BEGINNER

GETTING STARTED

Add E-Mail to Your Apps

by Chris Barlow

Encapsulate e-mail functionality into ActiveX DLLs for easy reuse later.

Click & Retrieve Source CODE!

Users have come to expect all Windows applications to feature the ability to send e-mail. If your application creates a data file, you definitely want to provide a menu item that attaches that data file to an e-mail message and sends the message.

Only a few years ago, it was difficult to mail-enable your application. You needed to install an e-mail client, such as MSMail, as well as several Messaging Application Programming Interface (MAPI) DLLs to link to a MAPI service provider, such as a Microsoft Mail (MSMail) post office or Exchange. Today, e-mail is ubiquitous and most computers are set up for e-mail. The interfaces have been simplified, and your Windows Messaging or Outlook client can communicate with your organization's

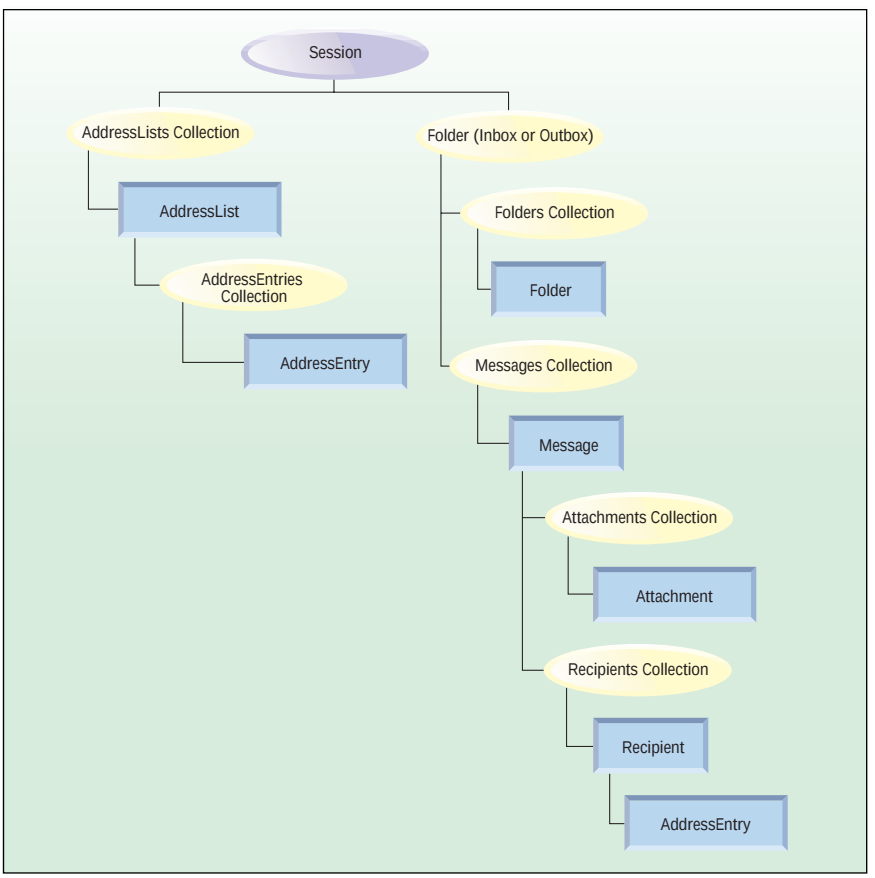

FIGURE 1 ***Simplified Object Model.*** *The object model for the Active Messaging Library is a hierarchical model. In this figure, each indented object is considered a child of the object under which it is indented. An object is the parent of every object at the next level of indention under it. For example, an Attachments collection and a Recipients collection are both child objects of a Message object, and a Messages collection is a parent object of a Message object. However, a Messages collection is not a parent object of a Recipients collection.*

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support and supply chain applications. Chris, who is a frequent speaker at VBITS, Tech•Ed, and DevDays and has been featured in two Microsoft videos, holds degrees from Harvard Business School and Dartmouth College. Reach Chris at ChrisB@SunOpTech.com or through SunOpTech's Web server at www.SunOpTech.com.

Exchange Server through MAPI service providers such as CompuServe, Microsoft Network, and America Online, or through any Simple Mail Transfer Protocol (SMTP)/Post Office Protocol 3 (POP3) mail server.

In previous articles, I suggested you use the MAPI controls that come with Visual Basic, or direct calls to the MAPI DLL, to add e-mail functionality to your apps ["A New Outlook on MAPI," *VBPJ* November 1996]. Now, with the release of Outlook 8.01 and Exchange 5.0, you should use the new Active Messaging 1.1 components released by Microsoft because Active Messaging is easier to use, more flexible, and more powerful. You can download these components from Microsoft's Web site at *ftp://ftp.microsoft.com/services/technet/samples/boes/bo/mailexch/exchange/appfarm/actmsg.exe*.

