Official Guide

Toolbar Controls
for Microsoft® Office®

Getting Started
The ADX Toolbar Controls for Microsoft Office, .NET Edition

The ADX Toolbar Controls™ for Microsoft® Office®

Document version **1.0**
Revised at **28-Jul-06**
Product version **1.0**

Content

Introduction

Microsoft gives Office developers a way to extend and improve its Office applications using several special technologies. Office 2000 started IDTExensibility2 for creating COM add-ins. With Office 2002 (XP) Microsoft published the Smart Tag technology to introduce context sensitivity into its applications and the Excel RTD Server technology to replace archaic DDE with 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 embedding their applied code into Microsoft Office applications. However, Office developers want to achieve much more.

This document describes the ADX Toolbar Controls™ for Microsoft Office and what it provides to extent the technologies supported by Microsoft.

Welcome to the Toolbar Controls

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

The Toolbar Controls for Microsoft Office is built on our own exclusive Add-in Express™ technology and based on true RAD approaches inherited from Microsoft shared solutions which provides a flexible and the fastest way to program stable and powerful Office-based solutions. The flexibility of the Toolbar Controls object model lets you address every level of the object hierarchy; the Toolbar Controls event model lets you code to the precise action.

The Toolbar Controls for Microsoft Office was architected from the ground up to overstep the limits of the existing technology and to deliver tomorrow's solutions today. We gathered all our experience in MS Office, .NET and VCL development and, in fact, released a unique product.

What the Toolbar Controls for Microsoft Office is

The Toolbar Controls for Microsoft Office adds one feature to Add-in Express. With the Toolbar Controls your can use any controls, not only Office built-in controls, on your command bars. Now you can add tree-views, grids, diagrams, edit boxes, reports, etc. to your command bars. It is what the Toolbar Controls is intended for.

Be Aware

The examples and their descriptions given here are very simple and brief. Don't feel intimidated. To use the Toolbar Controls for Microsoft Office is quite easy. This tool allows you to start quickly and avoid any difficulties and pitfalls when adding your controls to Office command bars. However, you should take into account that we deliberately avoid any descriptions of MS Office objects, their properties, methods and events. This documentation implies that you have some experience in using Add-in Express for developing Microsoft Office add-ins as well as some experience with the Microsoft Office object models.

System requirements

The Toolbar Controls for Microsoft Office supports Visual Basic .NET 2003 and 2005, Visual C# 2003 and 2005, Visual C++ .NET 2003 and 2005, and RemObjects Chrome 1.5.3 and higher. The complete system requirements are listed below.

The Toolbar Controls for Microsoft Office is compatible with all public Office versions starting from Office 2000 and requires Add-in Express installed.

Required Add-in Express versions

Toolbar Controls .NET	Toolbar Controls VCL
▪ Add-in Express .NET version 2.7 or higher 2.x	▪ N/A

Supported Outlook versions

Toolbar Controls .NET	Toolbar Controls VCL
▪ Office 2000 with / without updates ▪ Office 2002 (XP) with / without updates ▪ Office 2003 with / without updates	▪ N/A

Supported programming languages

Toolbar Controls .NET	Toolbar Controls VCL
▪ Visual Basic .NET 2005 ▪ Visual Basic .NET 2003 ▪ Visual C# .NET 2005 ▪ Visual C# .NET 2003 ▪ RemObjects Chrome 1.5 ▪ Borland Delphi for .NET 2005	▪ N/A

▪ Borland Delphi for NET 2006	

Supported IDE versions

Toolbar Controls .NET	Toolbar Controls VCL
▪ Visual Studio .NET 2005, Team System (all editions) ▪ Visual Studio .NET 2005, Professional ▪ Visual Studio .NET 2005, Standard ▪ Visual Basic .NET 2005, Standard ▪ Visual C# .NET 2005, Standard ▪ Visual Basic .NET 2005, Standard ▪ Visual C++ .NET 2005, Standard ▪ Visual Studio .NET 2003, Enterprise (all editions) ▪ Visual Studio .NET 2003, Professional ▪ Visual Studio .NET 2003, Standard ▪ Visual Basic .NET 2003, Standard ▪ Visual C# .NET 2003, Standard ▪ Visual C++ .NET 2003, Standard ▪ Borland Delphi for .NET 2005, Professional and other high-level edition ▪ Borland Delphi for .NET 2005, Professional and other high-level edition	▪ N/A

Note

▪ Evaluation or trial versions of Visual Studio, RemObjects Chrome and Borland Delphi are not supported.

