**How To Build an Office COM Add-in by Using Visual Basic .NET**

Article ID: 302896 - [View products that this article applies to.](http://support.microsoft.com/kb/302896#appliesto)

This article was previously published under Q302896

[Expand all](javascript:void(0);) | [Collapse all](javascript:void(0);)

**http://support.microsoft.com/library/images/support/en-us/20x20_grey_minus.png**[**On This Page**](javascript:void(0);)

* [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)SUMMARY](http://support.microsoft.com/kb/302896)
  + [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)The IDTExensibility2 Interface](http://support.microsoft.com/kb/302896)
    - [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)OnConnection](http://support.microsoft.com/kb/302896)
    - [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)OnDisconnection](http://support.microsoft.com/kb/302896)
    - [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)OnAddInsUpdate](http://support.microsoft.com/kb/302896)
    - [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)OnStartupComplete and OnBeginShutdown](http://support.microsoft.com/kb/302896)
  + [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)COM Add-in Registration](http://support.microsoft.com/kb/302896)
  + [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)How to Build a COM Add-in by Using Visual Basic .NET](http://support.microsoft.com/kb/302896)
  + [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)Step-by-Step Example](http://support.microsoft.com/kb/302896)
* [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)REFERENCES](http://support.microsoft.com/kb/302896)
* [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)Properties](http://support.microsoft.com/kb/302896)
* [[http://support.microsoft.com/library/images/support/kbgraphics/public/en-us/downarrow.gif](http://support.microsoft.com/kb/302896)Give Feedback](http://support.microsoft.com/kb/302896)

**http://support.microsoft.com/library/images/support/en-us/20x20_grey_minus.png**[**SUMMARY**](javascript:void(0);)

Microsoft Office 2000 and later support a new, uniform design architecture for building application add-ins to enhance and to control Office applications. These add-ins are called COM add-ins. This step-by-step article discusses Office COM add-ins and describes how to build an Office COM add-in by using Microsoft Visual Basic .NET.

**The IDTExensibility2 Interface**

A COM add-in is an in-process COM server, or ActiveX dynamic link library (DLL), that implements the **IDTExensibility2** interface as described in the Microsoft Add-in Designer type library (Msaddndr.dll). All COM add-ins inherit from this interface and must implement each of its five methods.

**OnConnection**

The **OnConnection** event fires whenever the COM add-in is connected. The add-in may be connected on startup, by the end user, or through Automation. If **OnConnection** returns successfully, the add-in is said to be loaded. If an error message is returned, the host application immediately releases its reference to the add-in, and the object is destroyed.   
  
**OnConnection** takes the following four parameters:

* *Application* - A reference to the host application object.
* *ConnectMode* - A constant that specifies how the add-in is connected. The add-in can be connected in the following ways:
  + **ext\_cm\_AfterStartup** - The add-in is started by the end user from the **COM Add-ins** dialog box.
  + **ext\_cm\_CommandLine** - The add-in is connected from the command line. Note that this does not apply to building COM add-ins for Office applications.
  + **ext\_cm\_External** - The add-in is connected by an external application through Automation. Note that this does not apply to building COM add-ins for Office applications.
  + **ext\_cm\_Startup** - The add-in is started by the host at application startup. This behavior is controlled by a setting in the registry.
* *AddInInst* - A reference to the **COMAddIn** object that refers to this add-in in the **COMAddIns** collection for the host application.
* *Custom* - An array of **Variant** type values that can hold user-defined data.

**OnDisconnection**

The **OnDisconnection** event fires when the COM add-in is disconnected and just before it unloads from memory. The add-in should perform any cleanup of resources in this event and should restore any changes made to the host application.   
  
**OnDisconnection** takes the following two parameters:

* *RemoveMode* - A constant that specifies how the add-in was disconnected. The add-in can be disconnected in the following ways:
  + **ext\_dm\_HostShutdown** - The add-in is disconnected when the host application closed.
  + **ext\_dm\_UserClosed** - The add-in is disconnected by the end user or an Automation controller.
* *Custom* - An array of **Variant** type values that can hold user-defined data.

