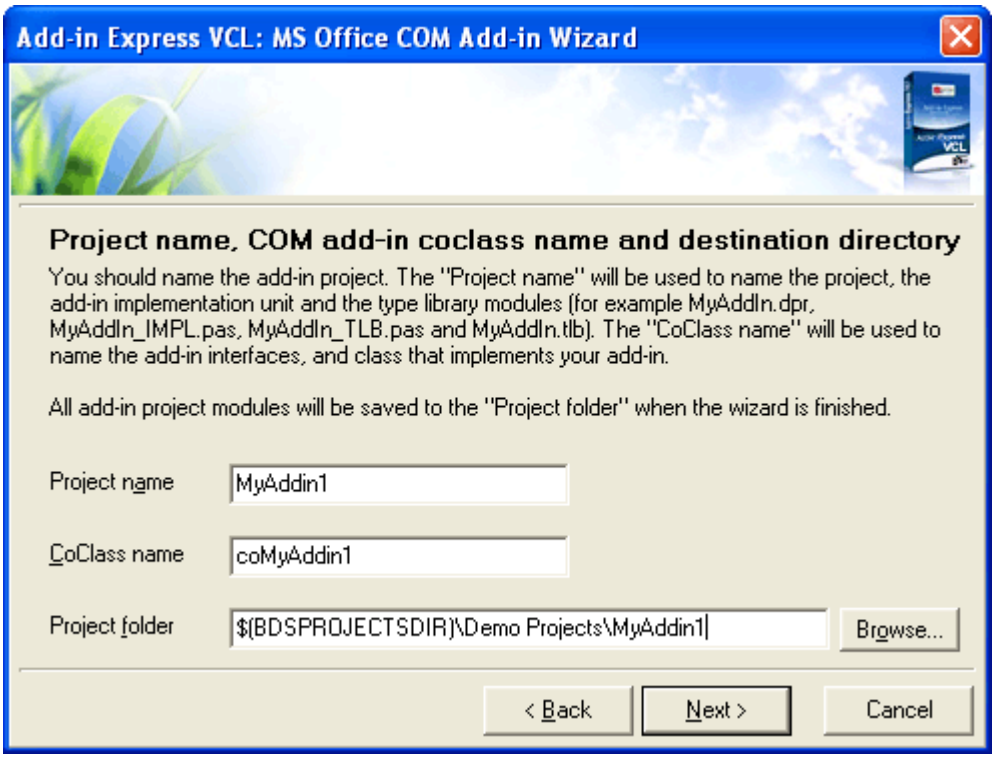
Please note, Add-in Express provides additional components for COM Add-ins in Outlook. See Your First Microsoft Outlook COM Add-in.

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 New Item dialog of Borland Delphi for Microsoft Windows.



When you select the template and click OK, the MS Office COM Add-in Wizard starts. In the wizard windows, you choose the project options.



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

The add-in project includes the following items:

- The project source files (ProjectName.*);
- The type library files: binary (ProjectName.tlb) and Object Pascal unit (ProjectName_TLB.pas);
- The COM add-in module (ProjectName_IMPL.pas and ProjectName_IMPL.dfm) discussed in the following step.

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

The COM Add-in Module (MyAddin1_IMPL.pas and MyAddin1_IMPL.dfm) is the core component of the COM add-in project (see COM Add-ins). It is the container of the Add-in Express components, which allow you to concentrate on 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.

The code for MyAddin1_IMPL.pas is as follows:

```
unit MyAddin1_IMPL;

interface

uses
  SysUtils, ComObj, ComServ, ActiveX, Variants, adxAddIn, MyAddin1_TLB;

type
  TcoMyAddin1 = class(TadxAddin, IcoMyAddin1)
  end;

  TAddInModule = class(TadxCOMAddInModule)
  private
  protected
  public
  end;

implementation

{$R *.dfm}

initialization
  TadxFactory.Create(ComServer, TcoMyAddin1, CLASS_coMyAddin1,
  TAddInModule);

end.
```

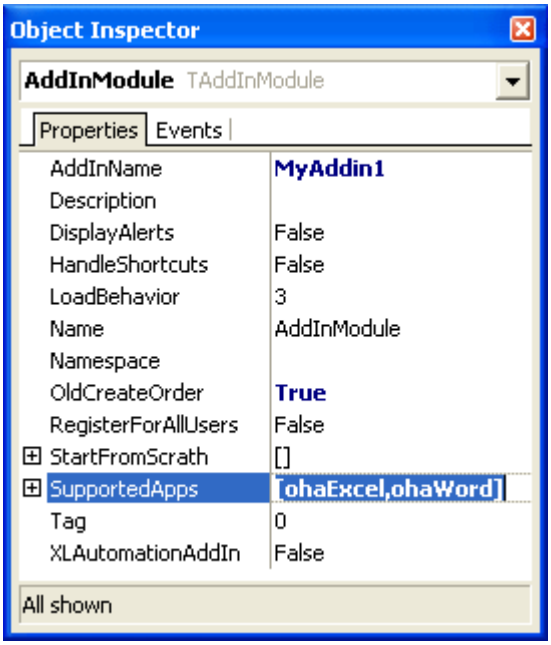
The add-in module contains two classes: the “interfaced” class (TcoMyAddin1 in this case) and the add-in module class (TAddInModule). The “interfaced” class is a descendant of the TadxAddIn class that implements the IDTExtensibility2 interface required by the COM Add-in architecture. Usually, you don't need to change anything in the TadxAddIn class.

The add-in module class implements the add-in functionality. It is an analogue of the Data Module, but unlike the Data Module, the add-in module allows you to set all properties of your add-in, handle its events, and create toolbars and controls.

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

The designer of COM Add-in Module allows setting add-in properties and adding components to the module.

In the Object Inspector for the module, choose the SupportedApps property and select Excel and Word.



To add an Add-in Express component to the AddinModule designer, you select it in the Tool Palette and drag-n-drop onto the designer.

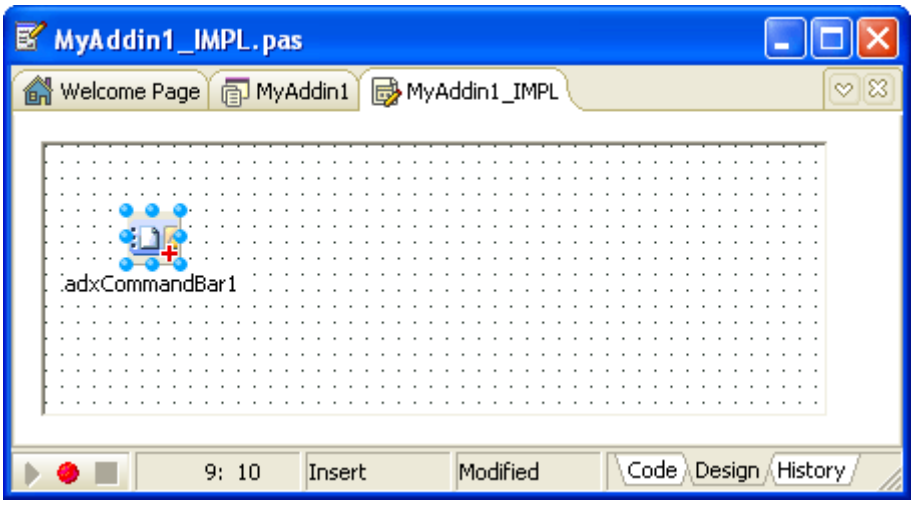
You can add the following Add-in Express components to the designer:

- TadxRibbonTab – represents a Ribbon tab in your add-in (see Office 2007 Ribbon Components)
- TadxRibbonQAT – represents the Ribbon Quick Access Toolbar your add-in (see Office 2007 Ribbon Components)
- TadxRibbonOfficeMenu – represents the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)
- TadxCommandBar – represents a command bar in your add-in (see Command Bars)
- TadxOlExplorerCommandBar – represents an Outlook Explorer command bar in your add-in (see Command Bars)
- TadxOlInspectorCommandBar – represents an Outlook Inspector command bar in your add-in (see Command Bars)
- TadxBuiltInControl – allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- TadxKeyboardShortcut – allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)

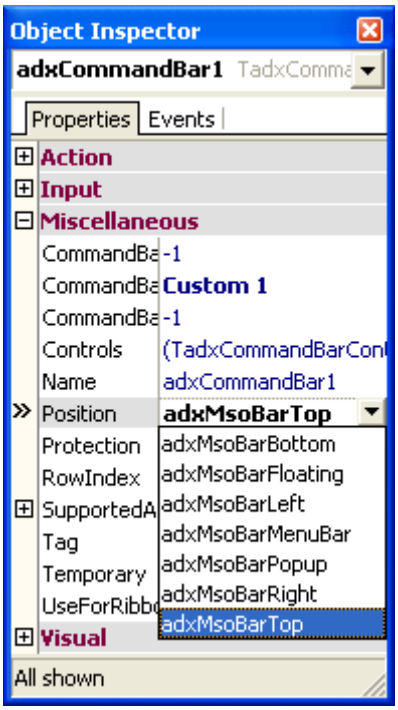
- TadxOlBarShortcutManager – allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- TadxOlFormsManager – embedding custom VCL forms into Outlook windows (see Outlook Forms Manager)
- Application Events – adds or deletes components that provide access to application-level 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, find the TadxCommandBar component in the Tool Palette and drag-n-drop it onto the TadxCOMAddinModule designer (see Command Bars).



Select the command bar component and, in the Object Inspector window, specify the command bar name using the CommandBarName property. Also, you select its position in the Position property.

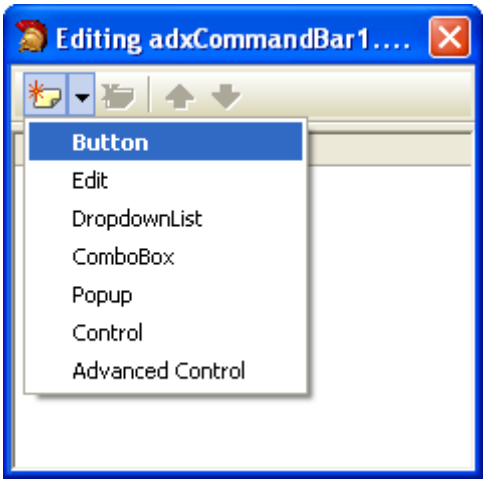


Command Bars in Office 2007

To display a 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 Object Inspector 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 Editing window, you select the command bar control type in the Add button (see also Command Bar Controls).

Specify the button's Caption property and set the Style property to adxMsoButtonIconAndCaption (default value = adxMsoButtonCaption). To handle the Click event of the button, in the Object Inspector window, switch to the Events tab and add the Click event handler: The code of the event handler follows below (it's empty as you can see)

```
procedure TAddInModule.adxCommandBar1Controls0Click(Sender: TObject);
begin

end;
```

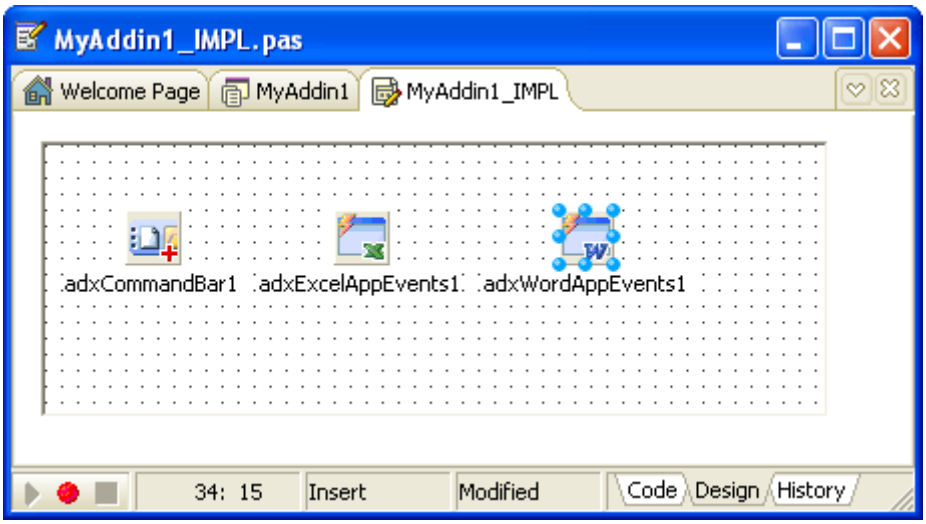
Step #6 – Accessing Host Application Objects

The Add-in Module provides the HostApplication property that returns the Application object (of the OleVariant type) of the host application the add-in is currently running in. For your convenience, Add-in Express provides the <HostName>App properties, say ExcelApp of the TExcelApplication type and WordApp of the TWordApplication type. Together with the HostType property, this allows us to write the following code to the Click event of the button just added.

```
procedure TAddInModule.adxCommandBar1Controls0Click(Sender: TObject);
begin
    case Self.HostType of
      ohaExcel:
        ShowMessage('The current cell is '
          + Self.ExcelApp.ActiveCell.AddressLocal
            [False, False, xlA1, EmptyParam, EmptyParam]);
      ohaWord:
        if Self.WordApp.Selection <> nil then
          ShowMessage('There are '
            + IntToStr(Self.WordApp.Selection.Words.Count)
            + ' words currently selected');
      else
        ShowMessage(Self.AddInName
            + ' doesn`t support the current host application');
    end;
end;
```

Step #7 – Handling Host Application Events

Add-n Express provides several components that provide the application-level events for the COM Add-in Module (see Application-level Events). To add Excel events to the add-in, find the TadxExcelAppEvents component in the Tool Palette and drag-n-drop it onto the module. The TadxWordAppEvents component supplies the module with Word events.



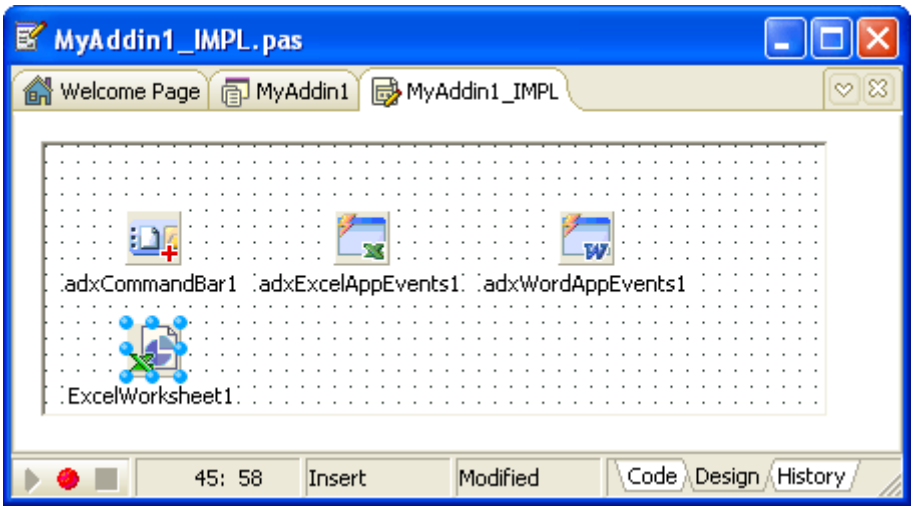
With the Events components, you handle any application-level events of the host application. You might see that the Click event handler in the previous step will fire an exception when there are no workbooks or documents open. To prevent this, you disable the button when a window deactivates and enable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

The code is as follows:

```
procedure TAddInModule.adxExcelAppEvents1WindowActivate(ASender: TObject;
   const Wb: _Workbook; const Wn: Window);
begin
    adxCommandBar1.Controls[0].Enabled := true;
end;

procedure TAddInModule.adxExcelAppEvents1WindowDeactivate(ASender: TObject;
   const Wb: _Workbook; const Wn: Window);
begin
    adxCommandBar1.Controls[0].Enabled := false;
end;

procedure TAddInModule.adxWordAppEvents1WindowActivate(ASender: TObject;
   const Doc: _Document; const Wn: Window);
begin
    adxCommandBar1.Controls[0].Enabled := true;
end;

procedure TAddInModule.adxWordAppEvents1WindowDeactivate(ASender: TObject;
   const Doc: _Document; const Wn: Window);
begin
    adxCommandBar1.Controls[0].Enabled := false;
end;
```

Step #8 – Handling Excel Worksheet Events

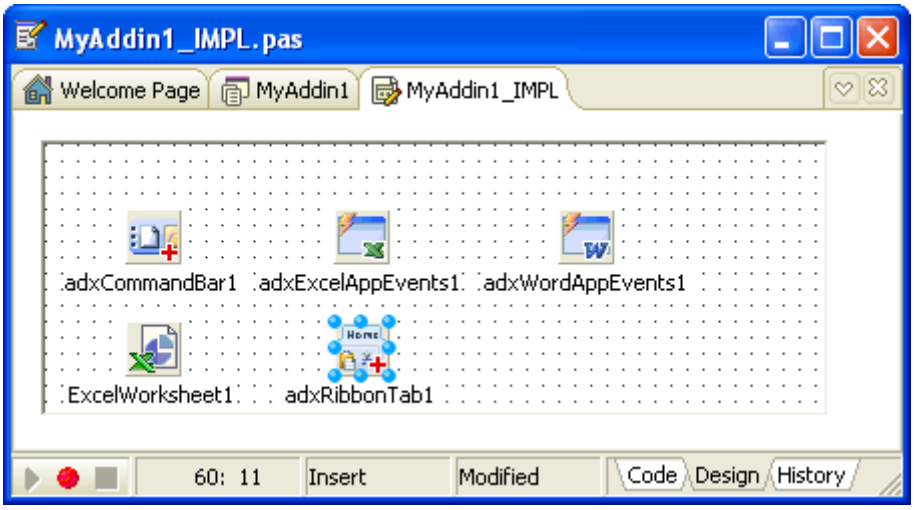
In the Tool Palette, find the Servers tab and add an instance of TExcelWorksheet to the add-in module. Now you can connect the instance to a worksheet, e.g. the active one.