Getting support

If you own a copy of the Toolbar Controls, you are entitled to certain benefits regarding support services offered by the Add-in Express Team.

Register on website

Visit our website and create a member profile. It is important that your most current information is entered into your member profile. This will ensure that our support service team can deliver support correspondence.

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. Every registered user can submit support issues via a special web-form. In addition, on our website there is a customer community actively supported by the Add-in Express Team, the HOWTOs section with "how to" examples, reference add-ins (ADX Toys), Premium Zone and much more.

Important

- Please consult the support service options of your subscription before submitting a support issue.

Getting started on .NET

The Toolbar Control for Microsoft Office is a visual tool. It is based on the Windows API and comprised of over twenty internal classes. But all the internal classes were designed to provide only one type of public components, the Control Adapters. This makes the Toolbar Controls easy-to-use in accordance with the true RAD paradigm. So, the Toolbar Control for Microsoft Office provides a very comfortable way to enhance the GUI of your MS Office add-ins with minimal service coding. You write your applied code only.

This section shows how you can quickly get started with the Toolbar Control for Microsoft Office in Visual Studio .NET and describes what the Toolbar Controls adds to your add-ins based on Add-in Express.

What the Toolbar Controls adds to Add-in Express

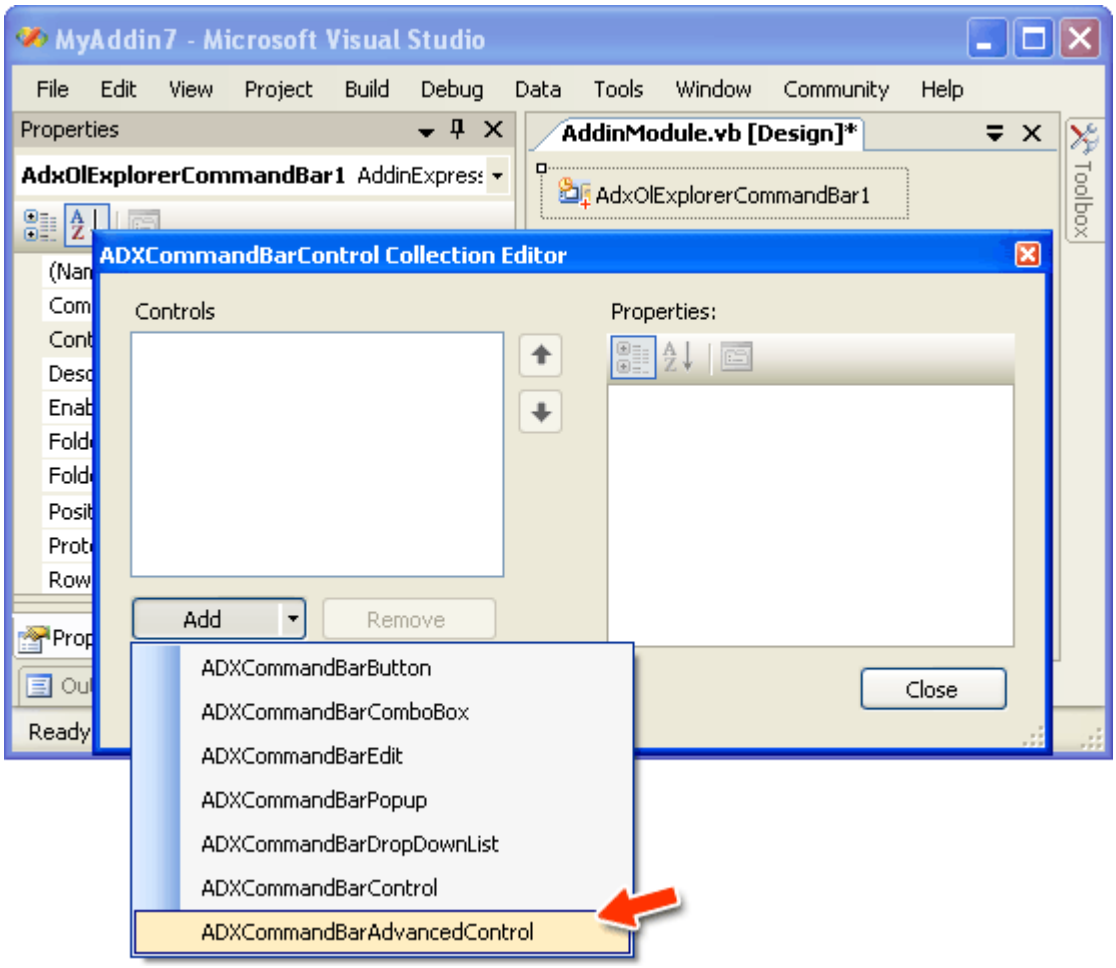
Terms

To make the text below easy to read, let's define three terms, namely:

- *Command bar controls* are controls such as command bar buttons and command bar combo boxes provided by the Office object model. These controls are built-in Office controls and supported by Add-in Express.
- *Non-Office controls* are any controls, both .NET built-in and third-party controls, such as tree-views, grids, edit boxes, labels, user controls, etc. Usually, you use these controls on forms of your Windows applications.
- *Advanced command bar control* is an instance or ADXCommandBarAdvancedControl or the ADXCommandBarAdvancedControl class itself (depending on the context).

What ADXCommandBarAdvancedControl is

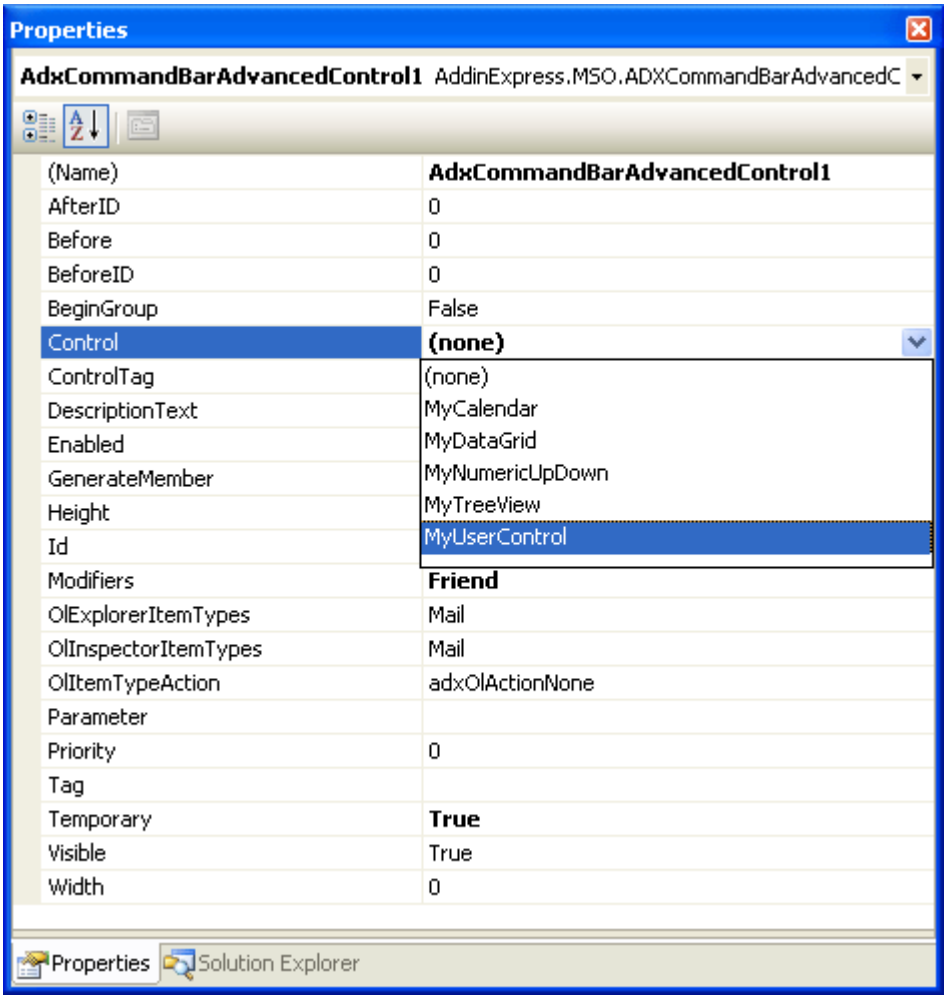
If you have developed at least one add-in based on Add-in Express, you probably ran into ADXCommandBarAdvancedControl when adding *command bar controls* to your command bars. Yes, it is that strange item of the Add button on the ADXCommandBarControl Collection editor (see the red arrow on the picture below). Add-in Express documentation says nothing about it. On our forums, to all questions regarding this we gave the answer like "ADXCommandBarAdvancedControl is reserved for future purposes". It's true. We reserved it for one more plug-in for our Add-in Express. That plug-in is the Toolbar Controls.



