Connecting HTML Help to Visual Basic Programs

by Don Lammers

Last updated 2000-10-23

Presented with author's permission by

Shadow Mountain Tech

Contents

Scope

If you are trying to call HTML Help from Visual Basic I generally recommend using David Liske's Help subclassing module and tutorial available at http://mvps.org/htmlhelpcenter. It makes life a lot easier, lets you do things that you could not do otherwise (like use Training Card Help) and works with both Visual Basic 5 and Visual Basic 6. It comes with its own tutorial, so I will not attempt to explain it here.

However, it always helps to understand how the original was designed to work before trying to override its functionality. Or you may have a project that does not need any more functionality than is provided by Microsoft. This document is intended to show you how to hook HTML Help to Visual Basic 6 and later *without* using add-ons. Earlier versions of Visual Basic will require you to make all calls directly to the HTML Help API or to use an add-on like David Liske's subclassing module.

Acknowledgements

The following information is from my own experimentation and from people who have walked this path before me. Thanks to all of the following:

David Liske, Dana Cline, Microsoft HTML Help WorkShop Help, Microsoft Knowledge Base, *The Developer's Guide to WINHELP.EXE* (Jim Mischel), *Building Windows 95 Help* (Nancy Hickman), Paul O'Rear, *Developing Online Help for Windows 95* (Boggan, Farkas, and Welinske), Gordon F. MacLeod, Burt Abreu.

Connecting Context Sensitive HTML Help to Visual Basic

Visual Basic (versions 6 and later) provides directly for calling window level HTML Help topics using the F1 key, **or** providing What's This? Help for each control, including activation from the F1 key. The choice between What's This? Help and form level F1 help can be made on a form by form basis.

Only version 6 and later of Visual Basic recognize .chm files as a valid help file format. For any earlier versions of Visual Basic, you must make all HTML Help calls directly to the HTML Help API.You also cannot use the Common Dialog control to call HTML Help. No current version of this control recognizes .chm files a help files.

This section shows the steps necessary to hook context sensitive HTML Help into Visual Basic.

Set the Help File Name

Before using the built in context sensitive help features of Visual Basic, you need to set the App.HelpFile property so that the program knows the help file to call and the window in which WinHelp should display the topic.

To set the help file name:

- Set the App.HelpFile property. The basic syntax is:

  ```
  App.HelpFile = App.Path & "\helpfile.hlp>hWindow"
  ```

  helpfile.hlp is the name of the WinHelp file.

  hWindow is the name of the secondary window in which to display the topics. If not specified, the topic is displayed in the main window.

Tips:

- If you are calling both secondary windows and pop-ups from your application and only have a single help file, you may want to skip the secondary window name in this property and append it to App.HelpFile only when you call help that requires a secondary window.
- The App.HelpFile property can be changed at runtime, so you can reset it in code as needed to call additional help files.
- If you have more than one help file or window style to call from the program you should put the code for changing help file names and windows in a separate subroutine. This makes any changes during the project much easier to implement.
- You can let the help author edit the help file name and window specification by putting them in a separate text file. Setting this file up in INI file format lets you use standard INI file commands to access this "text database."

Make Sure WinHelp Can Find the Help File

Make sure that WinHelp can always find the help file. Since your program can be started from shortcuts created by the user, the installation program should register the help file or you should call help from your program using the full path and file name.

To ensure that the program and WinHelp can always find the help file:

- Register the help file by creating a string value under HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\HTMLHelp. The name of the string value is the help file name (without path) and the value is the path.
- Always use the full path and file name when setting App.HelpFile, even if the help file is in the same folder as the program executable.

Connect the F1 key

Visual basic uses the F1 key as one means to trigger help. For each form in your project, you can connect either form level or control level help to the F1 key, or turn the F1 key off completely.

To disconnect F1 help from a form:

1. Set the form's WhatsThisHelp property to False. On forms that support the WhatsThisButton property it will also be set to False. MDI forms do not support the WhatsThisButton property.
2. Set the form's HelpContextID property to 0 (zero).
3. Set the control's WhatsThisHelpID property to 0 (zero) for all controls in the form.