Find the code for handling worksheet events in the code of this add-in.

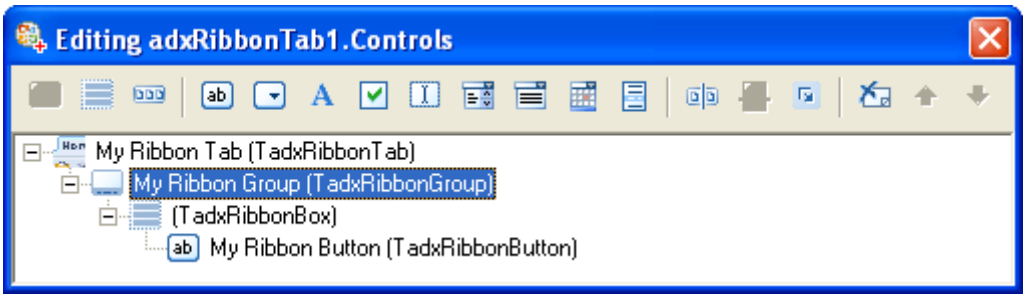
Step #9 – Customizing the Office 2007 Ribbon User Interface

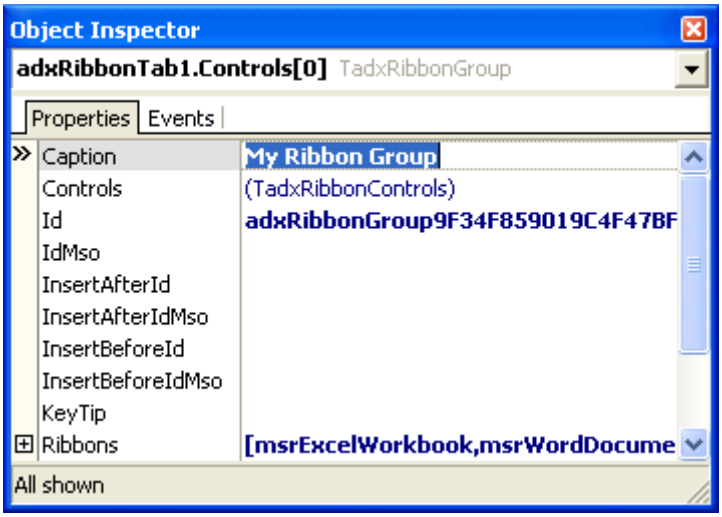
To add a new tab to the Ribbon, you add the TadxRibbonTab component to the module.



In the Object Inspector 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.





Now add the event handler to the Click event of the newly added Ribbon button. Write the following code:

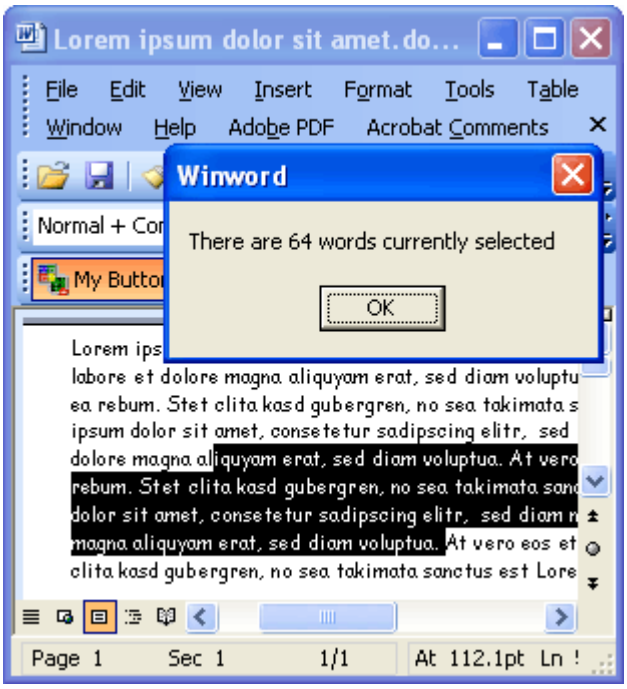
```
procedure TAddInModule.adxRibbonTab1Controls0Controls0Controls0Click(
  Sender: TObject; const RibbonControl: IRibbonControl);
begin
    adxCommandBar1Controls0Click(nil);
end;
```

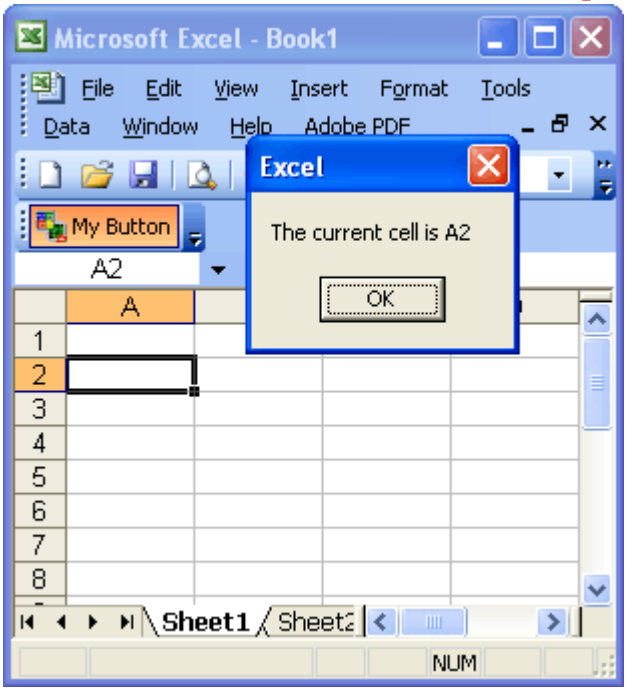
Remember, the TadxRibbonTab Controls editor performs the XML-schema validation automatically, so from time to time you will run into the situation when you cannot add a control to some Ribbon level. It is a restriction of the Ribbon XML-schema.

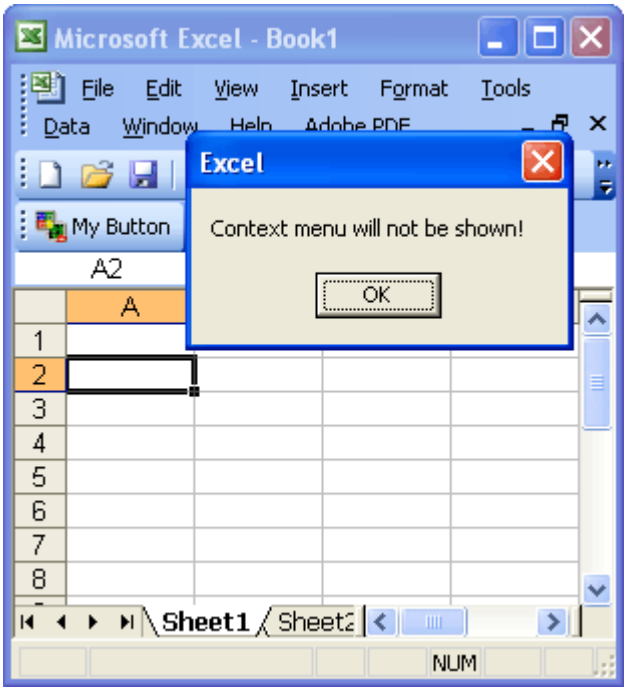
See also Office 2007 Ribbon Components.

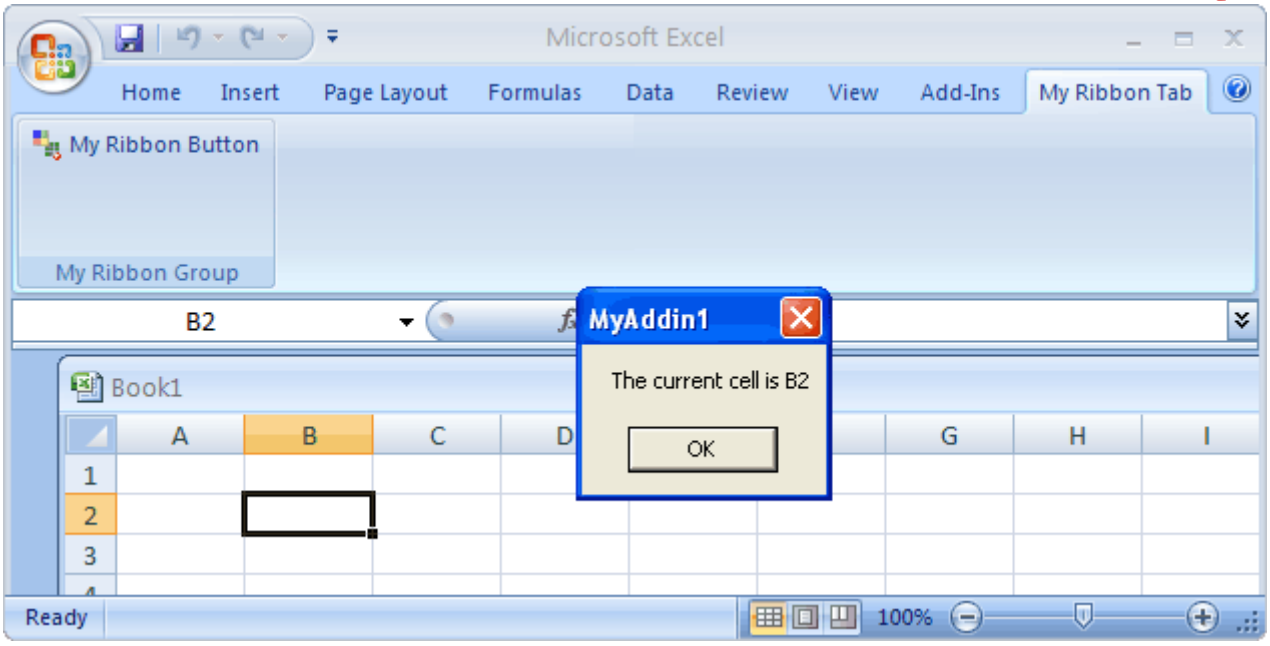
Step #10 – Running the COM Add-in

Choose the Register ActiveX Server item in the Run menu, restart the host application(s) you selected, find your toolbar and click the button.

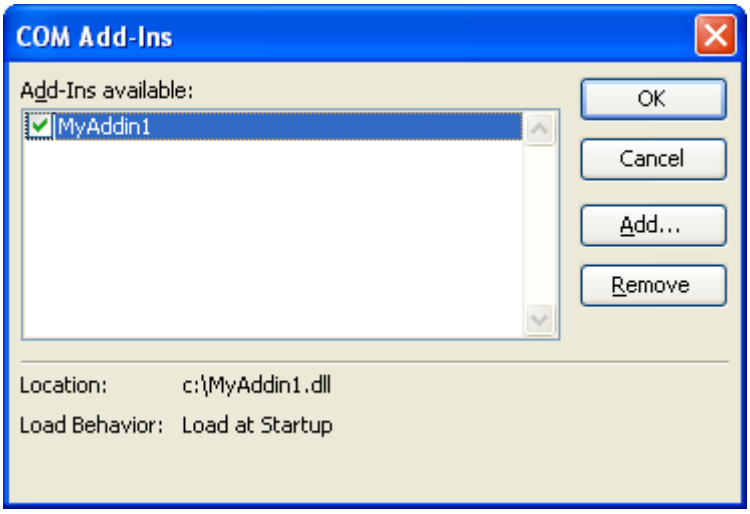

Lorem ipsum dolor sit amet.do...
File Edit View Insert Format Tools Table
Window Help Adobe PDF Acrobat Comments
Winword
There are 64 words currently selected
OK
My Button
Page 1 Sec 1 1/1 At 112.1pt Ln !


Microsoft Excel - Book1
File Edit View Insert Format Tools
Data Window Help Adobe PDF
Excel
The current cell is A2
OK
My Button
A2
A
1
2
3
4
5
6
7
8
Sheet1 Sheet2
NUM


Microsoft Excel - Book1
File Edit View Insert Format Tools
Data Window Help Adobe PDF
Excel
Context menu will not be shown!
OK
My Button
A2
A
1
2
3
4
5
6
7
8
Sheet1 Sheet2
NUM



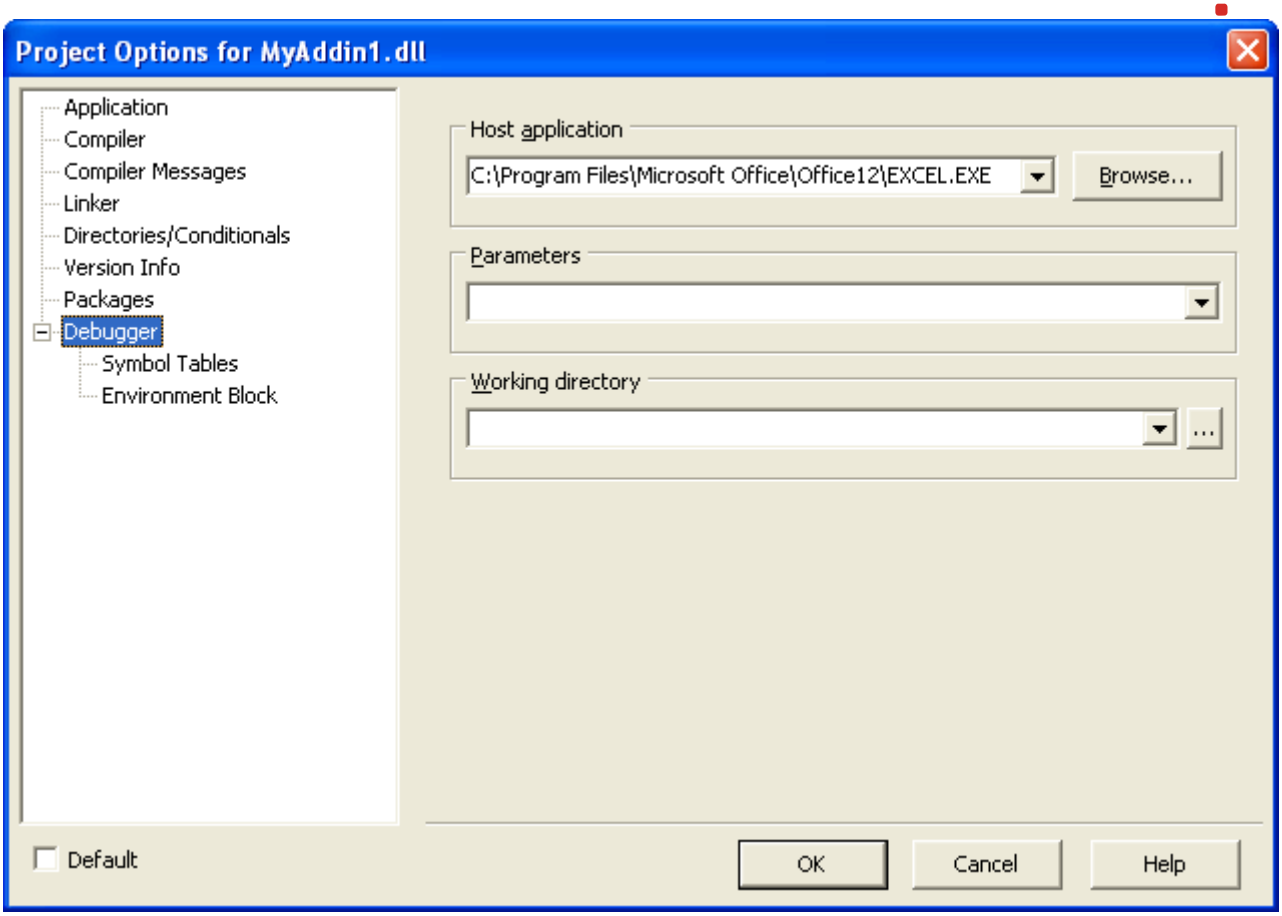
You find your add-in in the COM Add-ins dialog:



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 in the Host Application field in the Project Options window.