We reserved ADXCommandBarAdvancedControl to give you a chance to use any *non-Office controls* such as tree-views, grinds, labels, edit and combo boxes, diagrams on any Office command bars. Now you can add ADXCommandBarAdvancedControl, *an advanced command bar control*, to your command bar and bind it to any *non-Office control* you placed on the add-in module. Your grid, tree-view or image is placed on your command bar as the result.

ADXCommandBarAdvancedControl is a host for your controls

In addition to properties common for Office *command bar controls*, ADXCommandBarAdvancedControl has one more property. It is the **Control** property, the most important property. With this property you can select a *non-Office control* to place it on your command bar. Have a look at the picture below. The add-in module contains five controls – MyCalendar, MyDataGrid, MyNumericUpDown, MyTreeView and MyUserControl. The Control property asks you to select one of these controls. If you select MyUserControl, your add-in adds MyUserControl to your command bar. With the Control property ADXCommandBarAdvancedControl becomes a host for your *non-Office controls*.

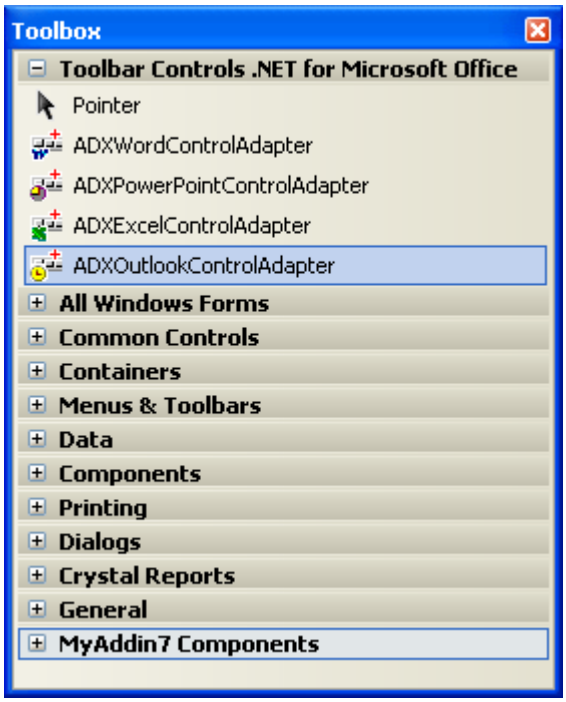


On .NET, ADXCommandBarAdvancedControl supports all controls based on System.Windows.Forms.Control. So, on your command bars, you can use both built-in controls and third-party controls based on System.Windows.Forms.Control. Just add them to the add-in module, add an *advanced command bar control* to your command bar and select your *non-Office control* in the Control property of ADXCommandBarAdvancedControl.

The Toolbar Controls is a set of control adapters

You may ask us what the Toolbar Controls described above does and what it is itself, if ADXCommandBarAdvancedControl is already included in Add-in Express. In general, ADXCommandBarAdvancedControl is still abstract in Add-in Express but it is implemented by the Toolbar Controls if it is plugged in Add-in Express. So, our answer is: the Toolbar Controls for Microsoft Office implements ADXCommandBarAdvancedControl for each Office application.

The Toolbar Controls adds a new tab, "Toolbar Controls for Microsoft Office", to the Toolbox and places several components on the tab (see the picture below). The Toolbar Controls supports each Office application by special components that we call Control Adapters. Only Control Adapters know how to add your controls to applications specific command bars. So, the Control Adapters are the Toolbar Controls itself.



All Control Adapters can be contained by the add-in module only. For example, you should add an ADXExcelControlAdapter to the add-in module if you wand to use *non-Office controls* in your Excel add-in. To use *non-Office controls* on several Office applications you should add several Control Adapters. For example, if you need to use *your controls* in your add-in that supports Outlook, Excel and Word, you should add three Control Adapters: ADXExcelControlAdapter, ADXWordControlAdapter and ADXOutlookControlAdapter to the add-in module.