Active Messaging gives you easy access to sending and receiving e-mail from within a Visual Basic application. It also gives you full access to the folders within your information store. To send e-mail from an application with Active Messaging, start a new Visual Basic 5 project and design a simple form to send e-mail. Make sure you create text-box controls for the recipient, subject, message, and attachment, as well as a CommandButton control labeled Send. Select the References menu item on the Project menu, and add a reference to the Microsoft Active Messaging 1.1 Object Library. Double-click on the Send button and add this code to call the new procedure SendWithAM:

```
Private Sub butSend_Click()
SendWithAM
End Sub
```

Calling a procedure from your button Click events rather than writing the code inline gives you more flexibility to call the procedure from a different place in your application.

The object model for Active Messaging is simple. The Session object is the starting point for all use of the Active Messaging components. Begin an Active Messaging session by creating a MAPI.Session object. All other objects are derived from the Session object (see Figure 1 for a simplified version of the object model from the Active Messaging help file). For example, you can create an outgoing message by adding a Message object to the Messages collection in the

VB5

```
Option Explicit
  Public Profile As String
  Private mLoggedOn As Boolean

  Public Property Get LoggedOn() As Boolean
    LoggedOn = mLoggedOn
  End Property

  Private Sub Class_Initialize()
  On Error Resume Next
  Set oSession = CreateObject("MAPI.Session")
  If oSession Is Nothing Then
    Err.Raise vbObjectError + Err, "SOTMapi", _
       "Create failed"
    Exit Sub
  End If
  End Sub

  Private Sub Class_Terminate()
  oSession.Logoff
  Set oSession = Nothing
  End Sub

  Public Function Logon_
    (Optional sProfile As String = "") As Long
  'first try to logon with passed profile
  On Error Resume Next
  If Len(sProfile) Then
    oSession.Logon ProfileName:=sProfile
    If Err = 0 Then
       mLoggedOn = True
       Profile = oSession.ProfileName
       Exit Function
    End If
  End If
  'then try Profile property
  If Len(Profile) Then
    oSession.Logon ProfileName:=Profile
    If Err = 0 Then
       mLoggedOn = True
       Exit Function
    End If
  End If
  'then try to logon to existing session
  oSession.Logon ShowDialog:=False, NewSession:=False
  If Err = 0 Then
    mLoggedOn = True
    Exit Function
  End If
  'logon failed so try to get default profile
  Profile = GetDefaultProfile
  If Len(Profile) Then oSession.Logon ProfileName:=Profile
  If Err = 0 Then
    mLoggedOn = True
    Exit Function
  End If
  'finally try a normal logon
  oSession.Logon
  If Err = 0 Then
    mLoggedOn = True
    Exit Function
  End If

  Err.Raise vbObjectError + Err, "SOTMapi", "Logon failed"
  Logon = Err
  End Function

  Public Function SendQuick(SendTo$, Subject$, Optional _
    Message$ = "", Optional Attach$ = "") As Long
  Dim oMessage As Message
  On Error Resume Next
  If Not mLoggedOn Then Logon
  If Err Then Exit Function
  Set oMessage = oSession.Outbox.Messages.Add
  With oMessage
    .Subject = Subject
    .Text = " " & Message
  End With
  With oMessage.Recipients.Add
    .Name = SendTo
    .Type = mapiTo
    .Resolve
  End With
  If Len(Attach) Then
    With oMessage.Attachments.Add
       .Position = 1
       .Type = mapiFileData
       .Name = Attach
       .ReadFromFile Attach
    End With
  End If
  oMessage.Send
  SendQuick = Err
  End Function
```

LISTING 1 ***MAPIClass Class Module.*** *This class module contains the properties and methods for the new class. Design the Logon method to perform a "quiet" logon—first, by using a given profile name from the class's Profile property, then by trying to log on to an existing session, and finally by finding the default profile from the registry.*

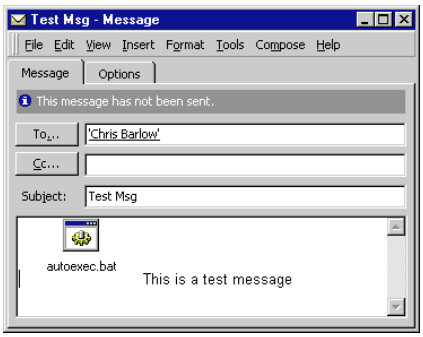

FIGURE 2 ***Send the Message.*** *This is how your message looks in your Outbox after you send it with these procedures. Notice that the icon for the attachment is located in the first position of the mail message text.*

Outbox folder.

Add code to dimension a Session object and a Message object, create the MAPI session, and call the Logon method of the Session object:

```
Private Sub SendWithAM()
Dim oSession As Object
Dim oMessage As Message
On Error Resume Next
Set oSession = _
   CreateObject("MAPI.Session")
If oSession Is Nothing Then
   Exit Sub
End If
oSession.Logon
If Err Then Exit Sub
```

Add a new outgoing message object, and set the Subject and Text properties from the text-box controls:

```
Set oMessage = _
   oSession.Outbox.Messages.Add
With oMessage
   .Subject = txtSubject
   .Text = " " & txtNoteText
End With
```

