Official Guide

ADX Extensions™
for Microsoft® Outlook®

- Introduction ■
- Getting Started ■
- Advanced Understanding ■
- Class Reference ■

Document version **1.1**

Revised at **7-Apr-06**

Product version **1.x**

Content

Introduction

Microsoft gives Office developers a way to extend and improve its Office applications using some special technologies. Office 2000 started IDTExensibility2 for creating COM add-ins. With Office 2002(XP) Microsoft published the Smart Tag technology to provide context sensitivity for its applications and the Excel RTD Server technology to replace the archaic DDE by a modern solution. Office 2003 enhanced applications' object models and supports these technologies by other applications. As a result, developers have several powerful and effective approaches to embed their applied code into Microsoft Office applications. However, Office developers want to get much more.

This document describes the ADX Extensions™ for Outlook and what it provides to extent the technologies supported by Microsoft.

Welcome to the ADX Extensions for Outlook

The ADX Extensions™ for Outlook (or **ADX X for Outlook**) is a plug-in for Add-in Express™ designed to provide unique features for creating commercial class UI for your Microsoft Outlook add-ins. Using ADX X for Outlook you can easily create sophisticated user interfaces comparable with today's most recognizable commercial Outlook add-ins. Now Outlook developers use Add-in Express and the ADX Extensions for Outlook to create protected Outlook-based solutions with feature-rich Outlook add-ins.

The ADX Extensions for Outlook is built on our exclusive Add-in Express™ technology and based on true RAD approaches inherited from Microsoft shared solutions which provides the most flexible and fastest way to program stable and powerful Outlook-based solutions. The flexibility of the ADX X for Outlook object model lets you access every level of the object hierarchy; the ADX X for Outlook event model gives you code the precise action.

The ADX Extensions for Outlook was architected from the ground up to overstep the limits of existing technology and to deliver tomorrow’s solutions today. We used all our experience in Outlook, .NET and VCL development and, in fact, released a unique product.

Editions of the ADX Extensions for Outlook

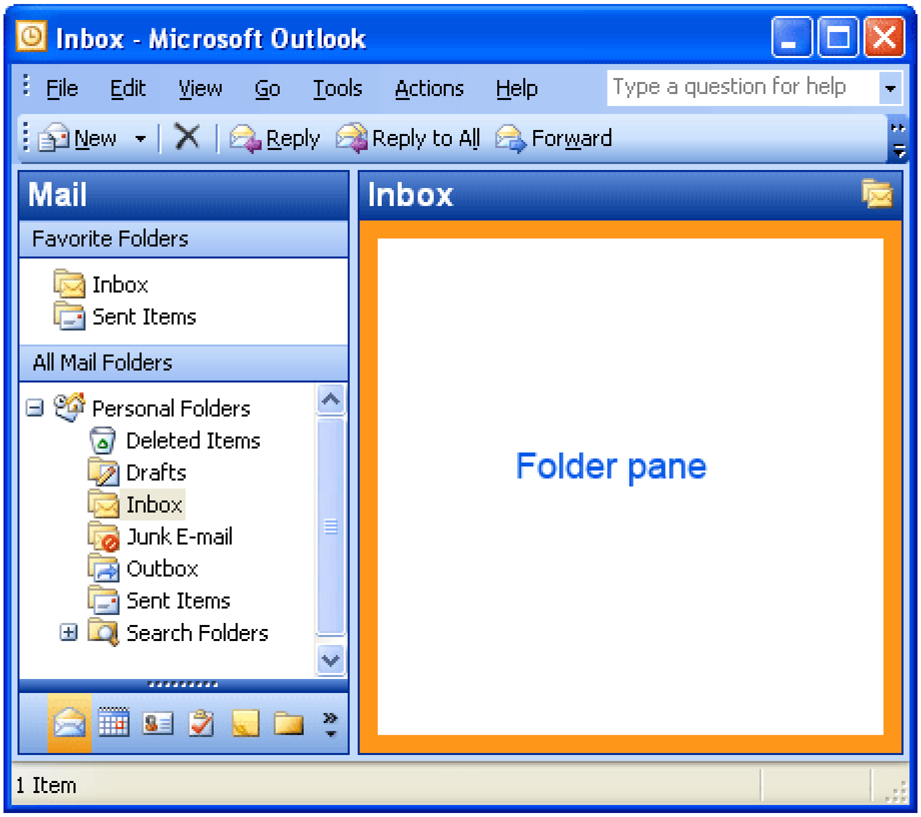
Now the ADX Extensions for Outlook is in one .NET edition. However, to support our VCL customers we plan to release its VCL edition. .NET Edition of the ADX Extensions for Outlook (ADX.NET X for Outlook) supports Visual Basic .NET, C# and Delphi .NET on .NET Framework 1.1 and 2.0. The complete system requirements are listed further.

What are folder panes and sub-panes

The ADX Extensions for Outlook allows a developer to embed his forms into folder panes and folder sub-panes. In the Microsoft Outlook terms, the folder pane is the area where items (messages, tasks, notes, etc.) of the selected Outlook folder are shown.