To connect form level help to a form:

1. Set the form's WhatsThisHelp property to False. On forms that support the WhatsThisButton property it will also be set to False. MDI forms do not support the WhatsThisButton property.
2. Set the form's HelpContextID property. This number must match the topic ID in the [MAP] section of the HTML Help project (.hhp) file. The topic will be opened in the window specified in App.HelpFile, or in the default window.

To connect control level help (also called What's This? Help) to a form:

1. Set the form's WhatsThisHelp property to True.
2. For each control with a What's This? Help topic, set the WhatsThisHelpID. This number must match the topic ID in the [MAP] section of the HTML Help project (.hhp) file.
3. To turn off What's This Help for a single control, set the control's WhatsThisHelpID to 0 (zero).

Notes:

- Once you put a topic number in the WhatsThisHelpID property of a control, the KeyDown, KeyUp, and KeyPress events of the control are no longer triggered by the F1 key. You can detect any other key press in these events, but not F1.
- The WhatsThisHelp property for a form cannot be changed at runtime. The only way to turn What's This? Help on or off at run time is to set and clear the WhatsThisHelpID for the control.

Activate the What's This? Help Button in a Dialog Box Title Bar

If the form is not resizeable, you can put a ? in the title bar to activate What's This? Help mode. If the form is resizeable you must handle this function using a separate button. See *Connect What's This? Help to Resizeable Windows*.

To activate the What's This? Help button in the title bar:

1. Set the form's BorderStyle property to Fixed Single or Fixed Dialog.
2. Set the form's WhatsThisButton property to True. This also sets the WhatsThisHelp property to True.
3. For each control with a What's This? Help topic, set the WhatsThisHelpID. This number must match the topic ID in the [MAP] section of the HTML Help project (.hhp) file.

Note:

- You cannot have a What's This? Help button in the title bar of any resizeable window, or any window set to the Fixed ToolWindow style.

Connect What's This? Help to Resizeable Windows

The following is only necessary to activate What's This? Help in a resizeable window or in a window set to the Fixed ToolWindow style. Otherwise if the form is not resizeable you can have Visual Basic put a ? in the title bar to activate What's This? Help mode (see *Activate the What's This? Help Button in a Dialog Box Title Bar*).

1. Set the form's WhatsThisHelp property to True.
2. Create a What's This? Help button for the toolbar that can be toggled.
3. In the MouseDown event of What's This? Help button, place any code necessary for putting the button in it's "down" state.
4. In the MouseUp or Click event of What's This? Help button, place the following code: Visual Basic does not correctly trigger What's This? Help mode if you put this code in the MouseDown event.

```
'**Set What's This? Help mode
WhatsThisMode
'**Toggle the button state if necessary.
'**Any commands placed here will not be executed until
'**What's This? Help mode is released.
....Code for setting the What's This? Help button to "up"
```

   Notes:

   - This activates What's This? Help mode for visible child forms, but not for any other non-modal forms that may be open (for instance, tool windows).
   - Any code in the subroutine after you call the WhatsThisMode method will not be executed until What's This? Help mode is released by Visual Basic.
   - The next mouse down event will be processed by the What's This? Help logic without triggering an event you can intercept.
   - If you put this code in the Click event you can call the button Click subroutine from the What's This? Help menu item without worrying about parameters.
5. For each control with a What's This? Help topic, set the WhatsThisHelpID. This number must match the topic ID in the [MAP] section of the HTML Help project (.hhp) file.
6. In the Help menu, create an item called What's This? Help and assign SHIFT-F1 as the shortcut. In the Click event for this menu item, call the What's This? button Click or Mouse Up event where you put the WhatsThisMode command.
7. In the MouseUp or Click event of What's This? Help button, activate What's This? Help mode for any non-child non-modal form (such as a tool) that you want included in the What's This? Help logic. These forms must have a WhatsThisHelpID set for each control, and their WhatsThisHelp property set to True. The code would look something like this:

```
'**Set What's This? Help mode for tools that are open
If mnu_ColorWheel.Checked = True then
  FrmTool1.WhatsThisMode
End If
```

Tips:

- You can trigger What's This? Help mode programmatically from any control that offers a MouseUp or Click event.
- In C++, a menu item and equivalent toolbar item usually have the same control ID and thus the same context ID (though this can be overridden). In Visual Basic you can easily set the WhatsThisHelpID for each of these separately.