**OnAddInsUpdate**

The **OnAddInsUpdate** event fires when the set of registered COM add-ins changes. In other words, whenever a COM add-in is installed or removed from the host application, this event fires.

**OnStartupComplete and OnBeginShutdown**

Both the **OnStartupComplete** and the **OnBeginShutdown** methods are called when the host application has left or is entering a state in which user interaction should be avoided because the application is busy loading or unloading itself from memory. **OnStartupComplete** is only called if the add-in was connected during startup, and **OnBeginShutdown** is only called if the host disconnects the add-in during shutdown.   
  
Because the user interface for the host application is fully active when these events fire, they may be the only way to perform certain actions that otherwise would be unavailable from **OnConnection** and **OnDisconnection**.

**COM Add-in Registration**

In addition to normal COM registration, a COM add-in needs to register itself with each Office application in which it runs. To register itself with a particular application, the add-in should create a subkey, using its ProgID as the name for the key, under the following location:

**HKEY\_CURRENT\_USER\Software\Microsoft\Office\*OfficeApp*\Addins\*ProgID***

The add-in can provide values at this key location for both a friendly display name and a full description. In addition, the add-in should specify its desired load behavior by using a DWORD value called

LoadBehavior

. This value determines how the add-in is loaded by the host application, and is made up of a combination of the following values:

* 0 = Disconnect - Is not loaded.
* 1 = Connected - Is loaded.
* 2 = Bootload - Load on application startup.
* 8 = DemandLoad - Load only when requested by user.
* 16 = ConnectFirstTime - Load only once (on next startup).

The typical value specified is 0x03 (Connected | Bootload).   
  
Add-ins that implement **IDTExtensibility2** should also specify a DWORD value called

CommandLineSafe

to indicate whether the add-ins are safe for operations that do not support a user interface. A value of 0x00 indicates False, and a value of 0x01 indicates True.

**How to Build a COM Add-in by Using Visual Basic .NET**

As mentioned earlier, an Office COM add-in is an in-process COM server that is activated by an Office application through the COM run-time layer. Therefore, developing a COM add-in in Visual Basic .NET requires that the add-in component be implemented in .NET and then exposed to the COM clients (that is, the Office applications) through the COM interop layer.  
  
To create a COM add-in in Visual Basic .NET, follow these steps:

1. In Visual Basic .NET, create a Class Library project.
2. Add a reference to the type library that implements **IDTExtensibility2**. The primary interop assembly for this is already available under the name **Extensibility**.
3. Add a reference to the Microsoft Office object library. The primary interop assembly for this is already available under the name **Office**.
4. Create a public class in the class library that implements **IDTExtensibility2**.
5. After the class library is built, register the library for COM interop. To do this, generate a strong named assembly for this class library and then register it with COM interop. You can use Regasm.exe to register a .NET component for COM interop.
6. Create registry entries so that Office applications can recognize and load the add-in.

You can choose to complete all of these steps, or you can create a .NET project of type **Shared Addin**. This starts the Extensibility Wizard, which helps you to create a COM add-in in .NET.   
  
The Extensibility Wizard creates a Visual Basic .NET class library project along with a **Connect** class that implements the **IDTExtensibility2** interface. The skeleton code that implements the empty members of **IDTExtensibility** is also generated. This project has references to Extensibility and Office assemblies. The build settings of the project have **Register for COM Interop** selected. The assembly key (.snk) file is generated and is referenced in the **AssemblyKeyfile** attribute in Assemblyinfo.vb.   
  
Along with the class library project, the wizard generates a setup project that you can use to deploy the COM add-in on other computers. You may remove this project if desired.