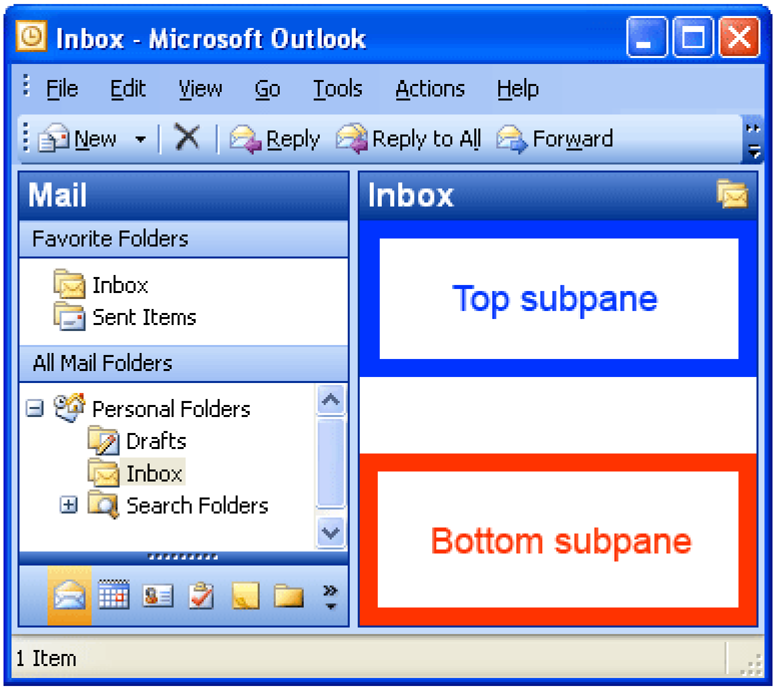
Folder panes

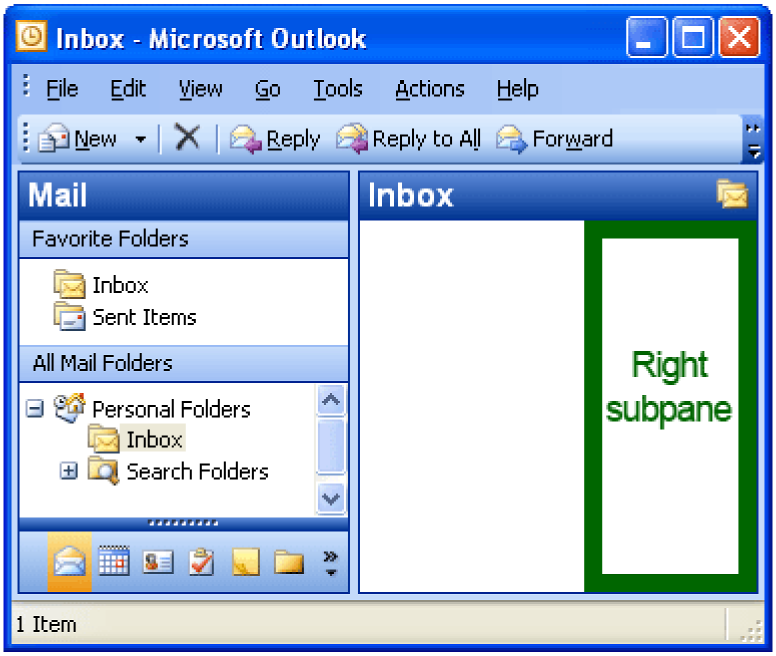
The ADX Extensions for Outlook allows you to replace the content of the folder pane with your own form. In this case ADX X for Outlook embeds your form into a special folder web-view.



Folder sub-panes

Also, the ADX Extensions for Outlook allows you to embed your custom form into a sub-pane as shown on the pictures below. Now in the view of UI design, the ADX Extensions for Outlook supports embedding one form into one sub-pane (top, bottom or right) only.





System requirements

The ADX Extensions for Outlook (ADX X for Outlook) is compatible with all Outlook versions starting from Outlook 2000 and requires Add-in Express installed.

Required Add-in Express versions

ADX.NET X for Outlook	ADX.VCL X for Outlook
▪ Add-in Express .NET version 2.4.1757 or higher	▪ N/A

Supported Outlook versions

ADX.NET X for Outlook	ADX.VCL X for Outlook
▪ Outlook 2000 with / without updates ▪ Outlook 2002 (XP) with / without updates ▪ Outlook 2003 with / without updates	▪ N/A

Supported programming languages

ADX.NET X for Outlook	ADX.VCL X for Outlook
▪ Visual Basic .NET 2005 ▪ Visual Basic .NET 2003 ▪ Visual C# .NET 2005 ▪ Visual C# .NET 2003	▪ N/A

Supported IDE versions

ADX.NET X for Outlook	ADX.VCL X for Outlook
▪ Visual Studio .NET 2005, Team Suite (all editions)	▪ N/A

ADX.NET X for Outlook	ADX.VCL X for Outlook
▪ Visual Studio .NET 2005, Professional ▪ Visual Studio .NET 2005, Standard ▪ Visual Studio .NET 2003, Enterprise (all editions) ▪ Visual Studio .NET 2003, Professional ▪ Visual Studio .NET 2003, Standard	

Note

- Evaluation or trial versions of Visual Studio and Borland Delphi are not supported.
- Express versions of Visual Studio 2005 are not supported.