Add a Recipient object to this message, and set the Name and Type properties. Your e-mail message can have many recipients. The Type property controls whether they appear in the To: or the CC: section of the message. You can use the With...End With statement to work with the Recipient object without having to dimension a variable to hold the Recipient object. Call the Resolve method to change the text name to a valid e-mail address. If you enter an ambiguous name, a dialog appears to resolve the name manually:

```
With oMessage.Recipients.Add
   .Name = txtTo
   .Type = mapiTo
   .Resolve
End With
```

If you enter a file path in the Attachment text-box control, add an Attachment object to the message and set the Position, Type, and Name properties. The Position property determines where to place the icon for the attached file in the text portion of the message. Use the ReadFromFile method to read the file contents as an attachment:

```
If Len(txtAttach) Then
   With oMessage.Attachments.Add
      .Position = 1
      .Type = mapiFileData
      .Name = txtAttach
      .ReadFromFile txtAttach
   End With
End If
```

Finally, send the message and log off the session (see Figure 2):

```
oMessage.Send
oSession.Logoff
End Sub
```

This is all you need to do to mail-enable your application, but let's go a step further. Messaging code has changed dramatically over the past few years and will undoubtedly continue to change. Rather than write an application that calls these properties and methods explicitly, why not encapsulate this functionality into an ActiveX DLL with a simple class your applications can call? This way, you won't need to change your applications the next time messaging changes because changing the DLL will upgrade your application.

ACTIVE MESSAGING CLASS

Select the Add Project menu item from the Project menu and choose ActiveX DLL. Right-click on the class module to display the Properties tab, change the name of the class to MAPIClass, and change Instancing to MultiUse. When creating any class, you need to plan the properties and methods of the class. With this simple example class, you only need two properties and two methods.

The LoggedOn property is True if the class is logged onto a messaging session.

This should be a read-only property of the class, so you need a private variable to hold the value and a Property Get procedure to return the value. You can use the Profile property to specify the profile to use when logging on. Because the application using the class sets the Profile property, the property can be a simple public string variable. The Logon method calls the Session object's Logon method, but has the capability to find the default profile if one is not specified. The SendQuick method accepts the message's recipient, subject, text, and attachment as arguments, and provides a single simple call to send a message with no more than one attachment to a single recipient.

You might find that you want to expand this class to allow for multiple recipients and multiple attachments, but let's start with this simple class for now. Dimension the public Profile property and the private mLoggedOn variable in the class module. Write the Property Get procedure to return the value of this private variable:

```
Option Explicit
Public Profile As String
Private mLoggedOn As Boolean

Public Property Get LoggedOn() As _
   Boolean
   LoggedOn = mLoggedOn
End Property
```

Create the MAPI Session object in the class's Initialize event, and log off the session in the Terminate event:

```
Private Sub Class_Initialize()
On Error Resume Next
Set oSession = _
   CreateObject("MAPI.Session")
If oSession Is Nothing Then
   Err.Raise vbObjectError + Err, _
      "SOTMapi", "Create failed"
   Exit Sub
End If
End Sub

Private Sub Class_Terminate()
oSession.Logoff
Set oSession = Nothing
End Sub
```

Because the SendQuick method is similar to the SendWithAM you wrote, copy that code and paste it into your class module within the public SendQuick method. Modify the code to check the value of the mLoggedOn variable, and call the Logon method if the session is not logged on. The message will be sent as before, except you need to substitute the method's arguments for the text-box controls (see Listing 1).

You should design the Logon method to perform a "quiet" logon—first, by using a given profile name from the class's Profile property, then by trying to log on to an existing session, and finally by finding the default profile from the registry. This last step can be a bit complicated. Article Q171422 in the Microsoft Knowledge Base (www.microsoft.com/kb) provides some good sample code that I included in the GetDefaultProfile procedure in the class's module. Download Listing 2 from the free, Registered Level of The Development Exchange (see the Code Online box for details). The class should display a dialog box only if these quiet attempts fail (see Listing 1).

Now build your ActiveX DLL, and change your test application to call your new class by adding the SendMailClass procedure to the form and changing the code in the Send button's Click event to call the new procedure:

```
Private Sub SendWithClass()
Dim MyMapi As New MAPIClass
On Error Resume Next
If MyMapi.SendQuick(txtTo, txtSubject, _
   txtNoteText, txtAttach) = 0 Then
   If Err Then
      MsgBox "Send Failed " & Err
   Else
      MsgBox "Message Sent"
   End If
Else
   MsgBox "Send Failed"
End If
End Sub
```

Notice that your app doesn't need to set the profile or call the Logon method explicitly because the SendQuick method calls the Logon method if the session is not logged on. You've now wrapped the Active Messaging library with your own class. If all your applications use your class, it will be easy for you to maintain compatibility as messaging continues to evolve. ×

Code Online

You can find all the code published in this issue of VBPJ on The Development Exchange (DevX) at http://www.windx.com. For details, please see "Get Extra Code in DevX's Premier Club" in Letters to the Editor.

Add E-Mail to Your Apps

Locator+ Codes

Listings ZIP file plus Listing 2 (free Registered Level): VBPJ1197

Listings for this article plus Listing 2 and the VB files that let you mail-enable your application (subscriber Premier Level): GS1197P