Step #12 – Deploying the COM Add-in

Make sure your setup project registers the add-in DLL. Say, in Inno Setup projects you use the 'regserver' command.

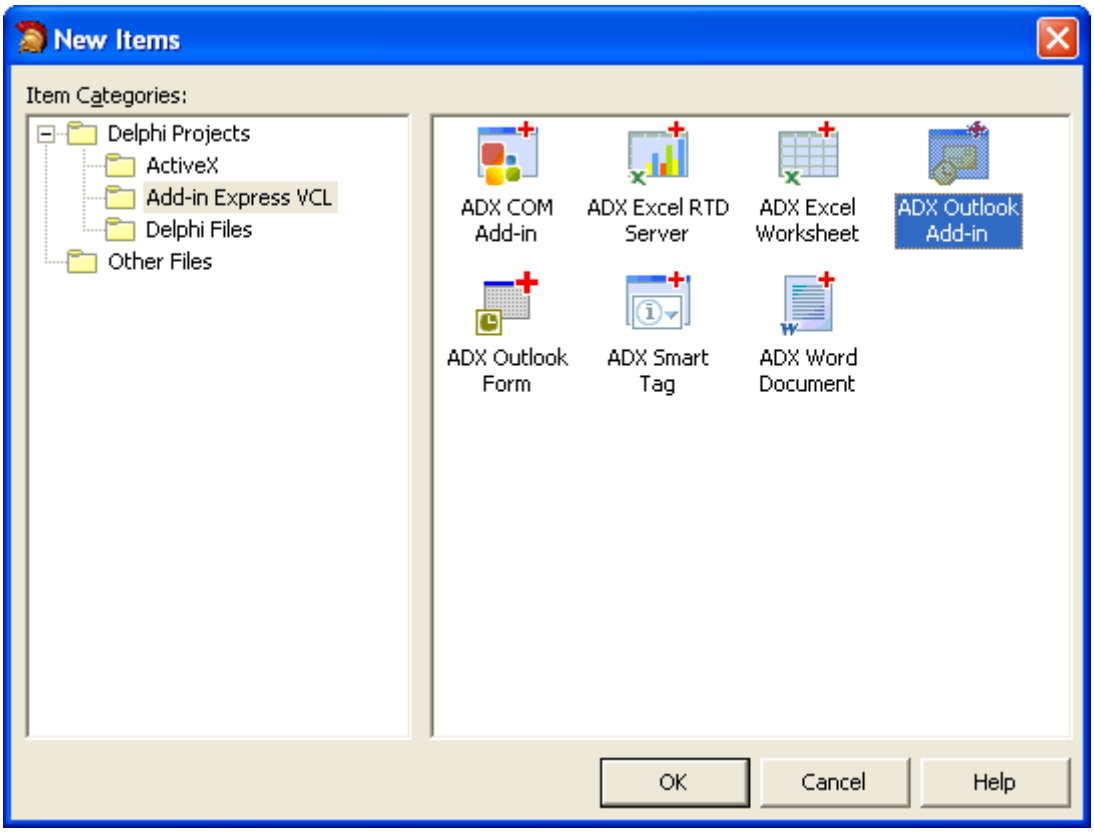
Your First Microsoft Outlook COM Add-in

Add-in Express provides the Outlook-specific COM add-in module and two Outlook-specific command bars: TadxOlExplorerCommandBar and TadxOlInspectorCommandBar. 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 command bar components 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.

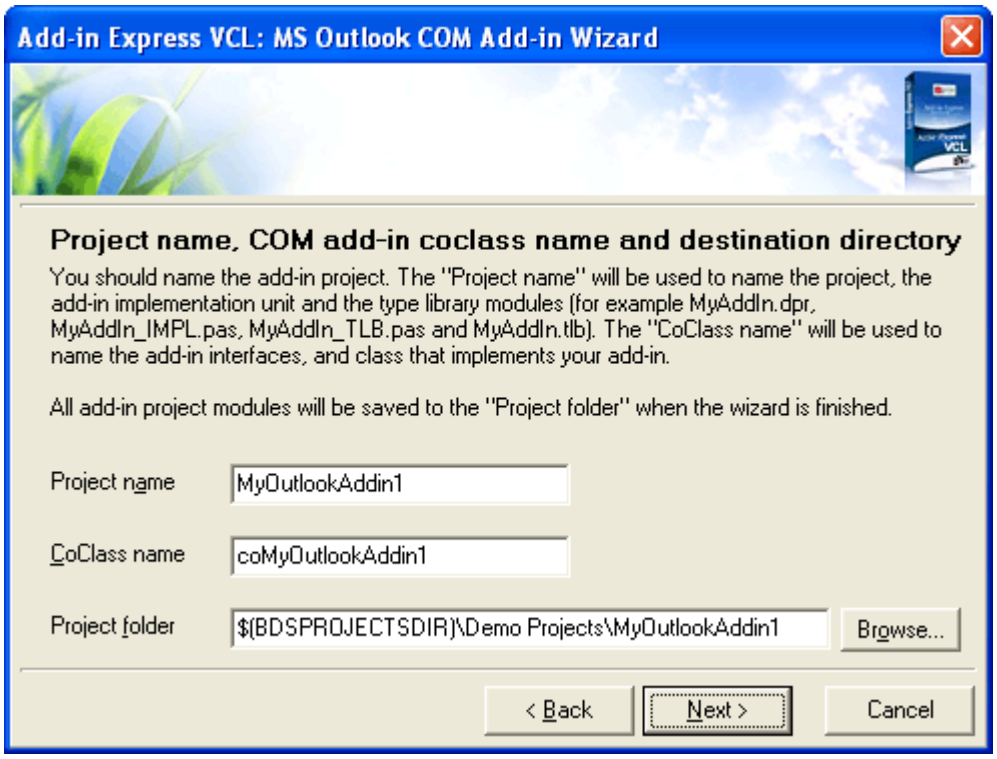
Additionally, Add-in Express Outlook Add-in wizards allows creating property pages to be added to the Options (Tools | Options menu) and folder Properties dialogs.

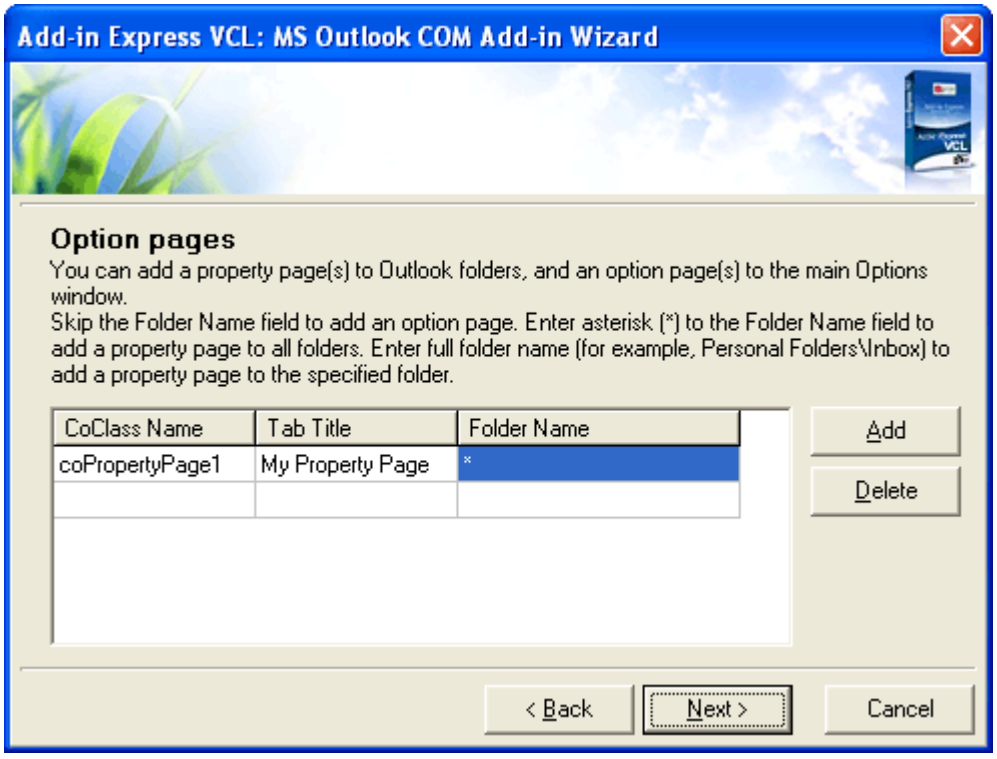
Step #1 – Creating an Add-in Express Outlook COM Add-in Project

Add-in Express adds the Add-in Express COM Add-in project template to the New Item dialog of Borland Delphi for Microsoft Windows.



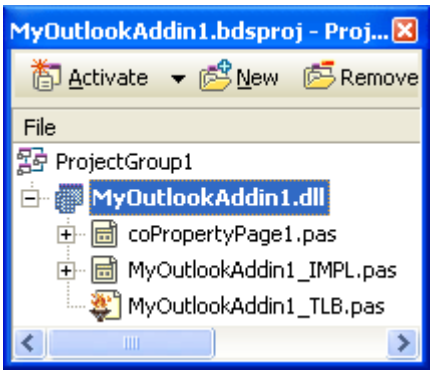
When you select the template and click OK, the MS Outlook COM Add-in Wizard starts. In the wizard windows, you choose the project options and create option pages for your add-in.





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

The add-in project includes the following items:



- The project source files (ProjectName.*);
- The type library files: binary (ProjectName.tlb) and Object Pascal unit (ProjectName_TLB.pas);
- The Outlook COM add-in module (ProjectName_IMPL.pas and ProjectName_IMPL.dfm) discussed in the following step.
- The Outlook Property Page (PropertPage_IMPL.pas and PropertPage_IMPL.dfm) discussed in the Step #10 – Adding Folder Property Pages.

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

The COM Add-in Module (MyOutlookAddin1_IMPL.pas and MyOutlookAddin1_IMPL.dfm) is the core part of the COM add-in project (see COM Add-ins). It is the container 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.

The code for MyAddin1_IMPL.pas is as follows:

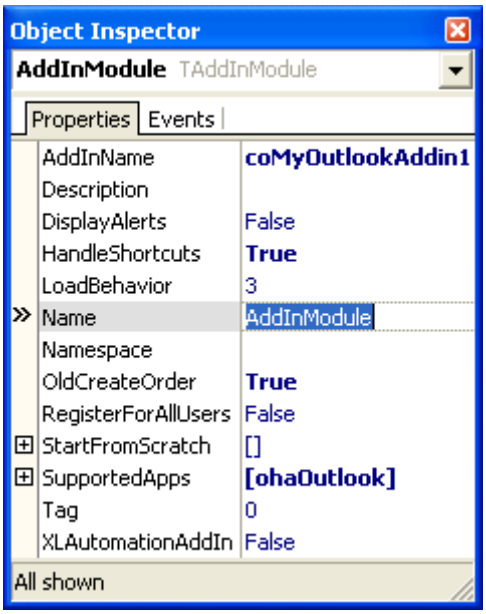
```
  unit MyOutlookAddin1_IMPL;

  interface

  uses
    SysUtils, ComObj, ComServ, ActiveX, Variants, adxAddIn,
MyOutlookAddin1_TLB, Outlook2000;

  type
    TcoMyOutlookAddin1 = class(TadxAddin, IcoMyOutlookAddin1)
    end;

    TAddInModule = class(TadxCOMAddInModule)
      procedure adxCOMAddInModuleAddInInitialize(Sender: TObject);
      procedure adxCOMAddInModuleAddInFinalize(Sender: TObject);
    private
    protected
      procedure NameSpaceOptionsPagesAdd(ASender: TObject;
        const Pages: PropertyPages; const Folder: MAPIFolder); override;
    public
    end;
```

```
  var
    adxcoMyOutlookAddin1: TAddInModule;

  implementation

  {$R *.dfm}

  procedure TAddInModule.adxCOMAddInModuleAddInInitialize(Sender: TObject);
  begin
    adxcoMyOutlookAddin1 := Self;
  end;

  procedure TAddInModule.adxCOMAddInModuleAddInFinalize(Sender: TObject);
  begin
    adxcoMyOutlookAddin1 := nil;
  end;

  procedure TAddInModule.NameSpaceOptionsPagesAdd(ASender: TObject;
    const Pages: PropertyPages; const Folder: MAPIFolder);

    function GetFullFolderName(const AFolder: MAPIFolder): string;
    var
      IDisp: IDispatch;
      Folder: MAPIFolder;
    begin
      Result := '';
      Folder := AFolder;
      while Assigned(Folder) do begin
        Result := '\' + Folder.Name + Result;
        IDisp := Folder.Parent;
        IDisp.QueryInterface(IID_MAPIFolder, Folder);
      end;
      if Result <> '' then Delete(Result, 1, 1);
    end;

  begin
    Pages.Add('MyOutlookAddin1.coPropertyPage1', 'My Property Page');
  end;

  initialization
    TadxFactory.Create(ComServer, TcoMyOutlookAddin1, CLASS_coMyOutlookAddin1,
TAddInModule);

  end.
```

The add-in module contains two classes: the “interfaced” class (TcoMyOutlookAddin1 in this case) and the add-in module class (TAddInModule). The “interfaced” class is a descendant of the TadxAddIn class that implements the IDTExtensibility2 interface required by the COM Add-in architecture. Usually, you don't need to change anything in the TadxAddIn class.

The add-in module class implements the add-in functionality. It is an analogue of the Data Module, but unlike the Data Module, the add-in module allows you to set all properties of your add-in, handle its events, and create toolbars and controls.

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



The designer of Outlook COM Add-in Module allows setting add-in properties and adding components to the module.

To add an Add-in Express component to the AddinModule designer, you select it in the Tool Palette and drag-n-drop onto the designer.

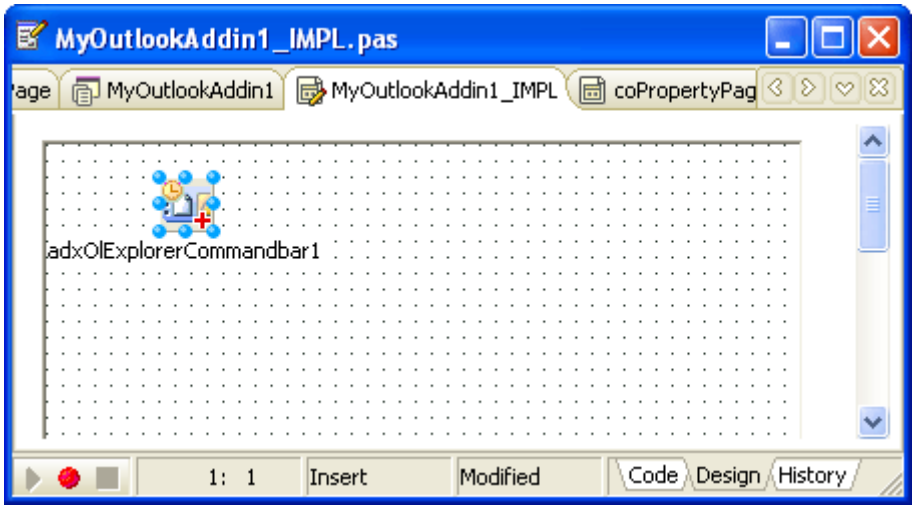
You can add the following Add-in Express components to the designer:

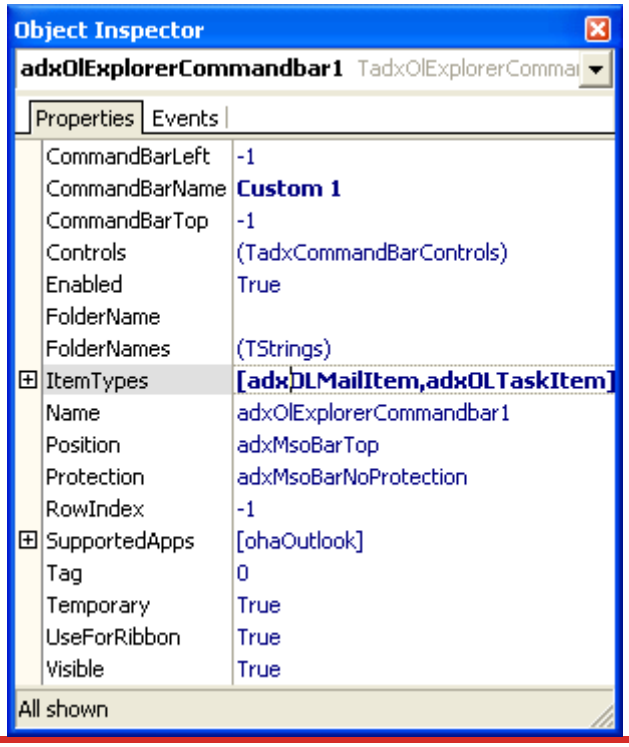
- TadxRibbonTab – represents a Ribbon tab in your add-in (see Office 2007 Ribbon Components)
- TadxRibbonQAT – represents the Ribbon Quick Access Toolbar your add-in (see Office 2007 Ribbon Components)
- TadxRibbonOfficeMenu – represents the Ribbon Office Menu in your add-in (see Office 2007 Ribbon Components)
- TadxCommandBar – represents a command bar in your add-in (see Command Bars)
- TadxOlExplorerCommandBar – represents an Outlook Explorer command bar in your add-in (see Command Bars)
- TadxOlInspectorCommandBar – represents an Outlook Inspector command bar in your add-in (see Command Bars)
- TadxBuiltInControl – allows intercepting the action of a built-in control of the host application(s) (see Built-in Control Connector)
- TadxKeyboardShortcut – allows intercepting application-level keyboard shortcuts (see Keyboard Shortcut)

- TadxOlBarShortcut – allows adding Outlook Bar shortcuts and shortcut groups (see Outlook Bar Shortcut Manager)
- TadxOlFormsManager – embedding custom VCL forms into Outlook windows (see Outlook Forms Manager)
- Application Events – adds or deletes components that provides 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 TadxOlExplorerCommandBar component from the Add-in Express group in the Tool Palette.