Getting Support

If you own a copy of the ADX Extensions for Outlook, you are entitled to certain benefits of support services offered by Add-in Express Team.

Register on website

Visit our website and create a member profile. It is important to have your latest information entered into your member profile. This ensures that our support service team will deliver you support e-mail.

Note

- You can use the direct link to create a member profile.

Getting Help

You can obtain technical support using our website at www.add-in-express.com. Each registered user can submit support requests via a special web-form. In addition, on our website there is a customer community where you can always get help or advise, the HOWTOs section with "how to" examples, reference add-ins (ADX Toys), Premium Zone and much more.

Important

- Please consult support service options of your Technical Support subscription before submitting a support request.

Getting started

The ADX Extensions for Outlook is a visual tool. It is based on Windows API and comprised of over twenty internal classes. But all the internal classes have been designed to provide only two public components. This makes ADX X for Outlook easy-to-use in accordance with the true RAD paradigm. So, the ADX Extensions for Outlook provides very comfortable way to enhance the GUI of your Outlook add-ins with minimal service coding. You write your applied code only.

This section shows how you can quickly get started using the ADX Extensions for Outlook in Visual Studio .NET and Borland Delphi and describes what the ADX Extensions for Outlook adds to your add-ins based on Add-in Express.

Be Aware

The examples and their descriptions given here are very simple and brief. Don’t feel intimidated. To use the ADX Extensions for Outlook is quite easy. This tool allows you to start quickly with its functionality, to avoid any difficulties and various pitfalls when embedding your forms into Outlook windows. However, you should take into account that we deliberately avoid any descriptions of Outlook objects, their properties, methods and events. This documentation implies that you have some experience in using Add-in Express for developing Outlook add-ins as well as some experience with the Outlook object model.

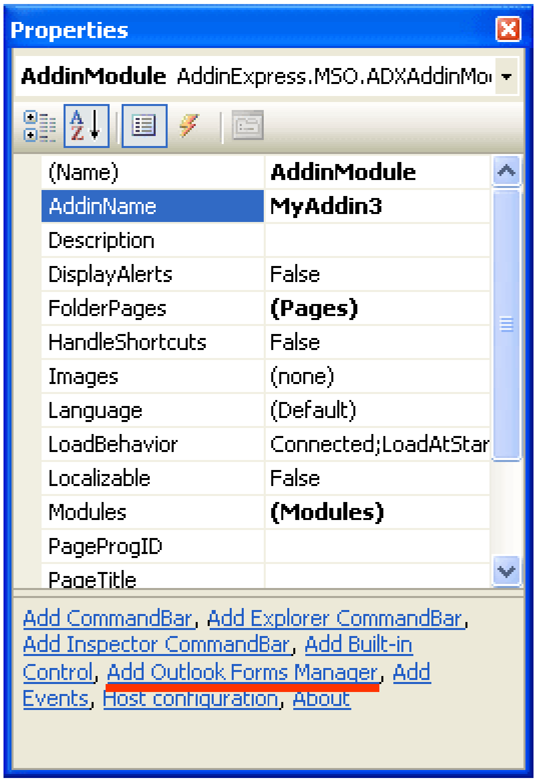
What ADX.NET X for Outlook adds to Add-in Express .NET

The ADX Extensions for Outlook is a plug-in for Add-in Express that adds its own command, wizard and components, namely:

- A **new command**, “Add Outlook Forms Manager”, added to the add-in module. It enables the ADX Extensions for Outlook for the current add-in and adds a special component that controls and centralizes your forms embedded into Outlook windows.
- A **new wizard**, “ADX Outlook Form”, added to the “Add New Item” dialog of the add-in project. It adds a new Outlook form module to the current add-in project.
- Several new components that implement the ADX X Extensions for Outlook functionality such as the **Outlook Forms Manager** and **Embedded Outlook Forms**.

New command

The ADX Extensions for Outlook adds a new command to the add-in module command set, “**Add Outlook Forms Manager**”.