**Step-by-Step Example**

1. On the **File** menu in Microsoft Visual Studio .NET, click **New**, and then click **Project**.
2. In the **New Project** dialog box, expand **Other Projects** under **Project Types**, select **Extensibility Projects**, and then select the **Shared Add-in** template.
3. Type MyCOMAddin as the name of the add-in, and then click **OK**.
4. When the Extensibility Wizard appears, follow these steps:
   1. On page 1, select **Create an Add-in using Visual Basic**, and then click **Next**.
   2. On page 2, select the following host applications, and then click **Next**:
      * Microsoft Word
      * Microsoft PowerPoint
      * Microsoft Outlook
      * Microsoft Excel
      * Microsoft Access
   3. On page 3, provide a name and description for the add-in, and then click **Next**.  
        
      **Note** The name and description of the add-in appear in the **COM Add-in** dialog box in the Office application.
   4. On page 4, select all of the available options, and then click **Next**.
   5. Click **Finish**.
5. Add the following member to the **Connect** class:

Dim WithEvents MyButton As CommandBarButton

1. Implement the code for all the members of **IDTExtensibility2** in the **Connect** class, as follows:
2. Public Sub OnBeginShutdown(ByRef custom As System.Array) Implements Extensibility.IDTExtensibility2.OnBeginShutdown
3. On Error Resume Next
4. ' Notify the user you are shutting down, and delete the button.
5. MsgBox("Our custom Add-in is unloading.")
6. MyButton.Delete()
7. MyButton = Nothing
8. End Sub
9. Public Sub OnAddInsUpdate(ByRef custom As System.Array) Implements Extensibility.IDTExtensibility2.OnAddInsUpdate
10. '
11. End Sub
12. Public Sub OnStartupComplete(ByRef custom As System.Array) Implements Extensibility.IDTExtensibility2.OnStartupComplete
13. Dim oCommandBars As CommandBars
14. Dim oStandardBar As CommandBar
15. On Error Resume Next
16. ' Set up a custom button on the "Standard" command bar.
17. oCommandBars = applicationObject.CommandBars
18. If oCommandBars Is Nothing Then
19. ' Outlook has the CommandBars collection on the Explorer object.
20. oCommandBars = applicationObject.ActiveExplorer.CommandBars
21. End If
22. oStandardBar = oCommandBars.Item("Standard")
23. If oStandardBar Is Nothing Then
24. ' Access names its main toolbar Database.
25. oStandardBar = oCommandBars.Item("Database")
26. End If
27. ' In case the button was not deleted, use the exiting one.
28. MyButton = oStandardBar.Controls.Item("My Custom Button")
29. If MyButton Is Nothing Then
30. MyButton = oStandardBar.Controls.Add(1)
31. With MyButton
32. .Caption = "My Custom Button"
33. .Style = MsoButtonStyle.msoButtonCaption
34. ' The following items are optional, but recommended.
35. ' The Tag property lets you quickly find the control
36. ' and helps MSO keep track of it when more than
37. ' one application window is visible. The property is required
38. ' by some Office applications and should be provided.
39. .Tag = "My Custom Button"
40. ' The OnAction property is optional but recommended.
41. ' It should be set to the ProgID of the add-in, so that if
42. ' the add-in is not loaded when a user clicks the button,
43. ' MSO loads the add-in automatically and then raises
44. ' the Click event for the add-in to handle.
45. .OnAction = "!<MyCOMAddin.Connect>"
46. .Visible = True
47. End With
48. End If
49. ' Display a simple message to show which application you started in.
50. MsgBox("Started in " & applicationObject.Name & ".")
51. oStandardBar = Nothing
52. oCommandBars = Nothing
53. End Sub
54. Public Sub OnDisconnection(ByVal RemoveMode As Extensibility.ext\_DisconnectMode, ByRef custom As System.Array) Implements Extensibility.IDTExtensibility2.OnDisconnection
55. On Error Resume Next
56. If RemoveMode <> Extensibility.ext\_DisconnectMode.ext\_dm\_HostShutdown Then \_
57. Call OnBeginShutdown(custom)
58. applicationObject = Nothing
59. End Sub
60. Public Sub OnConnection(ByVal application As Object, ByVal connectMode As Extensibility.ext\_ConnectMode, ByVal addInInst As Object, ByRef custom As System.Array) Implements Extensibility.IDTExtensibility2.OnConnection
61. MsgBox("On Connection In MyAddin")
62. applicationObject = application
63. addInInstance = addInInst
64. ' If you aren't in startup, manually call OnStartupComplete.
65. If (connectMode <> Extensibility.ext\_ConnectMode.ext\_cm\_Startup) Then \_
66. Call OnStartupComplete(custom)
67. End Sub
68. Private Sub MyButton\_Click(ByVal Ctrl As Microsoft.Office.Core.CommandBarButton, ByRef CancelDefault As Boolean) Handles MyButton.Click
69. MsgBox("Our CommandBar button was pressed!")
70. End Sub