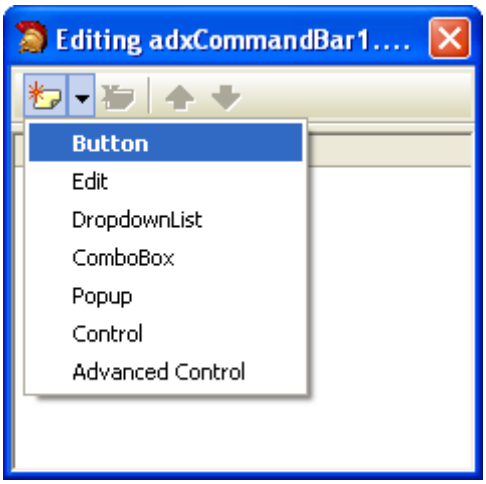


Select the Add-in Express Outlook Explorer Command Bar component, and, in the Object Inspector window, specify the command bar name using the CommandBarName property and choose its position (see the Position property). Outlook-specific versions of Add-in Express Command Bar component provides context-sensitive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

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 Object Inspector window, you select the Controls property and click the property editor button (the button in the property value field).

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

Specify the button's Caption property and set the Style property to adxMsoButtonIconAndCaption (default value = adxMsoButtonCaption). To handle the Click event of the button, In the Object Inspector window, switch to the Events tab and add the Click event handler: The code of the event handler follows below (it's empty as you can see)

```
procedure TAddInModule.adxCommandBar1Controls0Click(Sender: TObject);
begin

end;
```

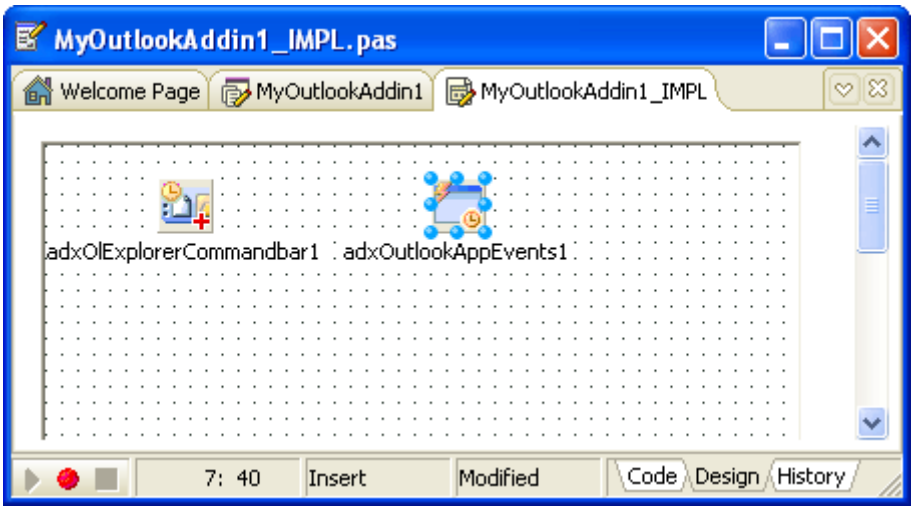
Step #6 - Accessing Outlook Objects

The Add-in Module provides the <HostName>App properties that return the Application object (of the OleVariant type) of the host application the add-in is currently running in. For your convenience, Add-in Express provides the OutlookApp property of the TOutlookApplication type. This allows you to write the following code to the Click event of the button just added.

```
procedure TAddInModule.adxOlExplorerCommandbar1Controls0Click(Sender:
TObject);
begin
  ShowMessage('The subject is ' +
    OleVariant(Self.OutlookApp.ActiveExplorer.Selection.Item(1)).Subject);
end;
```

Step #7 - Handling Outlook Events

Add-n Express provides several components that provide the application-level events for the COM Add-in Module (see Application-level Events). To add Outlook events to the add-in, find the TadxOutlookAppEvents component in the Tool Palette and drag-n-drop it onto the module.

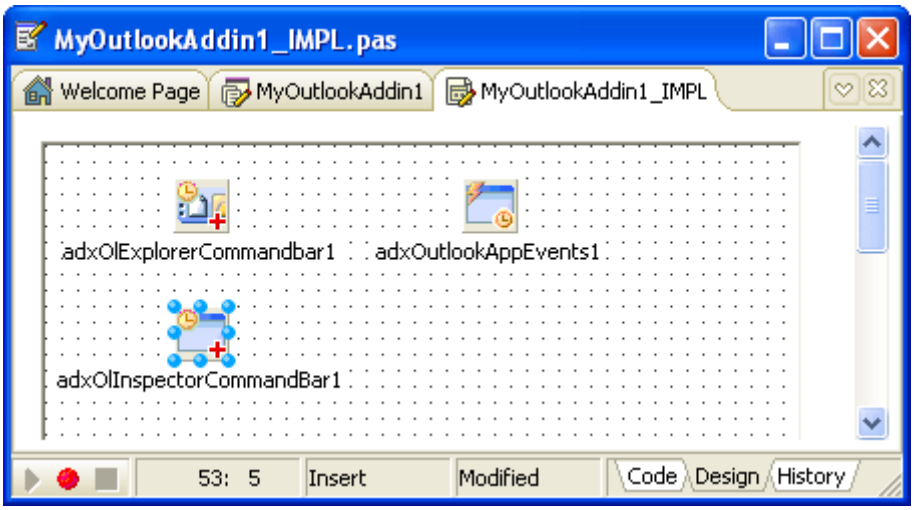


With the Outlook Events component, you handle any application-level events of the Outlook. For instance, the following code handles the BeforeFolderSwitch event of the Outlook Explorer class:

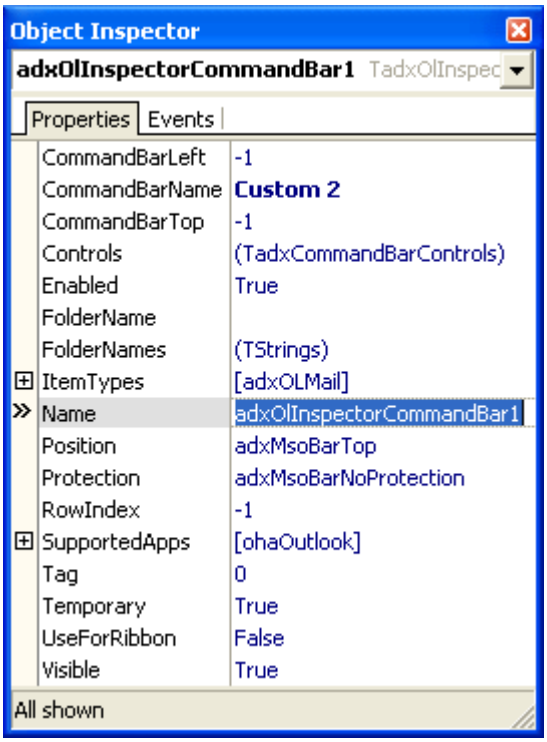
```
procedure TAddInModule.adxOutlookAppEvents1ExplorerBeforeFolderSwitch(
  ASender: TObject; const NewFolder: IDispatch; var Cancel: WordBool);
begin
    ShowMessage('You are switching to the '
      + (NewFolder as MAPIFolder).Name + ' folder');
end;
```

Step #8 – Adding a New Inspector Command Bar

To add a command bar to Outlook Explorer window, use the TadxOlExplorerCommandBar component from the Add-in Express group in the Tool Palette.



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:

```
   procedure TAddInModule.adxOlInspectorCommandBar1Controls0Click(Sender:
TObject);
   begin
     ShowMessage('The subject is ' +
       OleVariant(Self.OutlookApp.ActiveInspector.CurrentItem).Subject);
   end;
```

Outlook Inspector Command Bars in Outlook 2007

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

See also Command Bars and Outlook Command Bar Visibility Rules.

Step #9 – Handling Events of Outlook Items Object

The Outlook2000 unit provides the TItems component (of the TOleServer type). This component provides the following events: OnItemAdd, OnItemChange, and OnItemRemove. To process these events, you add the following declarations and code to the Add-in Module:

```
    TAddInModule = class(TadxCOMAddInModule)
  ...
    private
      Items: TItems;
      procedure ItemsAdd(ASender: TObject; const Item: IDispatch);
      function GetItemsConnected(): boolean;
    public
      procedure ConnectItems;
      procedure DisconnectItems;
      property AreItemsConnected: boolean read GetItemsConnected;
    end;
  ...
  procedure TAddInModule.adxCOMAddInModuleAddInStartupComplete(Sender:
TObject);
  begin
    ConnectItems;
  end;

  procedure TAddInModule.adxCOMAddInModuleAddInBeginShutdown(Sender: TObject);
  begin
      DisconnectItems;
  end;

  procedure TAddInModule.ConnectItems;
  begin
    Items := TItems.Create(nil);
    Items.OnItemAdd := ItemsAdd;
    Items.ConnectTo(
      Self.OutlookApp.GetNamespace('Mapi').
        GetDefaultFolder(olFolderInbox).Items);
  end;

  procedure TAddInModule.DisconnectItems;
  begin
    if not Assigned(Items) then exit;
    Items.Disconnect;
    Items.Free();
    Items := nil;
  end;

  function TAddInModule.GetItemsConnected: boolean;
  begin
    Result := false;
    if Assigned(Items) then
      Result := Items.DefaultInterface <> nil;
  end;
```

```
procedure TAddInModule.ItemsAdd(ASender: TObject; const Item: IDispatch);
begin
    ShowMessage('The item with subject "'
      + OleVariant(Item).Subject
      + '" has been added to the Inbox folder');
end;
```

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, the Add-in Express wizard provides you with the Option Pages window (see Step #1 – Creating an Add-in Express Outlook COM Add-in Project).

By default, an option page contains two controls only: a label and an edit box.

The edit box shows how to handle the OnChange event and update the corresponding option page.

```
procedure TcoPropertyPage1.Edit1Change(Sender: TObject);
begin
  GetPropertyPageSite;
  // TODO - put your code here
  UpdatePropertyPageSite;
end;
```

You add the TCheckBox component to the Property page, handle its OnClick event following the code template above, and connect or disconnect the TItems component in the Apply method. You initialize the check box using the OnCreate event of the property page:

```
procedure TcoPropertyPage1.ActiveFormCreate(Sender: TObject);
begin
  CheckBox1.Checked := adxcoMyOutlookAddin1.AreItemsConnected;
end;

function TcoPropertyPage1.Apply: HResult;
begin
  if CheckBox1.Checked then
    adxcoMyOutlookAddin1.ConnectItems
  else
    adxcoMyOutlookAddin1.DisconnectItems;
  FDirty := False;
  Result := S_OK;
end;
```

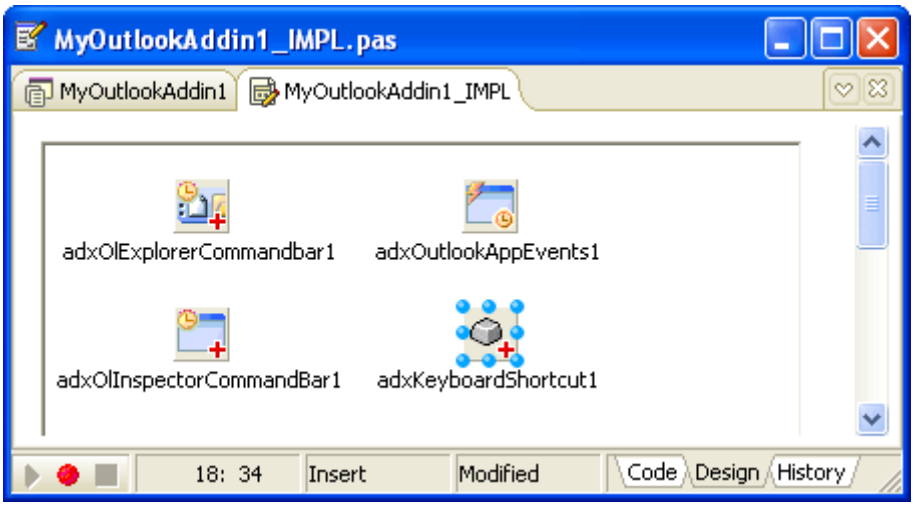
Finally, you change the NameSpaceOptionsPagesAdd event handler in the AddinModule code as follows:

```
    ...
procedure TAddInModule.NameSpaceOptionsPagesAdd(ASender: TObject;
  const Pages: PropertyPages; const Folder: MAPIFolder);
...
begin
  if Folder.EntryID =
    Self.OutlookApp.GetNamespace('Mapi').
    GetDefaultFolder(olFolderInbox).EntryID
  then
    Pages.Add('MyOutlookAddin1.coPropertyPage1', 'My Property Page');
end;
```

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 Object Inspector 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

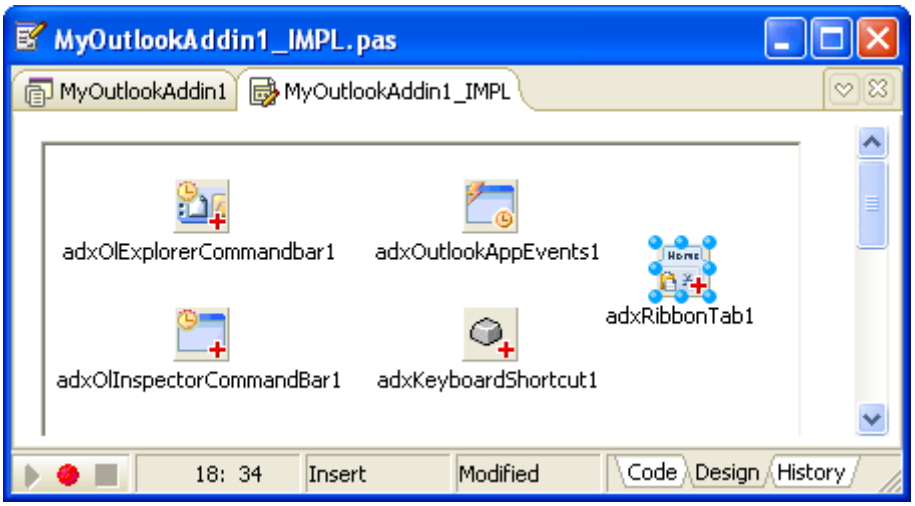
To use keyboard shortcuts, you need to set the HandleShortcuts property of AddinModule to true.

Now you handle the Action event of the component:

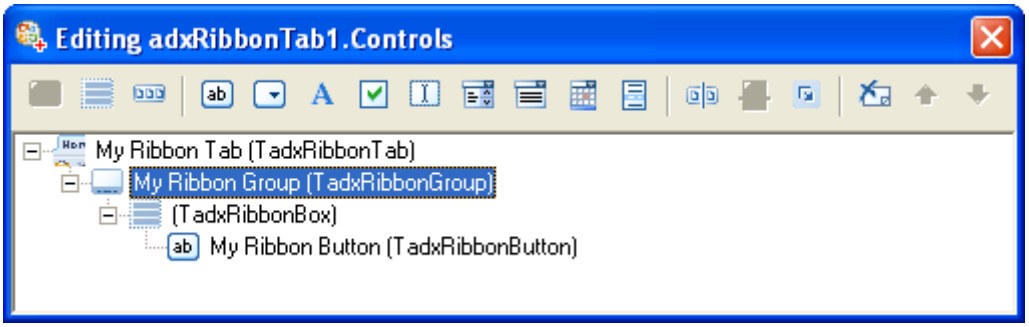
```
procedure TAddInModule.adxKeyboardShortcut1Action(Sender: TObject);
begin
  ShowMessage('You`ve pressed ' +
    TadxKeyboardShortcut(Sender).ShortcutText);
end;
```

Step #12 – Customizing the Outlook 2007 Ribbon User Interface

To add a new tab to the Ribbon, you add the TadxRibbonTab component to the module.