AddinModule AddinExpress.MSO.ADXAddinMo

(Name)	**AddinModule**
AddinName	**MyAddin3**
Description	
DisplayAlerts	False
FolderPages	**(Pages)**
HandleShortcuts	False
Images	(none)
Language	(Default)
LoadBehavior	Connected;LoadAtStar
Localizable	False
Modules	**(Modules)**
PageProgID	
PageTitle	

Add CommandBar, Add Explorer CommandBar, Add Inspector CommandBar, Add Built-in Control, Add Outlook Forms Manager, Add Events, Host configuration, About

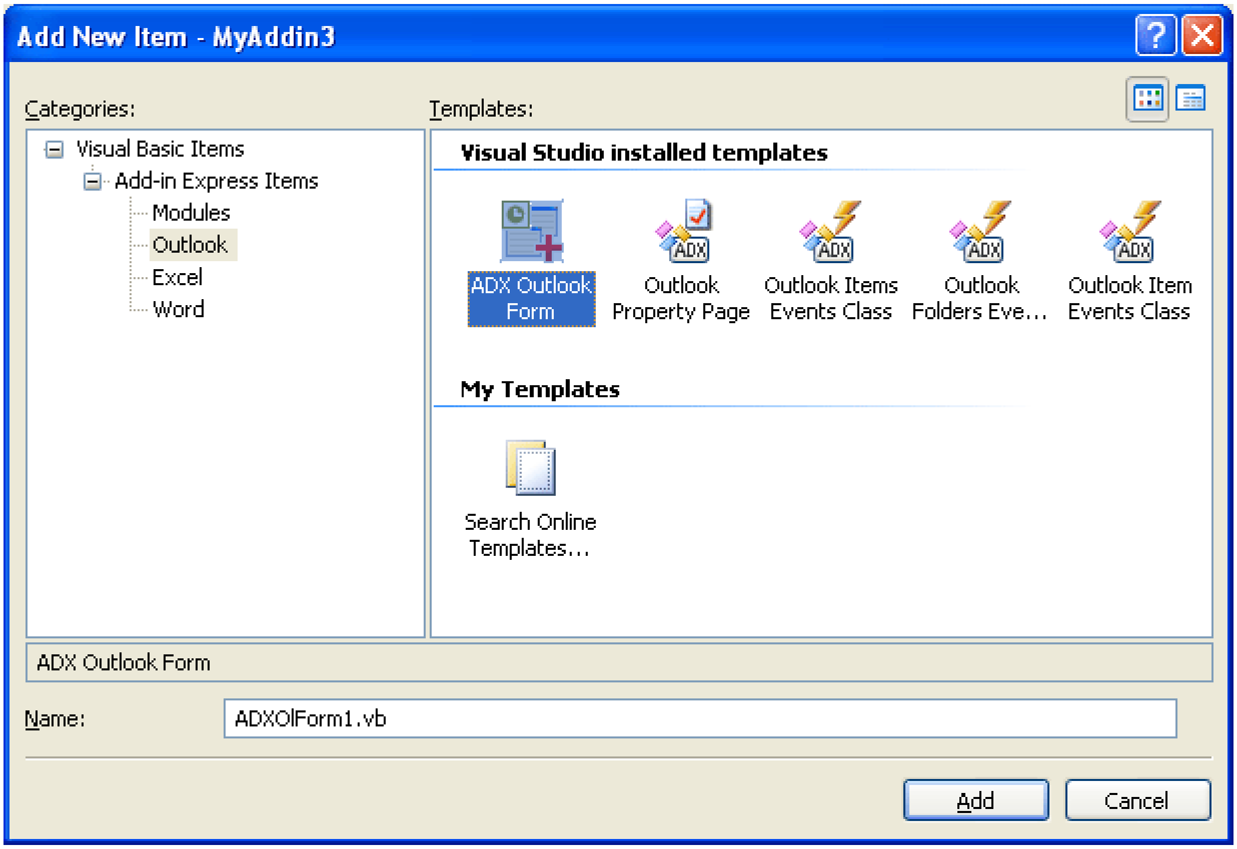
The command enables the ADX Extensions for Outlook for the current add-in solution, includes all references necessary to support ADX X for Outlook by the current add-in setup project and the add-in shim, and adds a special component, **ADXOlFormsManager** (the Outlook Forms Manager), to the add-in module.

Note

- You should run the command before adding embedded forms.

New wizard

The ADX Extensions for Outlook provides a special form class that implements a form embedded into the Outlook windows. You can add a new descendant of this class via a special wizard, **ADX Outlook Form**, available through the Add New Item dialog of the add-in project.



ADX Outlook Form

Name: ADXOlForm1.vb

Note

- You can run the wizard after running the “Add Outlook Forms Manager” command.

The Outlook Forms Manager

The ADX Extensions for Outlook publishes two components used for embedding your forms into Outlook. The most important component of ADX X for Outlook is the Outlook Forms Manager, an instance of the ADXOlFormsManager class added by the command described above.

The Outlook Forms Manager centralizes and controls your forms and binds them to Outlook folders. The table below shows some properties and events of the component. ADXOlFormsManager is a member of the **AddinExpress.OL.2005** (2003) namespace.