Your first advanced command bar control

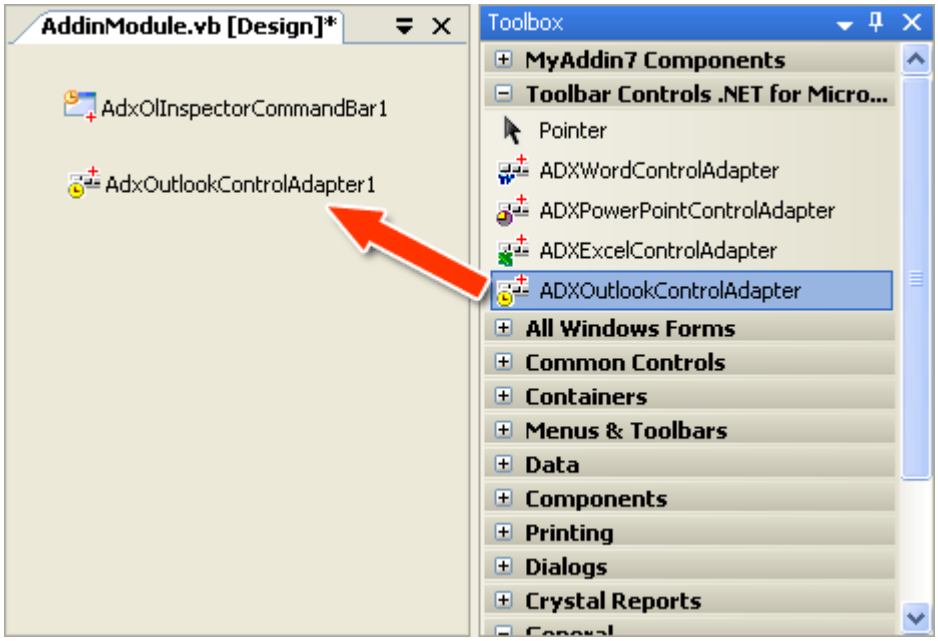
The first example is very simple. Now we are going to add a check box to a new right-docked command bar on the Outlook Inspector window. This checkbox, BossCheckbox :-) will allow us to add our boss's e-mail address to the BCC field. The example is described implying that you have some experience in using Add-in Express for developing Microsoft Office add-ins as well as some experience with the Microsoft Office object models.

To reproduce the example you can create a new Outlook add-in project or open the exiting one, add an ADXOlInspectorCommandBar to the add-in module and set adxMsoBarBottom to the Position property of the added command bar. Next step is to…

1. Add a Control Adapter

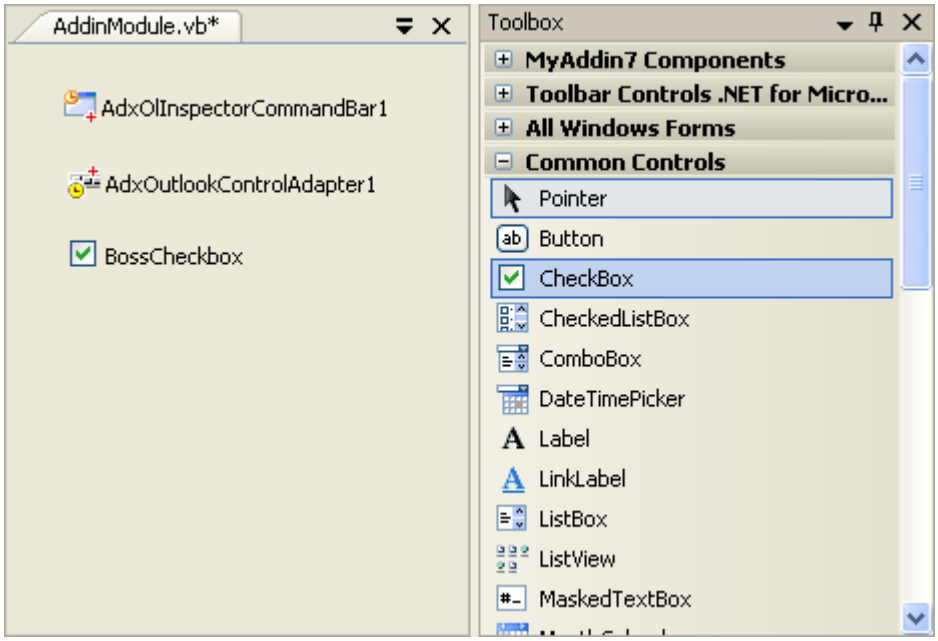
As we described above, the Toolbar Controls for Microsoft Office supports Office applications though special components that we call the Control Adapters. You can find them on the "Toolbar Controls for Microsoft Office" tab of the Toolbox.

The first step in using *non-Office controls* in your add-in is adding the corresponding control adapter to your add-in module. In this case we use an ADXOutlookControlAdapter.