In the Object Inspector 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.



Now add the event handler to the Click event of the button. Write the following code:

```
   procedure TAddInModule.adxRibbonTab1Controls0Controls0Controls0Click(Sender:
TObject; const RibbonControl: IRibbonControl);
   begin
     adxOlInspectorCommandBar1Controls0Click(nil);
   end;
```

Remember, the TadxRibbonTab Controls editor perform 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.

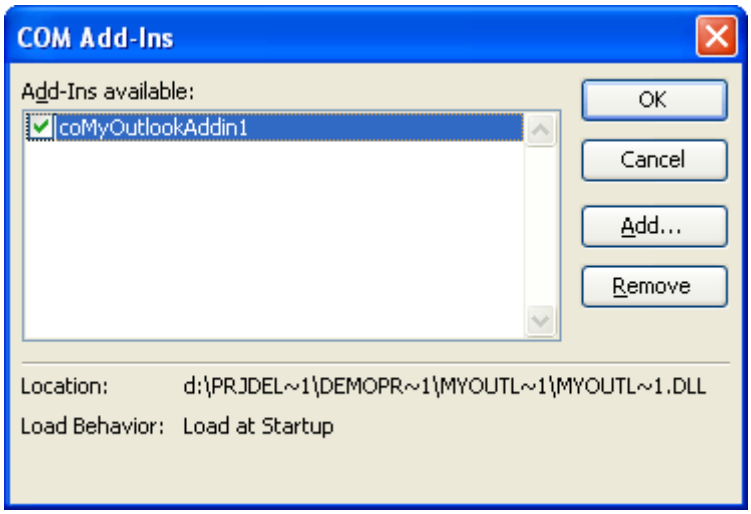
Note, unlike other Ribbon-based applications, Outlook has numerous ribbons. Please use the Ribbons property of your TadxRibbonTab components to specify the ribbons you customize with your tabs.

See also Office 2007 Ribbon Components.

Step #13 - Running the COM Add-in

Choose the Register ActiveX Server item in the Run menu, then restart Outlook and find your option page(s), command bars, and controls.

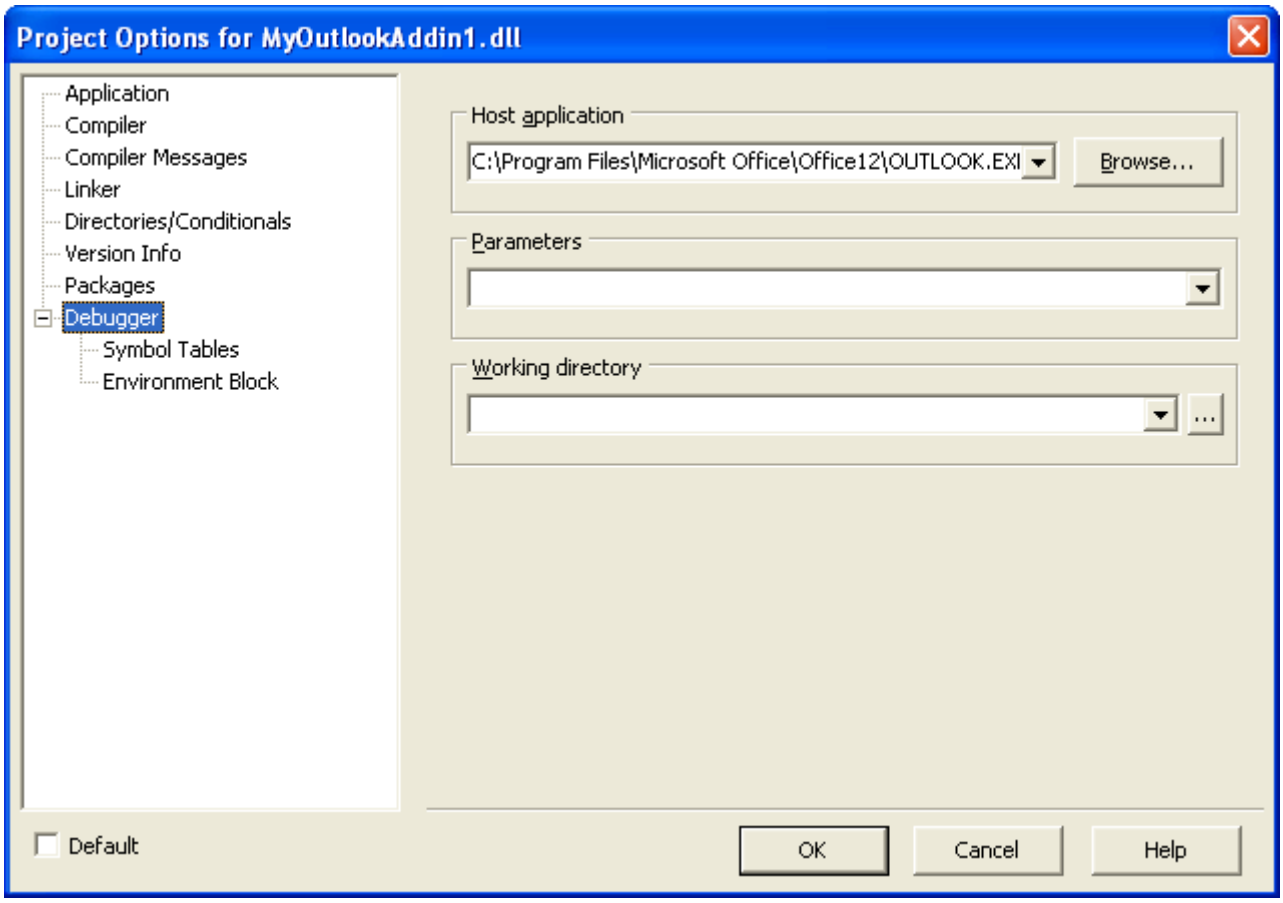
You find your add-in in the COM Add-ins dialog:



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 in the Host Application field in the Project Options window.



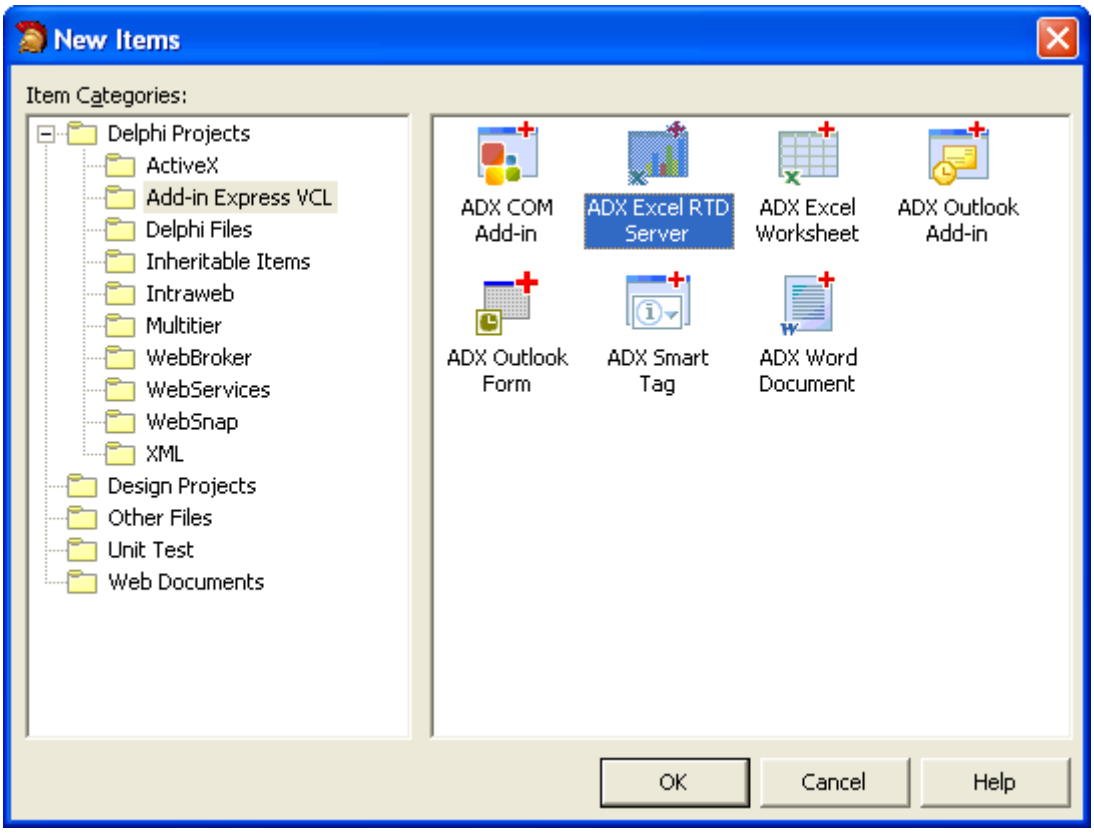
Step #15 – Deploying the COM Add-in

Make sure your setup project registers the add-in DLL. Say, in Inno Setup projects you use the 'regserver' command.

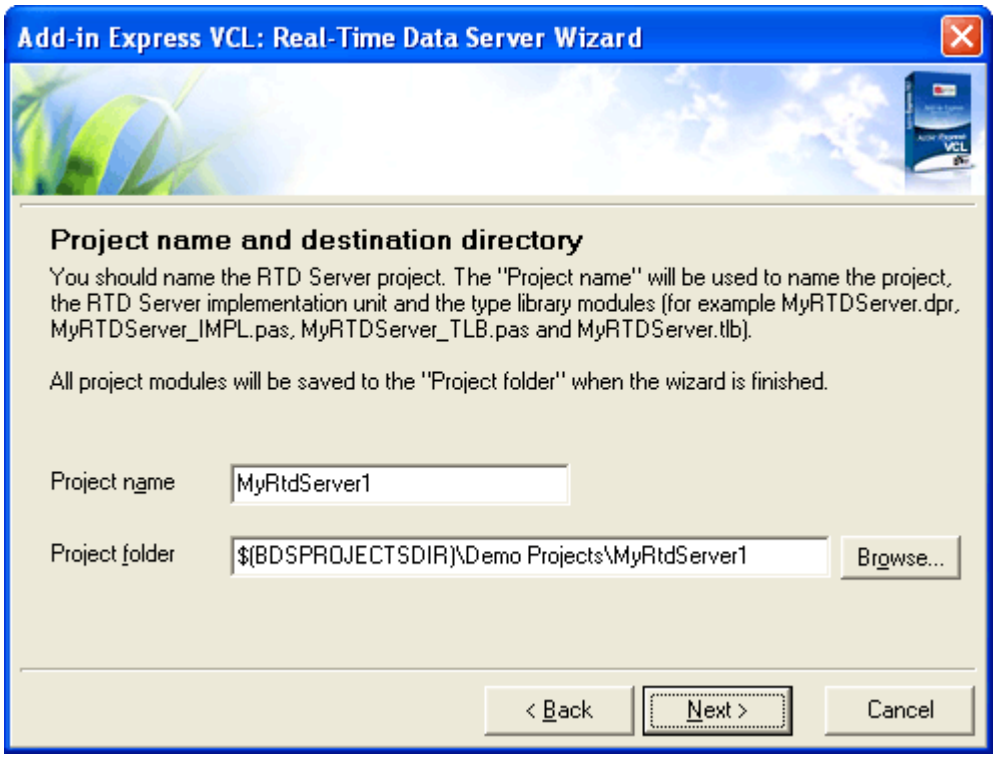
Your First Excel RTD Server

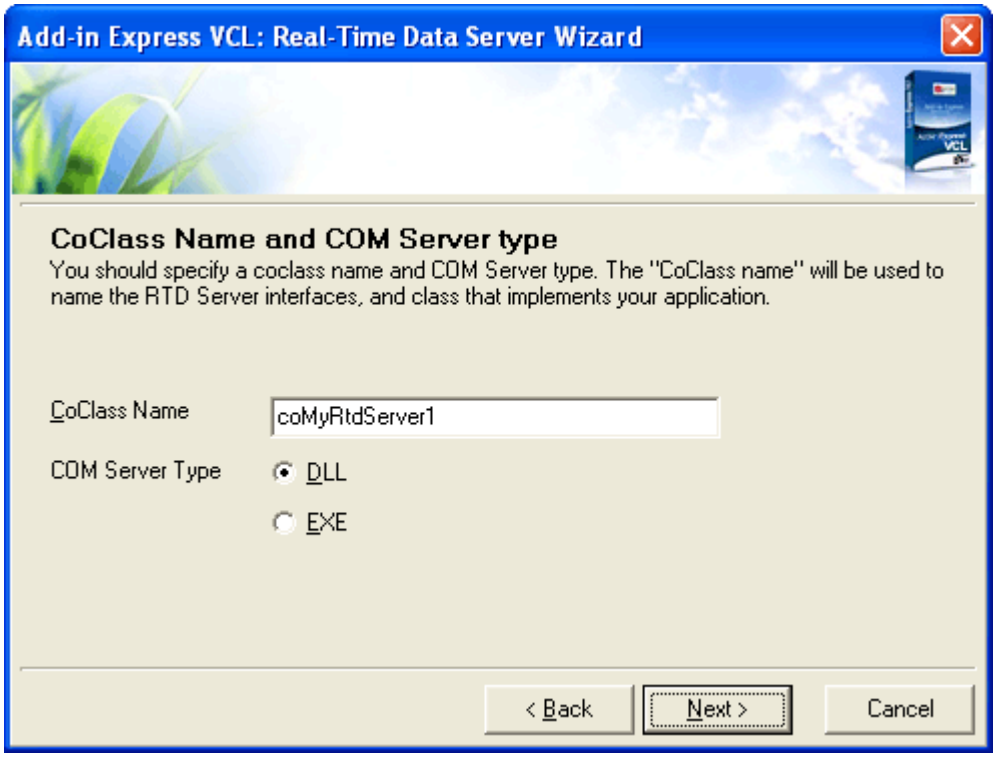
Step #1 – Creating a New Add-in Express RTD Server Project

Add-in Express adds the Add-in Express RTD Server project template to the New Item dialog of Borland Delphi for Microsoft Windows.



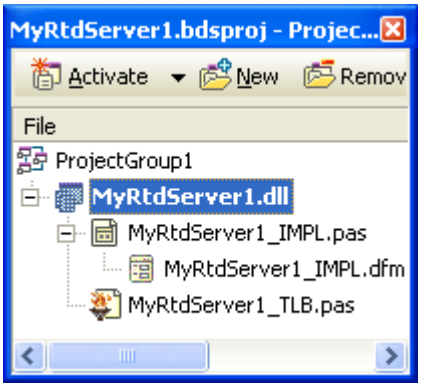
When you select the template and click OK, the RTD Server Project wizard starts. In the wizard windows, you choose the project options.





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

The RTD server project includes the following items:



- The project source files (ProjectName.*);
- The type library files: binary (ProjectName.tlb) and Object Pascal unit (ProjectName_TLB.pas);
- The RTD server module (ProjectName_IMPL.pas and ProjectName_IMPL.dfm) discussed in the following step.

Step #2 – Add-in Express RTD Server Module

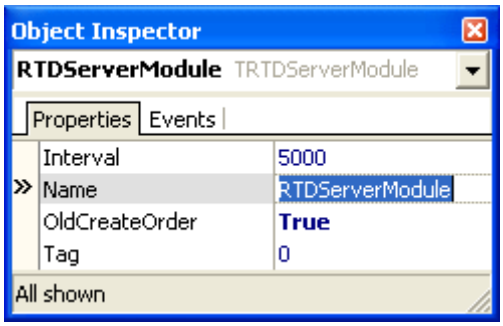
The RTD server module (MyRtdServer1_IMPL.pas and MyRtdServer1_IMPL.dfm) is the core part of the RTD Server project. The module is a container for TadxRTDTopic components.

The code of MyRtdServer1_IMPL.pas is as follows:

```
unit MyRtdServer1_IMPL;

interface

uses
  SysUtils, Classes, ComServ, MyRtdServer1_TLB, adxRTDServ;

type
  TcoMyRtdServer1 = class(TadxRTDServer, IcoMyRtdServer1);

  TRTDServerModule = class(TadxXLRTDServerModule)
  private
  protected
  public
  end;

implementation

{$R *.dfm}

initialization
  TadxRTDFactory.Create(ComServer, TcoMyRtdServer1, CLASS_coMyRtdServer1,
TRTDServerModule);

end.
```

Step #3 – Add-in Express RTD Server Designer

The module designer allows setting RTD Server properties and adding components to the module.