Implement F1 "Selected Text" Help

In any control where the user can highlight text, you can use this text to call an appropriate help topic. Because What's This? Help and F1 help are so closely tied together in Visual Basic, you cannot use F1 in this matter and still have an active What's This? Help button without additional programming. In many cases this kind of help is provided in child windows which would not have other forms of context sensitive help anyway. For instance, in Visual Studio Microsoft does not provide What's This? Help for the main windows but does provide "selected text" help. Mixing methods is discussed in the next section.

1. Set the control's WhatsThisHelpID property to 0 (zero).
2. Intercept the F1 KeyDown event. To do this, put the following code in the KeyDown event for the control:

```
'**Check for no highlighting
If frmChild1.Text1.selText = "" Then
  '**Nothing highlighted. We don't do anything.
  KeyCode = 0
ElseIf KeyCode = vbKeyF1 And Shift = 0 Then
  Dim selText$
  '**Strip spaces (if any)
  selText$ = Trim$(frmChild1.Text1.selText)
  ...any other code for determining the highlighted words
  '**Force the help file open to insure that the
  '**HELP_COMMAND command always work.
  '**The help file is forced open to the default topic,
  '**which is the "Keyword Not Found" topic, in the
  '**same secondary window that is normally used.
  success = WinHelp(hwnd, App.HelpFile, HELP_FORCEFILE, ByVal 0&)
  '**Set up the string for calling the KLink macro. We first need
  '**to use the TestKLink macro to see if there is a matching
  '**keyword. Note that the help compiler converts KLink to KL
  '**TestKLink to KL with the test parameter. Since this string
  '**is not going throught the help compiler, we need use the
  '**short form of the macro.
  HelpCommand$ = "IF(KL(`" & selText$ & "', 4, `', `'),`KL(`" &
selText$ & "', 1, `', `')')"
  '**Call the TestKLink macro and KLink macros.
  success = WinHelp(hwnd, App.HelpFile, HELP_COMMAND, ByVal
HelpCommand$)
else
  ...any other key trapping here
End If
```

Mixing What's This? Help with F1 "Selected Text" Help

Mixing standard What's This? Help with "selected text" help requires a bit of additional programming. The main reason is that the WhatsThisHelp property is cannot be changed at run time. To disconnect F1 help from What's This? Help requires that the WhatsThisHelpIDs for controls on a form be set to zero except when the form is in What's This? Help mode.

1. Create a subroutine to store the WhatsThisHelpID property for each control in its Tag property on startup.

```
Public Sub LoadTagsWithContextID(thisForm as Form)
  '**This just makes sure that the Tag property and the WhatsThisHelpID
  '**property are synchronized.
  For Each Control In thisForm.Controls
    Select Case LCase$(TypeName(Control))
    Case "commandbutton", "image", "picturebox"
      Control.Tag = Format(Control.WhatsThisHelpID, "General Number")
    Case "menu"
    End Select
  Next Control
End Sub
```

2. Create a subroutine to set and clear the What's This? Help status for the form:

```
Public Sub SetWhatsThisStatus(thisForm As Form, status%)
  '**This subroutine does the dirty work of copying the values
  '**from the Tag property ot erasing the WhatsThisHelpID.
  Dim fControls As Control
  Select Case status%
  Case 0
    '**No longer in What's This? Help. Erase all IDs
    For Each Control In thisForm.Controls
      Select Case LCase$(TypeName(Control))
      Case "commandbutton", "image", "picturebox", "textbox"
        Control.WhatsThisHelpID = 0
      Case "menu"
      End Select
    Next Control
    useWhatsThisHelp% = 0
  Case Else
  '**Going into What's This? Help. Copy IDs from Tag property
    For Each Control In thisForm.Controls
      Select Case LCase$(TypeName(Control))
      Case "commandbutton", "image", "picturebox", "textbox"
        Control.WhatsThisHelpID = Val(Control.Tag)
      Case "menu"
      End Select
    Next Control
    useWhatsThisHelp% = 1
  End Select
End Sub
```