Property, event or method	Description
▪ The **Items** collection	An item of the collection binds one specified form to one or several Outlook folders. The form is specified by the FormClassName property of the item, the folders are specified by three special properties, FolderName, FoldersNames and ExplorerItemTypes that are common for all Outlook-related components provided by Add-in Express.
▪ The **AddinModule** property	Returns a reference to the add-in module that contains the Outlook Forms Manager component.
▪ The **CurrentForm** property	Returns an instance of your form contained in the active Explorer window.
▪ The **OutlookAppObj** property	Returns an instance of Outlook.Application that hosts your add-in.
▪ The **ADXBeforeFolderSwitch** event	Occurs before the Outlook Explorer window goes to a new folder, either as a result of a user action or through the program code. You can use this event to finish all active processes in the current form embedded into the **current folder view**.
▪ The **ADXFolderSwitch** event	Occurs when the Outlook Explorer window goes to a new folder, either as a result of a user action or through the program code. You can use this event to start any process in the form embedded into the **new folder views**.

Forms embedded into Outlook

The ADX Extensions for Outlook implements a special form class that can be embedded into Outlook windows. The class is called **ADXOlForm** and is a descendant of Windows.Forms.Form. To be embedded into Outlook windows all your Outlook forms should be descendants of ADXOlForm. You can add a new embedded form to your add-in project via the wizard described below.

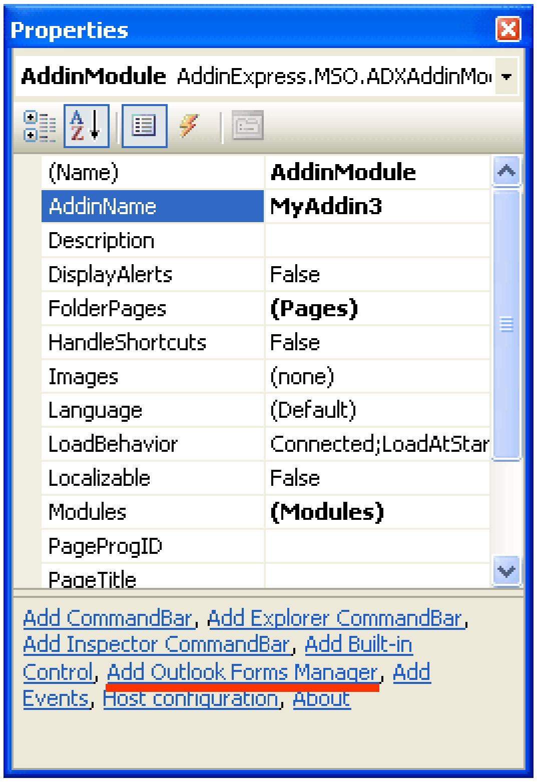
ADXOlForm published several Outlook-specific properties and events that can be used to access Outlook objects from an embedded form. The table below describes some properties and events of ADXOlForm. ADXOlForm is a member of the **AddinExpress.OL.2005** (2003) namespace.

Property, event or method	Description
Outlook-specific	
▪ The **OutlookAppObj** property	Returns a reference to the instance of Outlook.Application that hosts the add-in.
▪ The **ExplorerObj** property	Returns a reference to the instance of Outlook.Explorer in the context of which the form was created.
▪ The **FolderObj** property	Returns a reference to the instance of Outlook.MAPIFolder in the context of which the form was created.
▪ The **FolderItemsObj** property	Returns a reference to the Items collection of the FolderObj property.
▪ The **ADXBeforeFormShow** event	Occurs each time before the form is shown.
▪ The **ADXAfterFormShow** event	Occurs each time after the form is shown.
▪ The **ADXSelectionChange** event	Retranslates the SelectionChange event of Outlook.
Others	
▪ The **FormsManager** property	Returns a reference to the Outlook Forms Manager of the add-in.
▪ The **Item** property	Returns the item of the Items collection of the Outlook Forms Manager that created the form.

Your first folder sub-pane

1. Add the Outlook Forms Manager

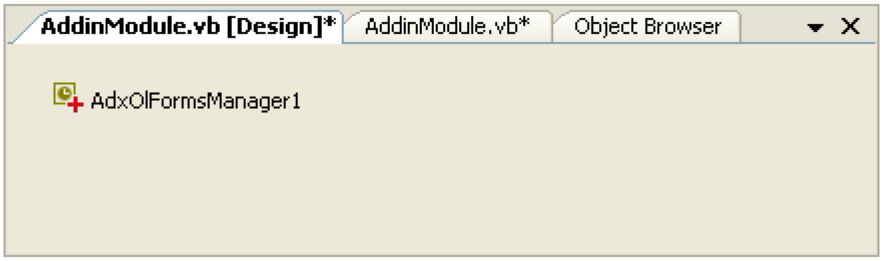
The ADX Extensions for Outlook adds to the add-in module commands set a special command, "**Add Outlook Forms Manager**", available on the command area of the Properties window or by right-clicking the add-in module.