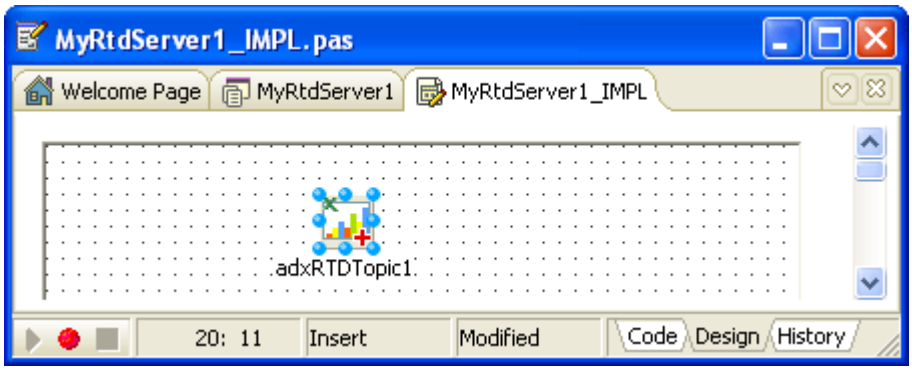


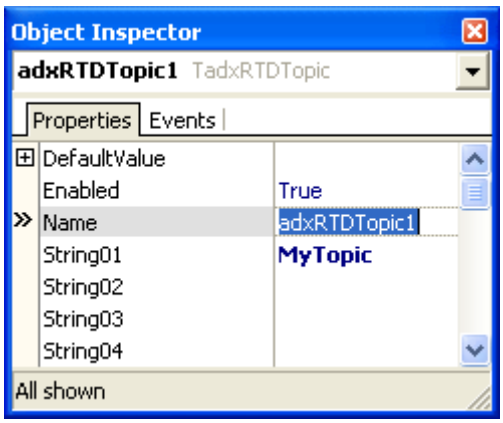
You set the properties of your RTD server module in the Object Inspector window (see RTD Servers).

The only Add-in Express component available for the module is the TadxRTDTopic component (see RTD Topic).

Step #4 – Adding and Handling a New Topic

To add a new topic to your RTD Server, in the Tool Palette, find the TadxRTDTopic component and drag-n-drop it onto the RTD Server Module (see RTD Topic).





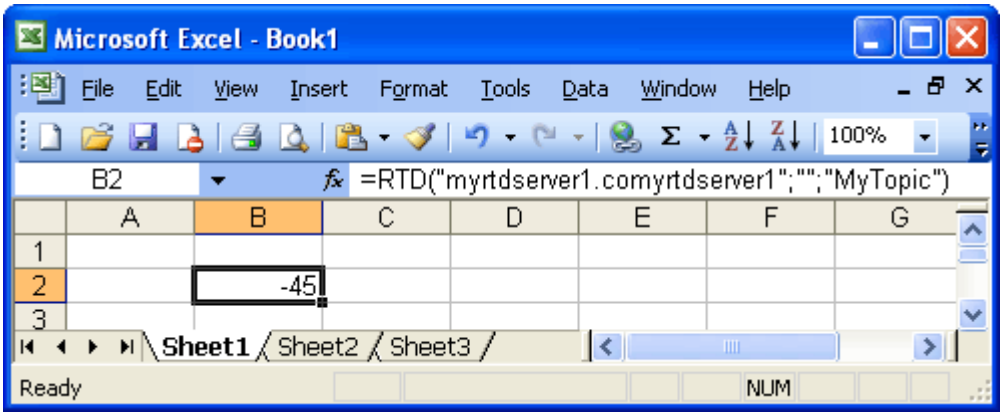
Select the newly added component and, in the Object Inspector window, specify the topic using the String## properties.

Write your code to handle the RefreshData event of the RTD Topic component:

```
   function TRTDServerModule.adxRTDTopic1RefreshData(Sender: TObject):
OleVariant;
   begin
     Result := RandomRange(-100, 100);
   end;
```

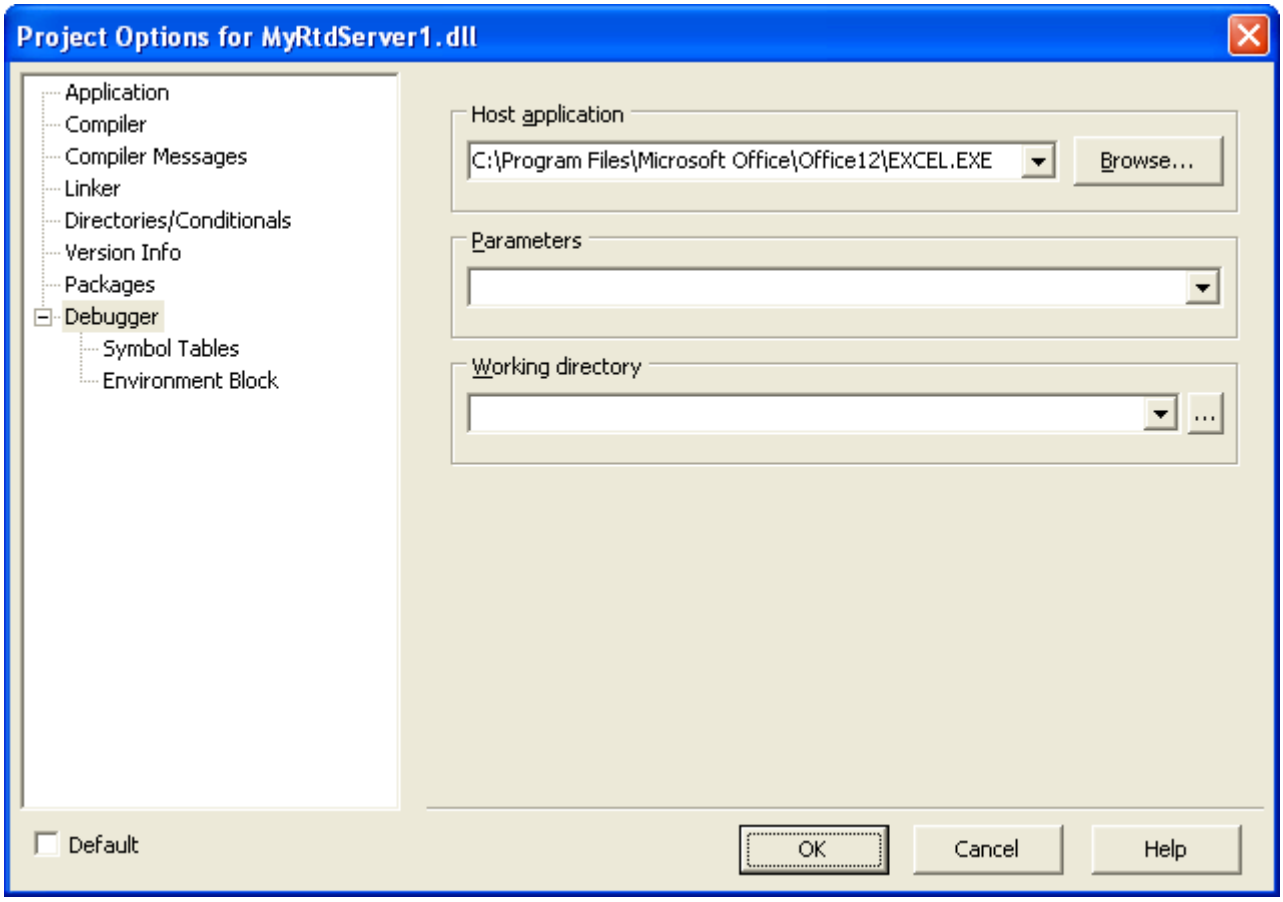
Step #5 - Running the RTD Server

Choose the Register ActiveX Server item in the Run menu, restart Excel, and enter the RTD function to a cell.



Step #6 - Debugging the RTD Server

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



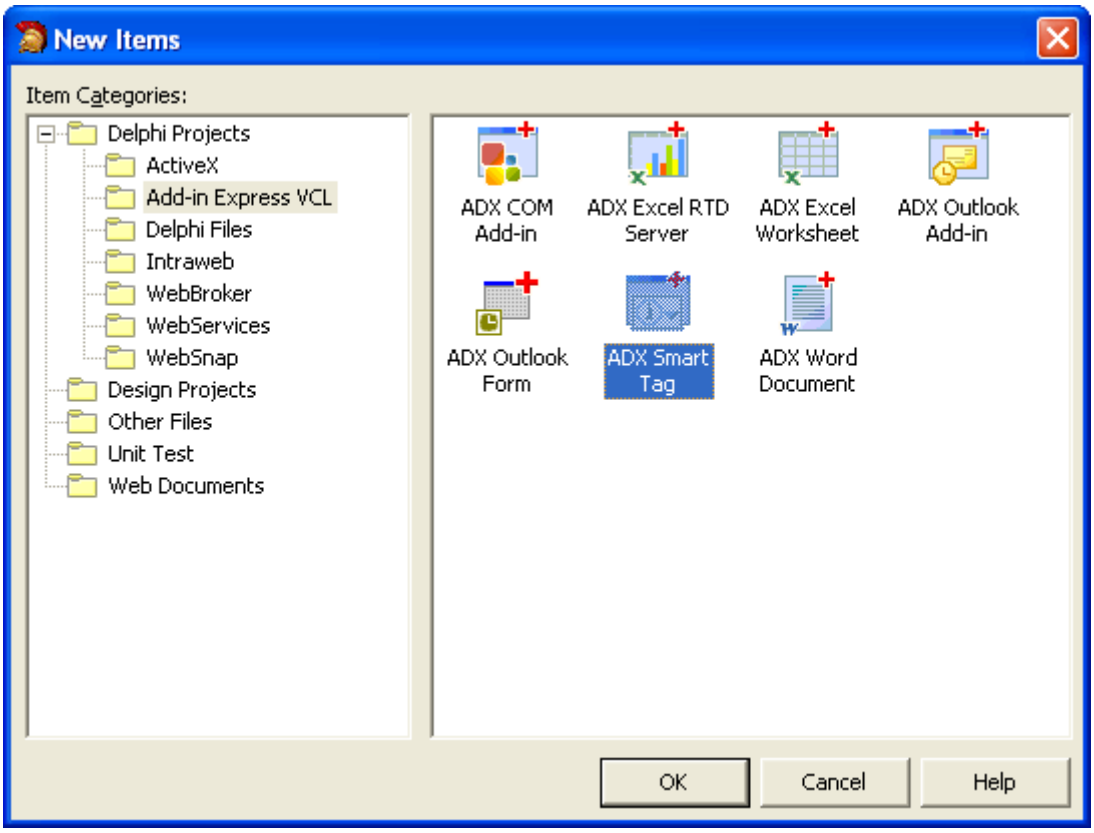
Step #7 – Deploying the RTD Server

Make sure your setup project registers the RTD Server DLL (or EXE). Say, in Inno Setup projects you use the 'regserver' command.

Your First Smart Tag

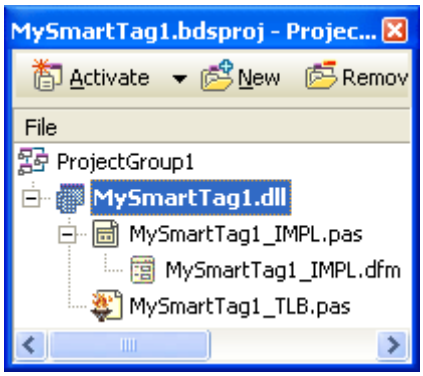
Step #1 – Creating a New Smart Tag Library Project

Add-in Express adds the Add-in Express COM Add-in project template to the New Item dialog of Borland Delphi for Microsoft Windows.



When you select the template and click OK, the Smart Tag Project wizard starts. In the wizard windows, you choose the project options.

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



The smart tag project includes the following items:

- The project source files (ProjectName.*);
- The type library files: binary (ProjectName.tlb) and Object Pascal unit (ProjectName_TLB.pas);
- The smart tag module (ProjectName_IMPL.pas and ProjectName_IMPL.dfm) discussed in the following step.

Step #2 – Add-in Express Smart Tag Module

The smart tag module (MySmartTag1_IMPL.pas and MySmartTag1_IMPL.dfm) is the core part of the smart tag project. The smart tag module is a container for TadxSmartTag components.

The code for MySmartTag1_IMPL.pas is as follows:

```
unit MySmartTag1_IMPL;

interface

uses
```

```
    SysUtils, ComObj, ComServ, ActiveX, Variants, adxSmartTag, adxSmartTagTLB,
MySmartTag1_TLB;

  type
    TcoMySmartTag1Recognizer = class(TadxRecognizerObject,
IcoMySmartTag1Recognizer)
    protected
    end;

    TcoMySmartTag1Action = class(TadxActionObject, IcoMySmartTag1Action)
    protected
    end;

    TSmartTagModule = class(TadxSmartTagModule)
    private
    protected
    public
    end;

  implementation

  {$R *.dfm}

  initialization
    TadxRecognizerFactory.Create(ComServer, TcoMySmartTag1Recognizer,
      CLASS_coMySmartTag1Recognizer, TSmartTagModule);

    TadxActionFactory.Create(ComServer, TcoMySmartTag1Action,
      CLASS_coMySmartTag1Action, TSmartTagModule);

  end.
```

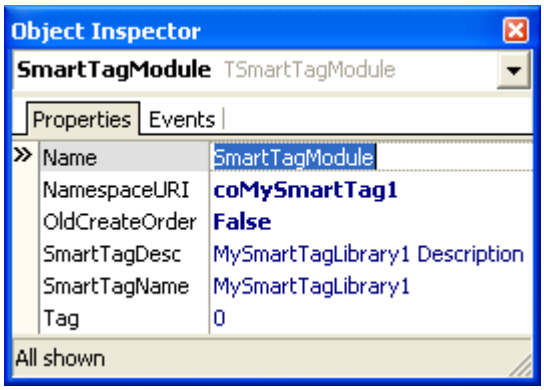
The smart tag module contains three classes:

- The “interfaced” classes (TcoMySmartTag1Recognizer and TMySmartTag1Action);
- The smart tag module class (TSmartTagModule).

The “interfaced” classes are descendants of the TadxRecognizerObject class and the TadxActionObject class that implement the smart tag specific interfaces required by the smart tag architecture: ISmartTagRecognizer, ISmartTagRecognizer2, ISmartTagAction and ISmartTagAction2. Usually, you don't need to change anything in theses classes.

In the smart tag module class, we write the functionality to be implemented by the smart tag. The smart tag module is an analogue of the Data Module, but unlike the Data Module, the smart tag module allows you to set all properties of your smart tags.

Step #3 – Add-in Express Smart Tag Designer



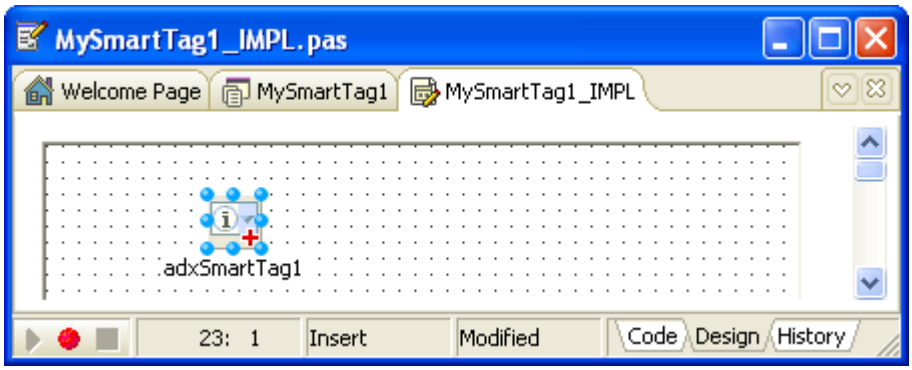
In the Project Manager window, select the smart tag module, activate the Object Inspector window, specify your smart tag name in the SmartTagName property (this name appears in the Smart Tags tab on the host application AutoCorrect Options dialog box), and enter the description of the smart tag through the SmartTagDesc property. These properties depend on Locale.

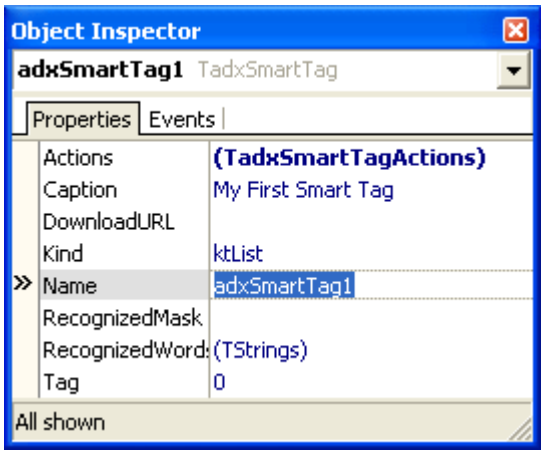
The designer of Smart Tag Module allows setting smart tag library properties and adding TadxSmartTag components to the module.