3. Activate What's This? Help mode.

   For this example we will assume that the button will handle its own state change on MouseDown and MouseUp. If you are using a button without separate bitmaps for the MouseDown and MouseUp states, you will also have to add code to switch the button bitmap appropriately.

```
Private Sub btn_WhatsThisHelp_Click()
   '**Set WhatsThisHelpID values from the Tag property.
   Call SetWhatsThisStatus(frmMain, 1)
   Call SetWhatsThisStatus(frmChild1, 1)
   '**Set What's This? Help mode.
   WhatsThisMode
   '**Reset the WhatsThisHelp IDs to 0 (zero). These commands will
   '**not be activated until WhatsThisHelp mode is cleared.
   Call SetWhatsThisStatus(frmMain, 0)
   Call SetWhatsThisStatus(frmChild1, 0)
End Sub
```

Connect Context Help to Menu Items

When you create each menu item, include the context ID in the Menu Editor's HelpContextID box. When there is a context ID in this box, Visual Basic will open that topic in the secondary help window specified by App.HelpFile when you highlight the menu item and press F1.

Connect Context Help to a Right Click on a Button

The traditional way of connecting help to a right click on a control is to provide a pop-up menu with an item named What's This?. When the user clicks this menu item, help opens to the appropriate topic. the basic steps in creating such a link are:

1. In the control's Click event, save the topic number to call.
2. Implement whatever code is needed to open the pop-up menu with the appropriate items listed.
3. In the Click event for the What's This? menu item, call the HTML Help API with the appropriate topic number.

Tip:

- Unless there are only a few controls that do not implement a right click menu, I recommend calling the topic directly if the user right clicks a button that does not otherwise have a right click menu. A pop-up menu with a single What's This? item looks a bit strange, and this saves the user one click.

Calling HTML Help Directly from Your Visual Basic Code

You can programmatically call help from anywhere in your code. The most common reasons for doing this would be to open the help Table of Contents from the Help menu or to call a general help topic from a Help button on a form, but you can also call help using keywords.

HTML Help API Syntax

The **HtmlHelp** function starts HTML Help and passes additional data indicating the nature of the help requested by the application. To use the following API calls you must attach HTMLHelp.bas or HTMLHelp.cls (see *References* for where to get these).

```
BOOL HtmlHelp( HWND hWndMain, LPCTSTR lpszHelp,
    UINT uCommand, DWORD dwData )
```

If the function succeeds, the return value is nonzero. If the function fails, the return value is zero.

hWndMain is the handle of the window requesting help. The **HtmlHelp** function uses this handle to keep track of which applications have requested help. If the *uCommand* parameter specifies HELP_CONTEXTMENU or HELP_WM_HELP, *hWndMain* identifies the control requesting help.

lpszHelp is a string containing the path, if necessary, and the name of the help file that HTML Help is to display. For some commands it can also contain the file name of the specific topic. The format for this string is:

```
Helpfile.chm[::/Topic.htm][>Window name]
```

Note: The filename may be followed by an angle bracket (>) and the name of a secondary window if the topic is to be displayed in a secondary window rather than in the primary window. The name of the secondary window must have been defined in the [WINDOWS] section of the help project (.hhp) file.

uCommand specifies the type of help requested. For a partial list of possible values and how they affect the value to place in the *dwData* parameter, see *Help Command Constants* below. For a full list, see the *Microsoft HTML Help Workshop Help*.

dwData specifies additional data. The value used depends on the value of the *uCommand* parameter. For a partial list of possible values, see *Help Command Constants* below. For a full list, see the *Microsoft HTML Help Workshop Help*.

HtmlHelp Command Constants

The following list shows the possible values for the *uCommand* parameter, its value, its action, and the corresponding formats of the *dwDat*a command. For details about each of the structures referred to in this section, see *Help Command Structures*.

HH_DISPLAY_TOPIC (0 or &H0) selects the **Contents** tab in the navigation pane of the HTML Help viewer (tri-pane window) and optionally displays a specified topic. If the topic is not specified in *lpszHelp* then *dwData* is a string containing the file name of the topic to open (including relative folder information if necessary). Otherwise *dwData* is 0& (NULL). If *dwData* is 0, you can specify the topic and/or defined window in *lpszHelp* using the following format:

```
Helpfile.chm[::/Topic.htm][>Window name]
```

HH_DISPLAY_TOC (1 or &H1) is the same as HH_DISPLAY_TOPIC.