(Name)	**AddinModule**
AddinName	**MyAddin3**
Description	
DisplayAlerts	False
FolderPages	**(Pages)**
HandleShortcuts	False
Images	(none)
Language	(Default)
LoadBehavior	Connected;LoadAtStar
Localizable	False
Modules	**(Modules)**
PageProgID	
PageTitle	

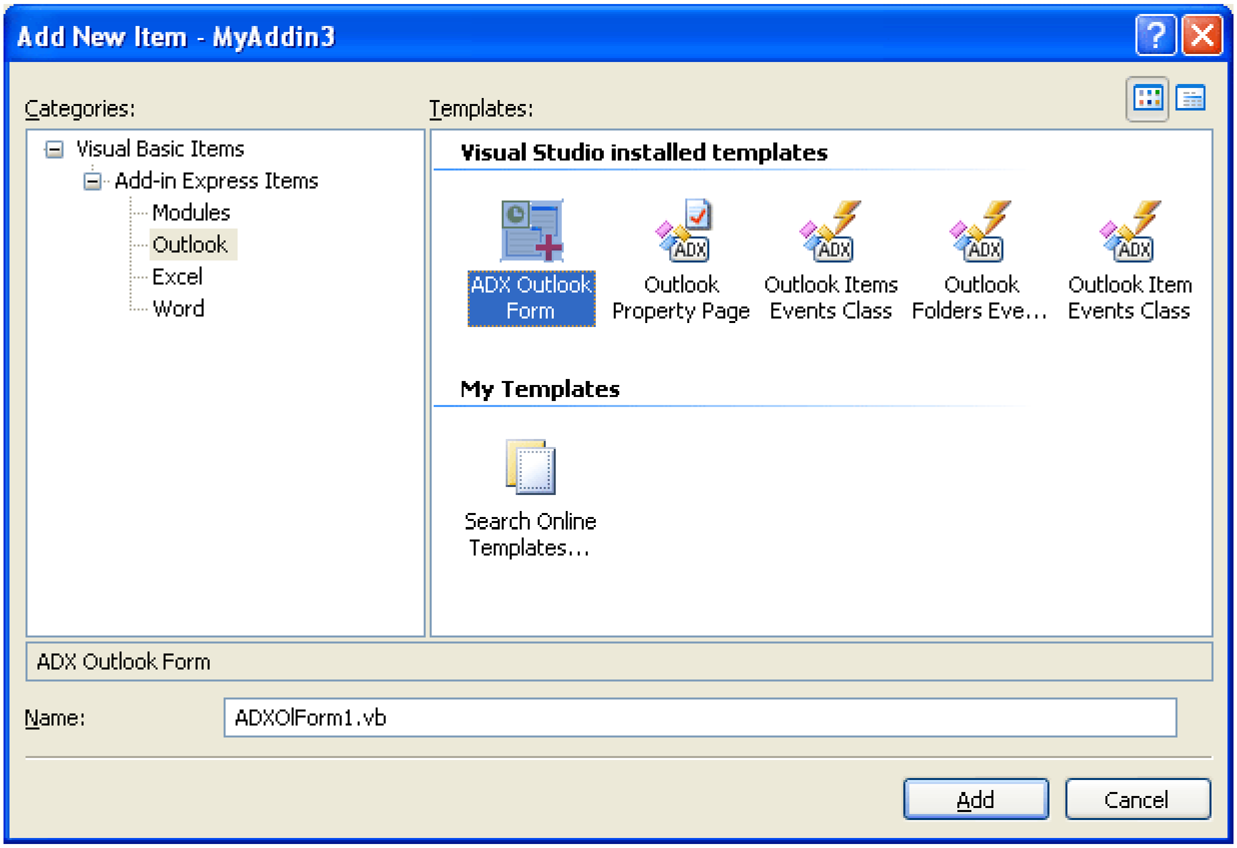
Add CommandBar, Add Explorer CommandBar, Add Inspector CommandBar, Add Built-in Control, Add Outlook Forms Manager, Add Events, Host configuration, About

To include the ADX Extensions for Outlook functionality in your Outlook add-in project, open the project, select the add-in module and run the "Add Outlook Forms Manager" command. The command adds the Outlook Forms Manager component to the add-in module.



2. Add a new embedded form

Then, run the "**ADX Outlook Form**" wizard from the Add New Item dialog of the add-in project. The wizard adds a new form module to the add-in project.

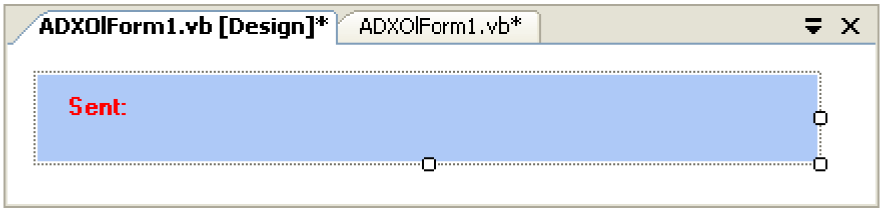


ADX Outlook Form

Name: ADXOlForm1.vb

3. Customize the form

On your form you can use any .NET controls such as calendars, edit boxes, grids, list views, etc. In this example we added a label to show when the selected message was sent.