2. Add your control

The add-in module can contain any components including controls. So, you can add BossCheckbox directly to your add-in module



Then, you can customize the checkbox. For example, like this:

```
VB.NET (from #Region "Component Designer generated code")
        Me.BossCheckbox.BackColor = System.Drawing.Color.FromArgb(CType(CType(255, Byte), Integer), _
           CType(CType(128, Byte), Integer), CType(CType(0, Byte), Integer))
        Me.BossCheckbox.Location = New System.Drawing.Point(0, 0)
        Me.BossCheckbox.Name = "BossCheckbox"
        Me.BossCheckbox.Size = New System.Drawing.Size(150, 24)
        Me.BossCheckbox.TabIndex = 0
        Me.BossCheckbox.Text = "BCC to my Boss"
        Me.BossCheckbox.UseVisualStyleBackColor = False
```

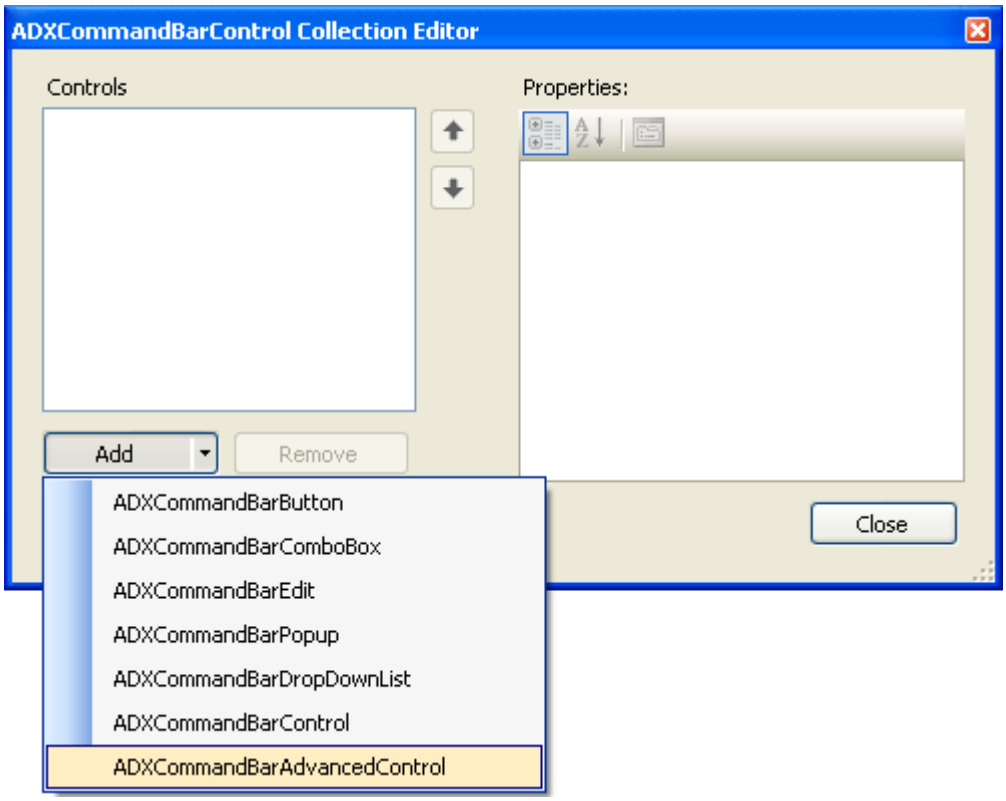
3. Handle your control

To BCC a message to your boss you need to handle the checkbox. We ourselves usually use the following code to BCC messages to our boss. Please note we do not describe Outlook programming here.

```
VB.NET
    Private Sub BossCheckbox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim Item As Outlook.MailItem
        Item = CType(OutlookApp.ActiveInspector.CurrentItem, Outlook.MailItem)
        If BossCheckbox.Checked Then
            Item.BCC = "myboss@mydomain.com"
        Else
            Item.BCC = ""
        End If
    End Sub
```

4. Bind your control to your command bar

You have added ADXOutlookControlAdapter and BossCheckbox to your add-in module but did not bind your check box to your command bar. To do this you should add an advanced command bar control to the Controls collection of your command bar:



Then, select the added ADXCommandBarAdvancedControl1 on the Properties window and select BossCheckbox in the Control property of the ADXCommandBarAdvancedControl1. That's all. You have done all necessary steps to place your checkbox on your command bar. Below we give the complete Initialize Components method of our add-in module that relates to our example:

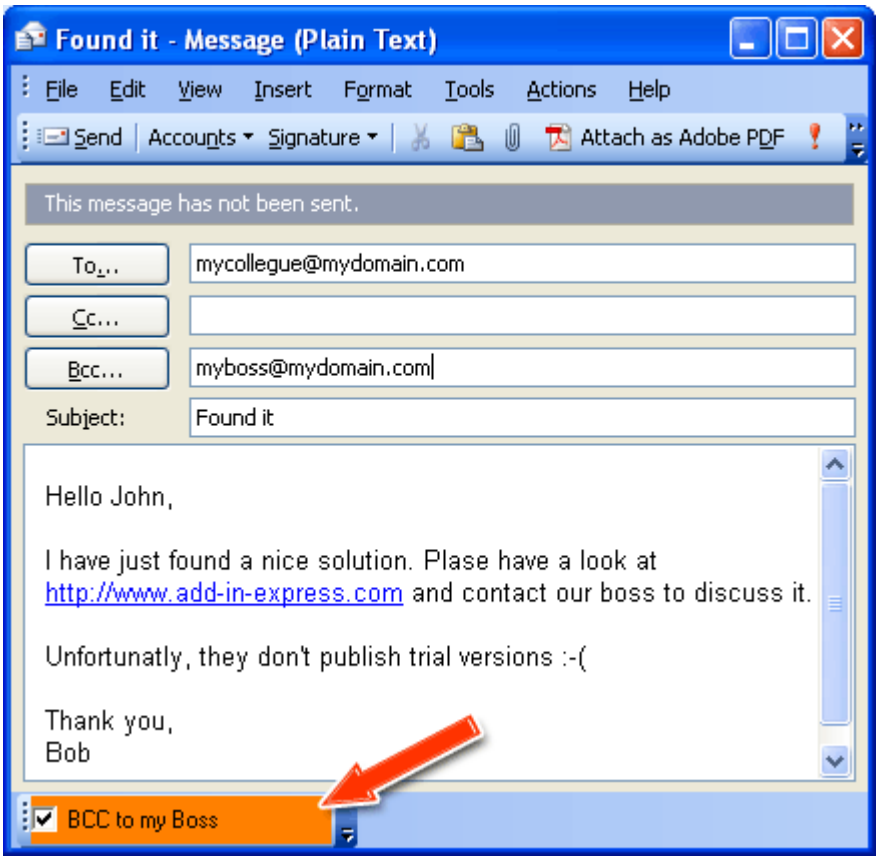
VB.NET

```
    Private Sub InitializeComponent()
        Me.components = New System.ComponentModel.Container
        Me.AdxOutlookControlAdapter1 = New _
            AddinExpress.ToolbarControls.ADXOutlookControlAdapter(Me.components)
        Me.AdxOlInspectorCommandBar1 = New AddinExpress.MSO.ADXOlInspectorCommandBar(Me.components)
        Me.BossCheckbox = New System.Windows.Forms.CheckBox
        Me.AdxCommandBarAdvancedControl1 = New AddinExpress.MSO.ADXCommandBarAdvancedControl(Me.components)
        '
        'AdxOlInspectorCommandBar1
        Me.AdxOlInspectorCommandBar1.CommandBarName = "AdxOlInspectorCommandBar1"
        Me.AdxOlInspectorCommandBar1.CommandBarTag = "6e212933-5790-40f5-8c25-4dd46c632f91"
        Me.AdxOlInspectorCommandBar1.Controls.Add(Me.AdxCommandBarAdvancedControl1)
        Me.AdxOlInspectorCommandBar1.Position = AddinExpress.MSO.ADXMsoBarPosition.adxMsoBarBottom
        Me.AdxOlInspectorCommandBar1.Temporary = True
        Me.AdxOlInspectorCommandBar1.UpdateCounter = 27
        '
        'BossCheckbox
        Me.BossCheckbox.BackColor = System.Drawing.Color.FromArgb(CType(CType(255, Byte), Integer), _
            CType(CType(128, Byte), Integer), CType(CType(0, Byte), Integer))
        Me.BossCheckbox.Location = New System.Drawing.Point(0, 0)
        Me.BossCheckbox.Name = "BossCheckbox"
        Me.BossCheckbox.Size = New System.Drawing.Size(150, 24)
        Me.BossCheckbox.TabIndex = 0
        Me.BossCheckbox.Text = "BCC to my Boss"
        Me.BossCheckbox.UseVisualStyleBackColor = False
        '
        'AdxCommandBarAdvancedControl1
        Me.AdxCommandBarAdvancedControl1.Control = Me.BossCheckbox
        Me.AdxCommandBarAdvancedControl1.ControlTag = "f9a07d50-6dce-41b1-911e-68d694f7d7eb"
        Me.AdxCommandBarAdvancedControl1.Temporary = True
        Me.AdxCommandBarAdvancedControl1.UpdateCounter = 1
```

```
        '
        'AddinModule
        Me.AddinName = "MyAddin"
        Me.ShimProgID = "MyAddinShim.Proxy"
        Me.SupportedApps = AddinExpress.MSO.ADXOfficeHostApp.ohaOutlook
    End Sub
```

5. Run your add-in

Finally, you can rebuild the add-in project, run Outlook and find your BossCheckbox:



Advanced understanding

At first glance, the Toolbar Controls for Microsoft Office seems very easy-to-use. All Office applications have unique features or troubles if using non-Office controls on their command bars.