See also Smart Tags.

Step #4 – Adding a New Smart Tag

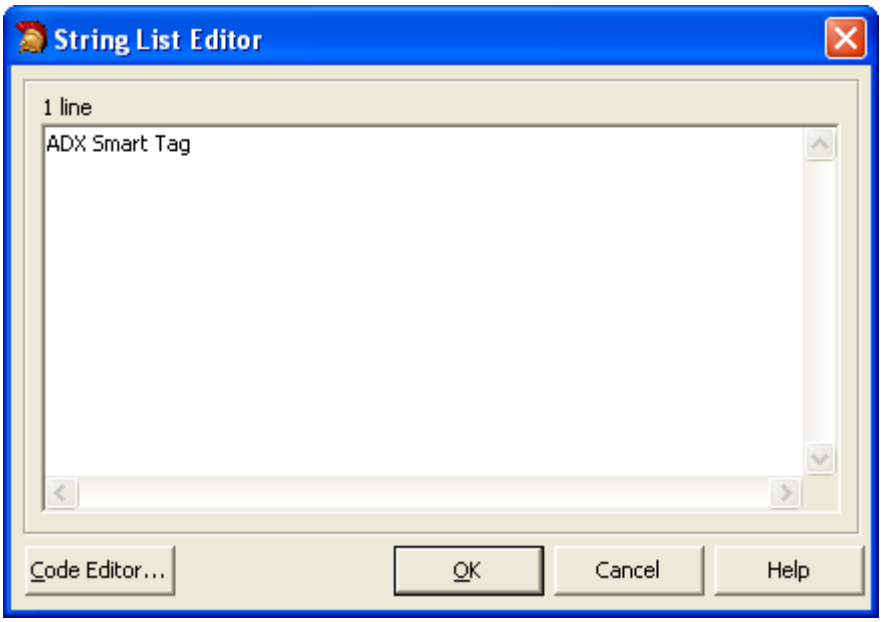
To add a new Smart Tag to your library, in the Tool Palette, find the TadxSmartTag component and drag-n-drop it onto the Smart Tag Module (see Smart Tag).



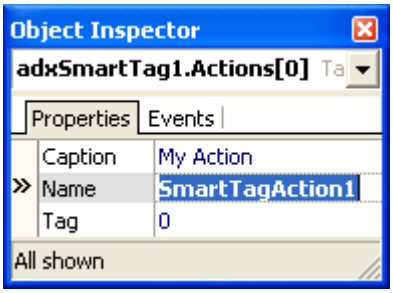


In the Object Inspector window, specify the caption for the added smart tag in the Caption property. The value of this property will become a caption of the smart tag context menu. Also, specify the phrase(s) recognizable by the smart tag in the RecognizedWords string collection.

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



Step #5 - Adding and Handling Smart Tag Actions



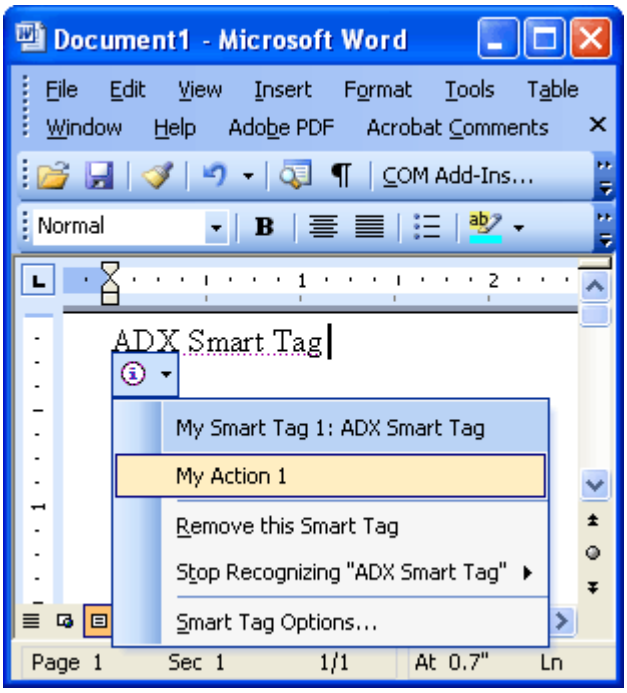
To add a new smart tag action, right-click the smart tag component, select the Smart Tag Actions item on the pop-up menu, and, in the Editing window, click the Add New button. In the Editing window, select the action, activate the Object Inspector window, and fill in the Caption property. It depends on Locale. The value of the Caption property is shown as an item of the smart tag context menu (pop-up).

To handle the click event of this menu item, select the Events tab of the Object Inspector, double click the OnClick event, and enter your code:

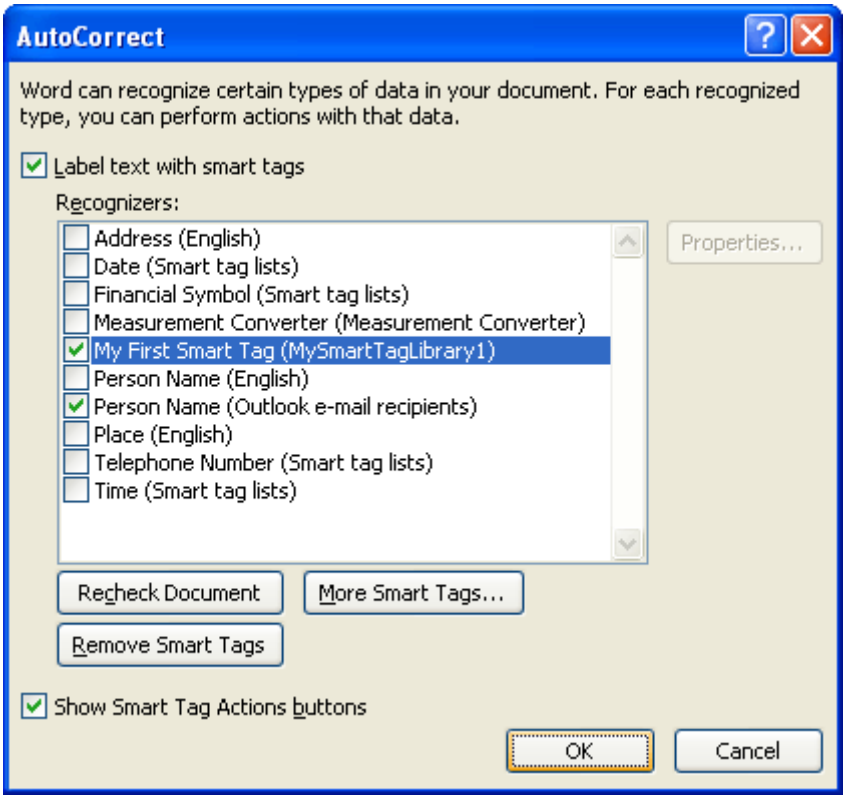
```
procedure TSmartTagModule.adxSmartTag1Actions0Click(Sender: TObject;
  const AppName: WideString; const Target: IDispatch; const Text,
  Xml: WideString; LocaleID: Integer);
begin
  ShowMessage('Recognized text is ' + Text);
end;
```

Step #6 - Running Your Smart Tag

Choose the Register ActiveX Server item in the Run menu, restart Word, put the words recognizable by your smart tag into a document, and see if the smart tag works.

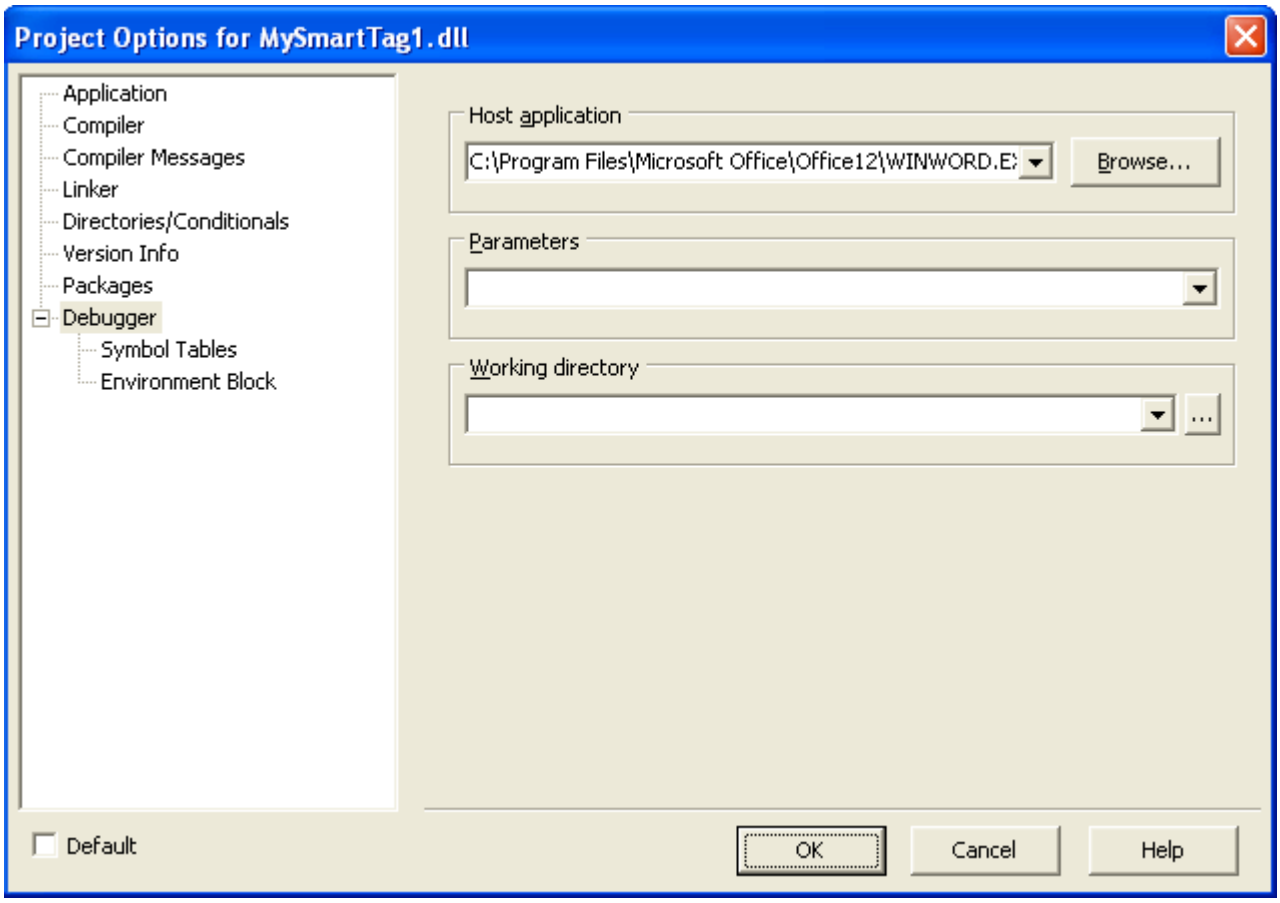


In Office 2003-2003, choose the Tools | AutoCorrect menu item and find your smart tag on the Smart Tags tab. In Office2007, the path to this dialog is as follows: Office button | Word Options | Add-ins | Mange | Manage Smart Tags | Go.



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.



Step #8 – Deploying the Smart Tag

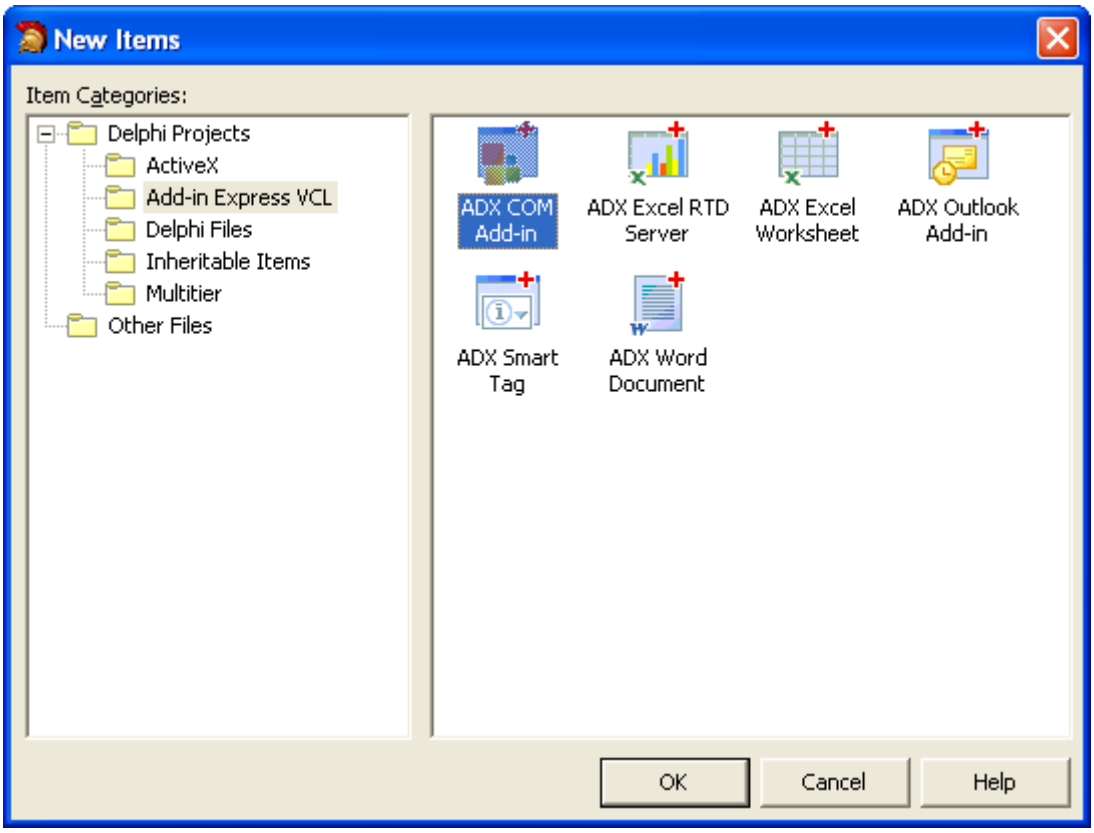
Make sure your setup project registers the smart tag DLL. Say, in Inno Setup projects you use the 'regserver' command.

Your First Excel Automation Add-in

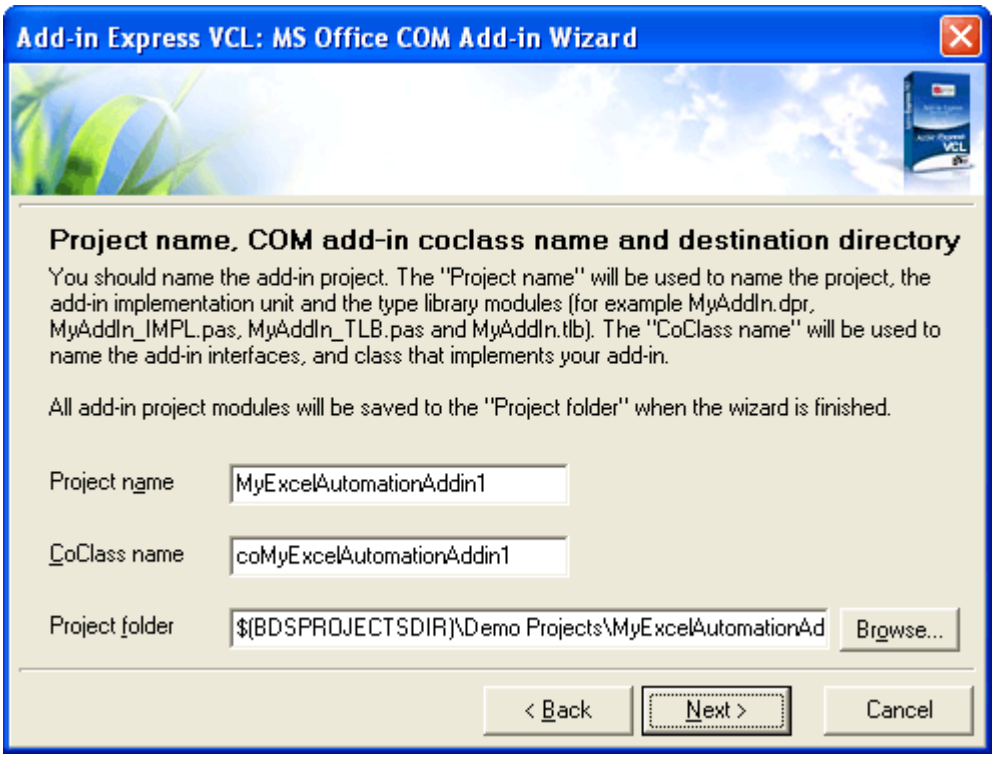
The fact is that Excel Automation Add-ins do not differ from COM Add-ins except for the registry entries. That's why Add-n Express bases Excel Automation Add-in projects on Add-in Express COM Add-in projects.