4. Access to Outlook object

ADXOlForm provides access to the base Outlook objects and events. For example, you can handle the ADXSelectionChange event to synchronize your form with selection changes in the current explorer window:

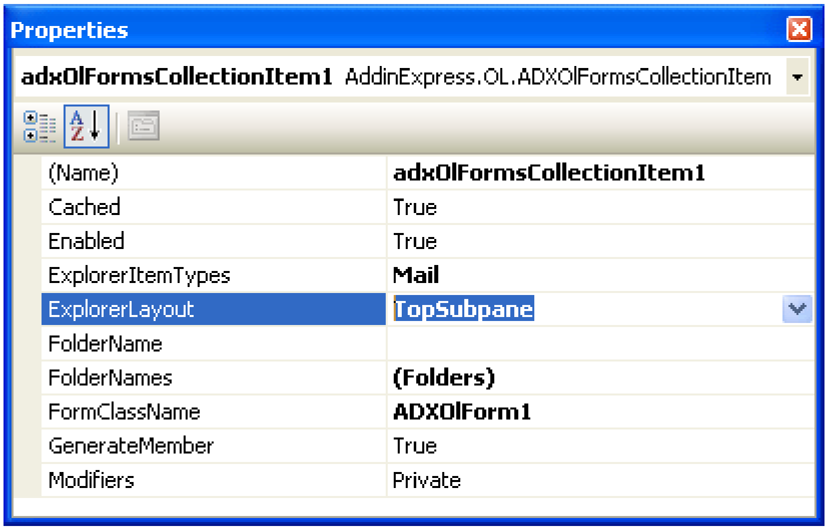
```
C#
  private void ADXOlForm1_ADXSelectionChange()
   {
      Outlook._Application OlApp = this.OutlookAppObj as Outlook._Application;
      Outlook.Selection Selection = OlApp.ActiveExplorer().Selection;
      if (Selection.Count != 0)
      {
         Outlook.MailItem item = Selection[1] as Outlook.MailItem;
         SentLabel.Text = "Sent: " + item.SentOn;
      }
   }

VB.NET
   Private Sub ADXOlForm1_ADXSelectionChange() Handles Me.ADXSelectionChange
      Dim OlApp As Outlook._Application = CType(OutlookAppObj, Outlook._Application)
      Dim Selection As Outlook.Selection = OlApp.ActiveExplorer().Selection
      Dim Item As Outlook.MailItem

      If Selection.Count <> 0 Then
         Item = CType(Selection(1), Outlook.MailItem)
         SentLabel.Text = "Sent: " + Item.SentOn
      End If
   End Sub
```

4. Binding the form to Outlook folders

To specify the folders which your form is displayed for, you should add a new item to the Items collection of the Outlook Forms Manager, select your form class in the FormClassName property and specify Outlook folder via three special properties, FolderName, FoldersNames and ExplorerItemTypes, that are common for all Outlook-related components provided by Add-in Express. Then, with the ExplorerLayout property you specify the pane or sub-pane which the form is placed on.

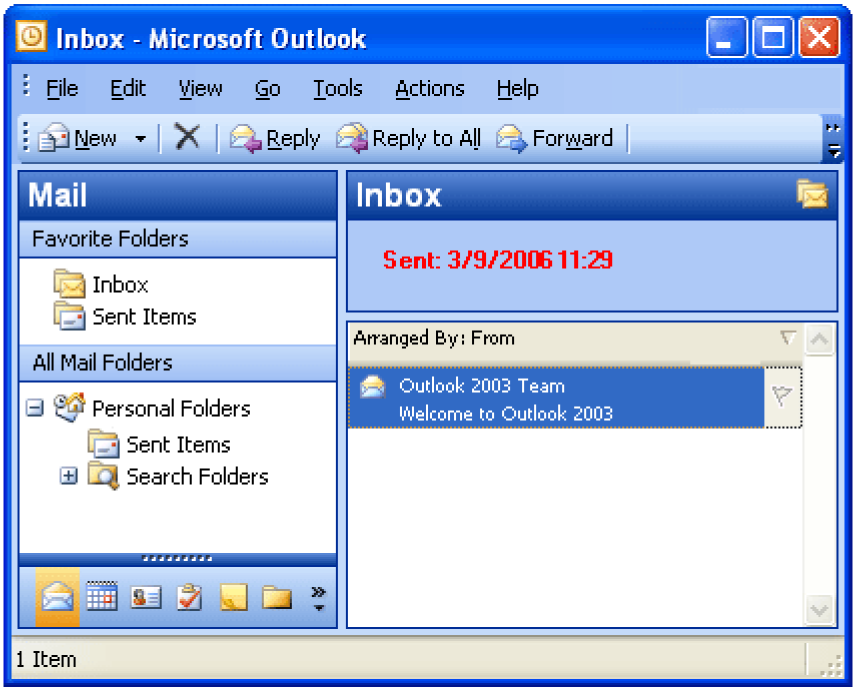


adxOlFormsCollectionItem1 AddinExpress.OL.ADXOlFormsCollectionItem

(Name)	**adxOlFormsCollectionItem1**
Cached	True
Enabled	True
ExplorerItemTypes	**Mail**
ExplorerLayout	**TopSubpane**
FolderName	
FolderNames	**(Folders)**
FormClassName	**ADXOlForm1**
GenerateMember	True
Modifiers	Private

5. Run your add-in and voila

Finally, you can rebuild the add-in project, run Outlook and find your form embedded.