ADXCommandBarAdvancedControl

As described above, the Toolbar Controls implements the ADXCommandBarAdvancedControl class that is still abstract in Add-in Express without the Toolbar Controls installed. In addition to properties common for all *command bar controls*, ADXCommandBarAdvancedControl provides two special properties related to the Toolbar Controls.

The Control property

The **Control** property binds its ADXCommandBarAdvancedControl to a *non-Office control* and can be used at design-time as well as at run-time. To place your *non-Office control* on your command bar you just select your control in the Control property at design-time, or set the Control property to an instance of your control at run-time:

```
VB.NET
    Private Sub AddinModule_AddinInitialize(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.AddinInitialize
        BossCheckbox = New System.Windows.Forms.CheckBox
        Me.AdxCommandBarAdvancedControl1.Control = BossCheckbox
    End Sub
```

The ActiveInstance property

The ActiveInstance property is read-only and returns the instance of the Control that was created for the current context. For example, you can initialize your control for the active Inspector by handling the InspectorActivate event:

```
VB.NET

    Private Sub adxOutlookEvents_InspectorActivate(ByVal sender As System.Object, _
        ByVal inspector As System.Object, ByVal folderName As System.String) _
        Handles adxOutlookEvents.InspectorActivate
```

```
    Dim ChkBox As System.Windows.Forms.CheckBox = Me.AdxCommandBarAdvancedControl1.ActiveInstance
    If Not IsNothing(ChkBox) Then ChkBox.Enabled = False
End Sub
```

Please note, the **ActiveInstance** property is not actual in most cases when you'd like to use it. You can always use any window activate events such as the InspectorActivate event of Outlook and WindowActivate event of Word. The table bellow shows you the order of event processing by the example of the Outlook Inspector window opened by the user.

1. Outlook fires the built-in NewInspector event. Add-in Express traps it and fires the NewInspector event of ADXOutlookEvents.			ActiveInstance returns NULL.
	2. ADXOutlookEvents runs your NewInspector event handlers.		ActiveInstance returns NULL.
		3. The Toolbar Controls creates an instance of your control.	ActiveInstance returns NULL.
		4. Outlook fires the built-in InspectorActivate event. Add-in Express handles it and fires the InspectorActivate event of ADXOutlookEvents.	ActiveInstance returns NULL.
		5. The Toolbar Controls creates an instance of your control for the opened Inspector. ADXOutlookEvents runs your InspectorActivate event handlers.	ActiveInstance returns the instance of your control that was cloned from your original control.

Control Adapters

All Office applications have different window architectures. All Office windows themselves are different. All our Control Adapters have a unified programming interface but different internal architectures that take into account the windows architecture of the corresponding applications. All features of all Control Adapters are described below.

Outlook

Outlook has two main windows – Explorer and Inspector windows. The user can open several Explorer and Inspector windows. Our Outlook Control Adapter supports *non-Office controls* on both the Explorer and

Inspector windows, and *creates an instance of your control whenever the user opens a new Explorer or Inspector window*.

Please note, *if Word is used as an e-mail editor*, Outlook uses MS Word as an Inspector window. In this case, Word is running in a separate process. It does not provide any ways to control all instances of your control. So, there is one feature with Word as an e-mail editor: the Outlook Control Adapter **hides** all instances of your control on all **inactive** Word Inspector windows, but shows them once the Inspector is activated.

Excel

In spite of the fact that Excel allows placing its windows on the Task Bar, all its command bars work like in MDI applications. So, your controls are created only once, when Excel is started. However, you can still use the WorkbookActivate, WindowActivate and SheetActivate events to initialize your *non-Office controls* according to the context.

Word

Word creates its command bars for all document windows, so your *non-Office controls* are instanced whenever the user opens a new window or a document. We recommend using the WindowActivate event to initialize your control for the current window.

PowerPoint

Notwithstanding the fact that PowerPoint makes possible placing its windows on the Task Bar, PowerPoint is an MDI application. So, your controls are created only once, when PowerPoint is started. However, you can still use the WindowActivate event to initialize your *non-Office controls* according to the context.