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

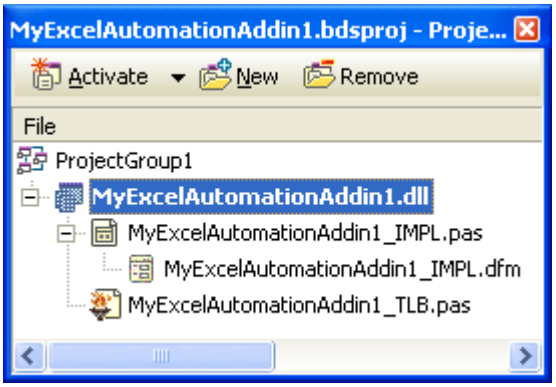
Add-in Express adds the Add-in Express COM Add-in project template to the New Item dialog of Borland Delphi for Microsoft Windows.



When you select the template and click OK, the MS Office COM Add-in Wizard starts. In the wizard windows, you choose the project options.



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

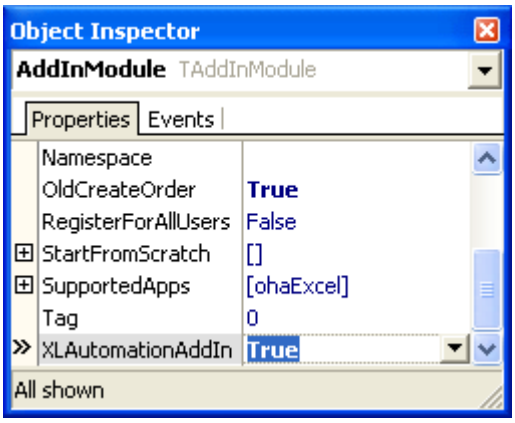


The add-in project includes the following items:

- The project source files (ProjectName.*);
- The type library files: binary (ProjectName.tlb) and Object Pascal unit (ProjectName_TLB.pas);
- The COM add-in module (ProjectName_IMPL.pas and ProjectName_IMPL.dfm) discussed in Your First Microsoft Office COM Add-in.

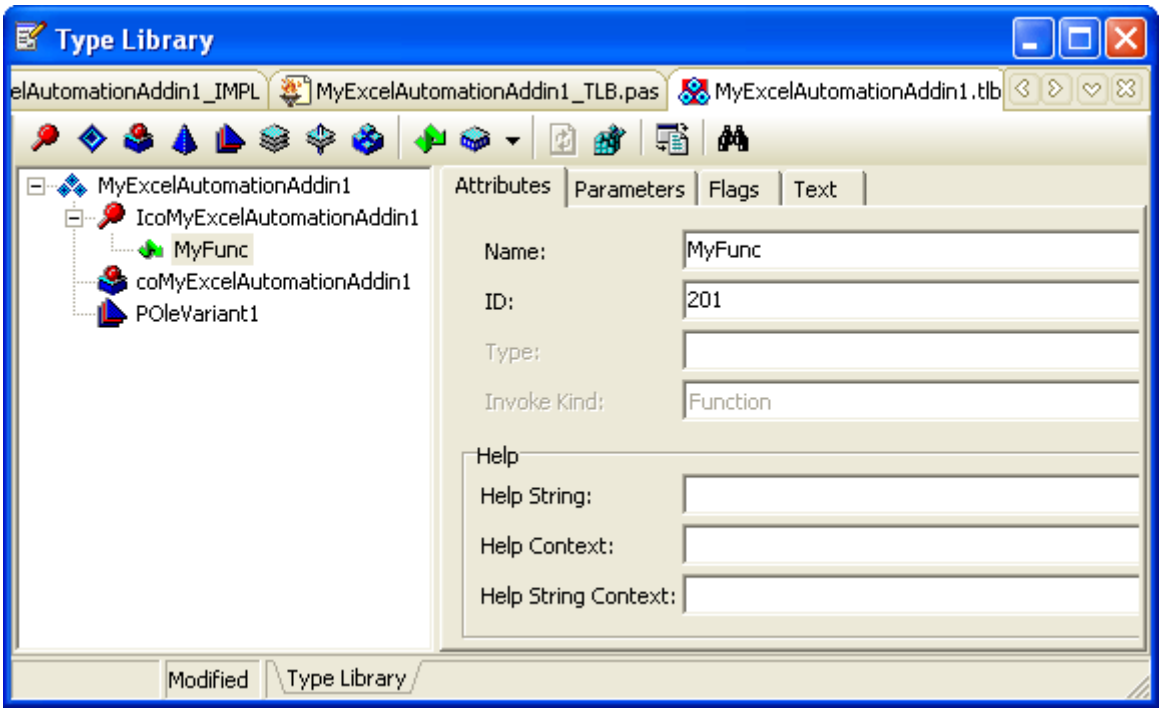
Step #2 – Involving Excel Automation Add-in Functionality

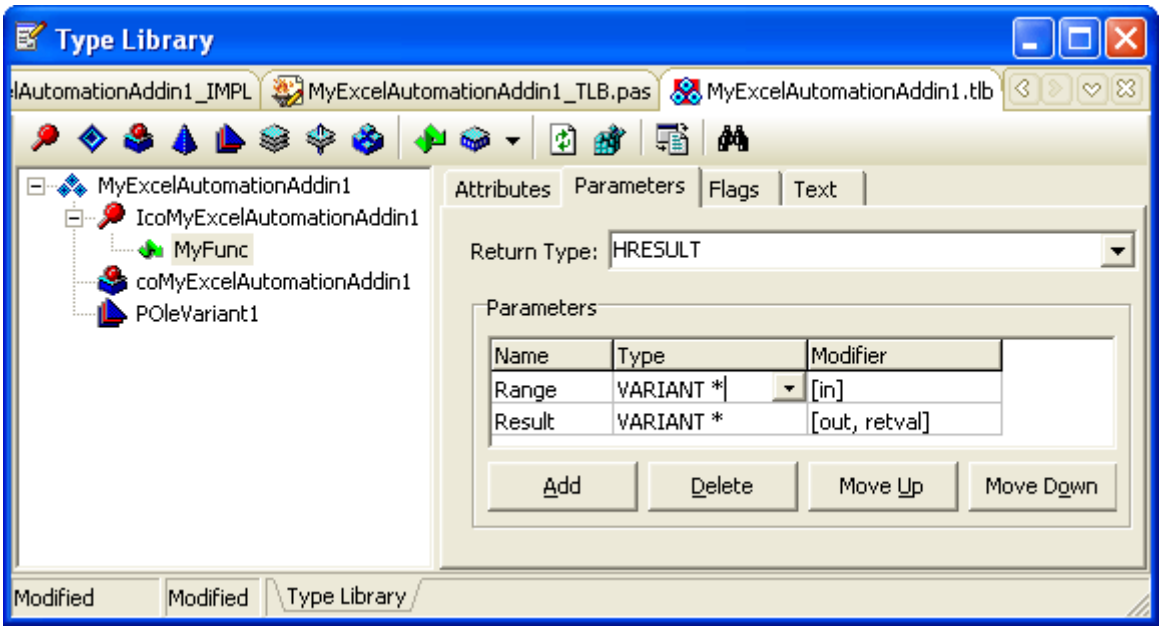
Before you start adding Excel user-defined functions to the COM Add-in, you set the XLAutomationAddin property of the add-in module to true.



Step #3- Creating User-Defined Functions

Open the project type library (View | Type Library menu). Add a new method to the type library and define its parameters.



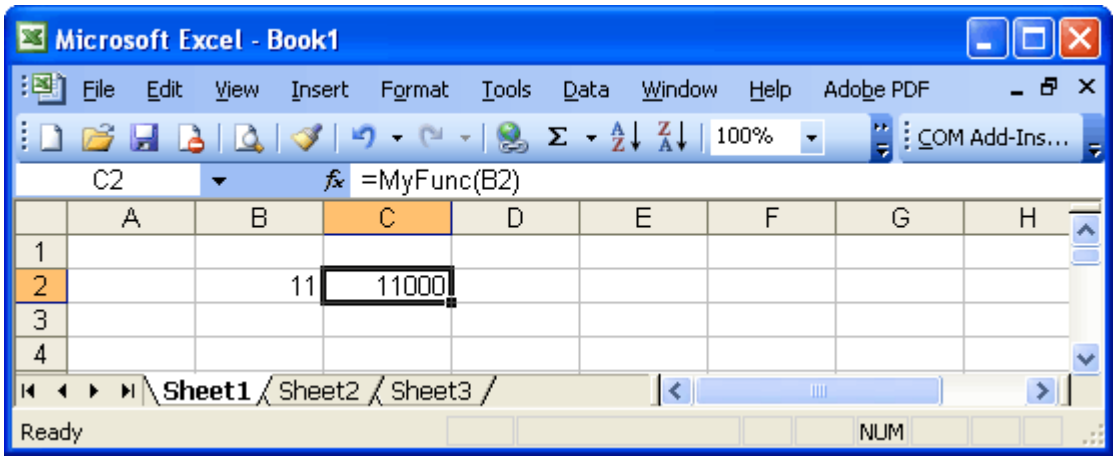


Click the Refresh button and write your code to the TcoMyExcelAutomationAddin1.MyFunc function:

```
   function TcoMyExcelAutomationAddin1.MyFunc(var Range: OleVariant):
OleVariant;
   begin
     Result := 0;
     case VarType(Range) of
       varSmallint, varInteger, varSingle,
       varDouble, varCurrency, varShortInt, varByte,
       varWord, varLongWord, varInt64: Result := Range * 1000;
     else
       try
         Result := Range.Cells[1, 1].Value * 1000;
       except
         Result := CVErr(xlErrValue);
       end;
     end;
   end;
```

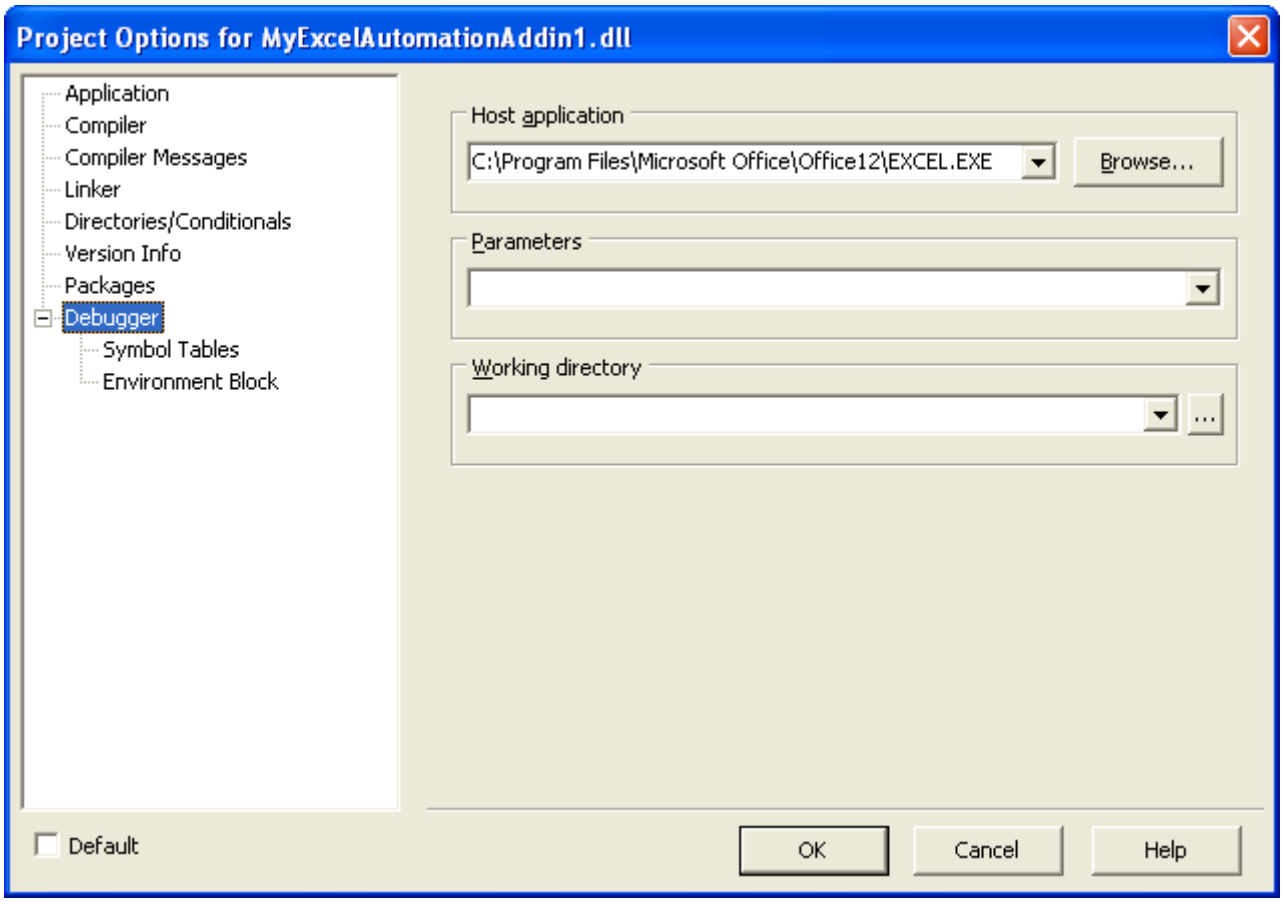
Step #4 – Running the Excel Automation Add-in

Choose the Register ActiveX Server item in the Run menu, restart Excel, and check if your add-in works.



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.



Step #6 – Deploying the Excel Automation Add-in

Make sure your setup project registers the add-in DLL. Say, in Inno Setup projects you use the 'regserver' command.

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.

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 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

In AddinModule, Add-in Express wizards generate the <HostName>App properties. They return the Application object (of the OleVariant type) of the host application the add-in is currently running in. To identify the host application, you can also use the HostType property of the module.

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 Add-in Express command bar 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 Add-in Express Command Bar instance and specify the name of the built-in command bar you need via the CommandBarName property.

In addition, 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 Command Bar Visibility Rules

You can use the FolderName(s) and ItemTypes properties to bind your toolbars to certain Outlook folders. Your toolbar is shown for a folder:

- If its full name (includes the folder path) is found in the FolderName or FolderNames properties.
- Or, if the folder type is found in the ItemTypes property.

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.

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).

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.

Sequence of Events When a Task Pane Shows

- AddinModule.OnTaskPaneBeforeCreate
- AddinModule.OnTaskPaneAfterCreate

- AddinModule.OnTaskPaneBeforeShow
- TaskPane.OnVisibleStateChange
- AddinModule.OnTaskPaneAfterShow

Adding a Task Pane to an Existing Add-in Express Project

1. Add an instance of ActiveForm to the project (File | New | Other | ActiveX | Active Form)
2. Change its AxBorderStyle property to afbNone.
3. Add the following declaration to the private section of the ActiveForm

```
  procedure WMMouseActivate(var Message: TWMMouseActivate); message
WM_MOUSEACTIVATE;
```

4. Change the method code as follows:

```
  var
    FocusedWindow: HWND;
    CursorPos: TPoint;
  begin
    inherited;
    FocusedWindow := Windows.GetFocus;
    if not SearchForHWND(Self, FocusedWindow) then begin
      Windows.GetCursorPos(CursorPos);
      FocusedWindow := WindowFromPoint(CursorPos);
      Windows.SetFocus(FocusedWindow);
      Message.Result := MA_ACTIVATE;
    end;
```

5. Add the following function used by the WMMouseActivate method (place it before the method):

```
  function SearchForHWND(const AControl: TWinControl; Focused: HWND): boolean;
  var
    i: Integer;
  begin
    Result := (AControl.Handle = Focused);
    if not Result then
      for i := 0 to AControl.ControlCount - 1 do
        if AControl.Controls[i] is TWinControl then begin
          if TWinControl(AControl.Controls[i]).Handle = Focused then begin
            Result := True;
            Break;
          end
          else
            if TWinControl(AControl.Controls[i]).ControlCount > 0 then begin
```

```
                Result := SearchForHWND(TWinControl(AControl.Controls[i]),
Focused);
                if Result then Break;
              end;
          end;
    end;
```

6. Add and override the ActiveForm destructor using the following code:

```
    destructor TMyTaskPane.Destroy;
    var
      ParkingHandle: HWND;
    begin
      ParkingHandle := FindWindowEx(0, 0, 'DAXParkingWindow', nil);
      if ParkingHandle <> 0 then
        SendMessage(ParkingHandle, WM_CLOSE, 0, 0);
      inherited Destroy;
    end;
```

7. Now you add an item to the TaskPanes collection of AddinModule and set its ControlProgID property to the ProgID of the Active Form – just select it in the dropdown list.
8. Don't forget about the Title property – the host application generates an exception if this property is left empty.

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.