Sent: 3/9/2006 11:29

Outlook 2003 Team
Welcome to Outlook 2003

Insight of Folder Sub-panes

At first glance, the ADX Extensions for Outlook seems very easy of use. It is true if you embed one sub-pane and bind it to one folder. However, you may face some difficulties if your project contains several embedded forms, if your forms are dynamically bound to folders, if two or more forms are bound to one folder, etc.

This section was created to give you the insight of “why” the ADX Extensions for Outlook does something exactly the way it does. Please note that ADX Extensions for Outlook HOWTOs contains several examples illustrating all described below.

Binding techniques

Above we described a very simple example that shows how to bind one form to several folders. Remember that your form is bound to Outlook folders by an item of the Items collection of the Outlook Forms Manager. In addition, ADX X for Outlook allows you to bind one form to several Outlook folders through several items of this collection. For example:

```
VB.NET

  ' Item #1 binds one form to all mail folders
  Me.AdxOlFormsManager1.Items.Add(Me.AdxOlFormsCollectionItem1)
  Me.AdxOlFormsCollectionItem1.ExplorerItemTypes = _
      AddinExpress.OL.ADXOlExplorerItemTypes.olMailItem
  Me.AdxOlFormsCollectionItem1.FormClassName = "ADXOlForm1"

  ' Item #2 binds the same form to the "Personal Folders\Special" folder
  Me.AdxOlFormsManager1.Items.Add(Me.AdxOlFormsCollectionItem1)
  Me.AdxOlFormsCollectionItem1.FolderName = "Personal Folders\Special"
  Me.AdxOlFormsCollectionItem1.FormClassName = "ADXOlForm1"
```

What can you need it for? Using different items binding the same form to Outlook folders you can create logical sets to control the form’s behavior, to change bound folders, etc. For example, you can disable or enable your forms separately for all mail folders or for the "Personal Folders\Special" folder. Also, you can change forms layout to place your form on the top sub-pane for mail folders and on the bottom sub-pane for the "Personal Folders\Special" folder. Another purpose is to replace the form class name. It gives you a possibility to dynamically and flexibly bind different forms to different folders.

When can you bind your forms to Outlook folders? First of all, you can do this via the Items collection designer of the Outlook Forms Manager, as shown above. If you need to do this by code (on-the-fly), you should handle the **ADXBeforeFolderSwitch** event of the Outlook Forms Manager and create new items here, delete existing items, enable/disable them, bind to other folders, etc. In general, only in the handler of ADXBeforeFolderSwitch you can change the Items collection of the Outlook Forms Manager. Below you can find more detailed information about events.

Cached forms

Outlook creates a new instance of its any windows embedded into the Explorer and Inspector windows whenever the user switches between folders. In contrast to this behavior, the ADX Extensions for Outlook may create one instance of the embedded form when the user visits a folder first time and disposes this instance when the user closes Outlook. So, we say that ADX X for Outlook caches embedded forms, which allows your add-in to considerably expedite switching between Outlook folders.

The ADX Extensions for Outlook provides two very important built-in caching strategies that you should know about. You can enable or disable caching via the Cached property of the item that binds your form to Outlook folders.

NewInstanceForEachFolder

By default, ADX X for Outlook creates one instance of your form for each folder visited by the user in the current Explorer window. This strategy is called NewInstanceForEachFolder. For example, if you bind your form to two folders, two instances will be created if the user visits both folders bound to your form in one Explorer window. If the user opens the second Explorer window and visits two folders in each Explorer window, four instances of your form will be created.

This caching strategy has been developed to give you a possibility to *create unique content for each folder visited by the user*. Please take it into account when developing embedded forms and their initialization and finalization code. Perhaps, you may need to have common data for all instances of your embedded forms and to show the data on each form instance. Then you can use the add-in module accessible from your form via the AddinModule property.

Note

- By default, all forms are cached by the NewInstanceForEachFolder strategy. You can disable or change this via the Cached property of the corresponding collection item of the Outlook Forms Manager.
- If you disable caching for your form, a new instance of your form will be created each time the user visits each folder bound to your form.
- Any run-time changes of the corresponding item dispose all cached instances.

OneInstanceForAllFolders

The second built-in caching strategy is OneInstanceForAllFolders. It creates one instance of your forms for all folders bound to your forms and shows this instance for all folders in one Explorer window. For example, if you bind one form to all mail folders, one instance will be created if the user visits any number of mail folders in one Explorer window. If the user opens the second Explorer window and visits any mail folders in two Explorer windows, two instances of your form will be created (for each Explorer).

Why are instances of your form created for each Explorer window? In general, MS Windows cannot show one instance (window) on two windows at the same time. So, your forms should have different instances for each Explorer window.

This caching strategy has been developed to give you a possibility to *create one form for all folders visited by the user*. Please take it into account when developing embedded forms and their initialization and finalization code.

Note

- Please note that caching strategies is a subject to change.