1. Build and test the COM add-in. To do this, follow these steps:
   1. On the **Build** menu, click **Build MyCOMAddin**. Note that building the COM add-in registers the .NET class with the COM interop.
   2. Start one of the Office applications that you selected as host applications for your add-in (for example, Microsoft Word or Microsoft Excel).
   3. After the add-in has started, the **OnConnection** event is fired, and you receive a message box. When you dismiss the message box, the **OnStartupComplete** event fires, and you receive a second message box. Dismiss the message box.
   4. Note that the add-in added a new custom button with the caption "My Custom Button" to the standard toolbar.
   5. Click **My Custom Button**. The **Click** event for the button is handled by the add-in and you receive a message box. Dismiss the message box.
   6. Quit the Office application.
   7. When you quit the application, the **OnBeginShutDown** event fires, and you receive a message box. Dismiss the message box to end the demonstration.

[[http://support.microsoft.com/library/images/support/en-us/uparrow.gif](http://support.microsoft.com/kb/302896#top)Back to the top](http://support.microsoft.com/kb/302896#top) | [Give Feedback](http://support.microsoft.com/kb/302896#survey)

**http://support.microsoft.com/library/images/support/en-us/20x20_grey_minus.png**[**REFERENCES**](javascript:void(0);)

For additional information about writing COM add-ins, click the article number below to view the article in the Microsoft Knowledge Base:

[190253](http://support.microsoft.com/kb/190253/EN-US) INFO: VB6 Designers Do Not Work in VB5

For more information about developing Web-based solutions for Microsoft Internet Explorer, visit the following Microsoft Web sites:

<http://msdn2.microsoft.com/en-us/ie/default.aspx>  
  
<http://support.microsoft.com/iep>

(c) Microsoft Corporation 2001, All Rights Reserved. Contributions by Ranjit R. Sawant, Microsoft Corporation.

[[http://support.microsoft.com/library/images/support/en-us/uparrow.gif](http://support.microsoft.com/kb/302896#top)Back to the top](http://support.microsoft.com/kb/302896#top) | [Give Feedback](http://support.microsoft.com/kb/302896#survey)

**http://support.microsoft.com/library/images/support/en-us/20x20_grey_minus.png**[**Properties**](javascript:void(0);)

Article ID: 302896 - Last Review: December 25, 2006 - Revision: 11.3

**APPLIES TO**

* Microsoft Visual Basic .NET 2002 Standard Edition
* Microsoft Office Excel 2003
* Microsoft Excel 2002 Standard Edition
* Microsoft Office Outlook 2003
* Microsoft Outlook 2002 Standard Edition
* Microsoft Office PowerPoint 2003
* Microsoft PowerPoint 2002 Standard Edition
* Microsoft Office Word 2003
* Microsoft Word 2002

|  |  |
| --- | --- |
| **Keywords:** | kbautomation kbhowtomaster KB302896 |