HH_DISPLAY_INDEX (2 or &H2) selects the **Index** tab in the navigation pane of the HTML Help viewer (tri-pane window) and searches for a keyword. *dwData* is a string containing keyword to search for.

HH_DISPLAY_SEARCH (3 or &H3) selects the **Search** tab in the navigation pane of the HTML Help viewer (tri-pane window) and performs a search. *dwData* an HH_FTS_QUERY structure that specifies the search parameters.

HH_KEYWORD_LOOKUP (13 or &HD) searches for one or more keywords in a compiled HTML Help (.chm) file. *dwData* is an HH_AKLINK structure containing the keywords to search for and the actions to be taken if no matches are found.

HH_DISPLAY_TEXT_POPUP (14 or &HE) opens a pop-up window and displays one of the following:

- An explicit text string.
- A text string based on a resource ID
- A text string based on a text file contained in a compiled HTML Help (.chm) file.

To use an explicit text string or a string based on a resource identifier, set *lpszHelp* to an empty string. To use a text string from a file contained in a compiled HTML Help (.chm) file, set *lpszHelp* to the name of the .chm file and the text file within the .chm file. In either case *dwData* is an HH_POPUP structure.

HH_HELP_CONTEXT (15 or &HF) displays the topic identified by the specified context identifier defined in the [MAP] section of the .HPJ file. *dwData* is an unsigned long integer containing the context identifier for the topic.

HH_CLOSE_ALL (18 or &H12) closes all windows that have been opened directly or indirectly by the calling program. *hWndMain* must be 0 (zero).*lpszHelp* must be an empty string. *uCommand* must be 0 (zero).

HH_ALINK_LOOKUP (19 or &H13) searches for one or more Associative link (Alink) names in a compiled HTML Help (.chm) file. *dwData* is an HH_AKLINK structure containing the names to search for and the actions to be taken if no matches are found.

Additional commands are available through the HtmlHelp API. See the *HTML Help Workshop Help* for descriptions of these additional commands.

Help Structures

The following list shows the data that goes into each of the structures referred to in the *Help Command Constants* section.

The HH_POPUP structure is defined as follows:

```
Public Type tagHH_POPUP
  cbStruct As Long
  hinst As Long
  idString As Long
  pszText As String
  pt As POINTAPI
  clrForeground As ColorConstants
  clrBackground As ColorConstants
  rcMargins As RECT
  pszFont As String
End Type
```

cbStruct is an Long variable which specifies the size of the structure. You must always fill in this value before passing the structure.

hinst is a Long variable containing the instance handle for the string resource.

idString is a Long variable specifying the string resource ID, or the text ID if *pszFile* is specified in the HtmlHelp call.

pszText is the explicit string to display. To display this string you must set *idString* to 0 (zero).

pt is a POINTAPI structure that specifies the top center of the popup window.

clrForeground specifies the foreground (text) color used in the pop-up. Use a VB color constant or a number in the format &HBBGGRR.

clrBackground specifies the background color used in the pop-up. Use a VB color constant or a number in the format &HBBGGRR.

rcMargins is a RECT structure indicating the amount of space between edges of window and text. Use -1 for each member to ignore.

pszFont is a string containing the font to use in the pop-up. The string is in the following format:

```
facename[, point size[, char set[, BOLD ITALIC UNDERLINE]]]
```

You can skip an attribute by entering a comma for the attribute.

Note:

- This structure contains standard Windows POINTAPI and RECT structures, which must be defined in the project. You can get these structures from the API Text Viewer that came with Visual Basic.

The HH_AKLINK structure is defined as follows:

```
Public Type HH_AKLINK
  cbStruct          As Long
  fReserved         As Boolean
  pszKeywords       As String
  pszUrl            As String
  pszMsgText        As String
  pszMsgTitle       As String
  pszWindow         As String
  fIndexOnFail      As Boolean
End Type
```

cbStruct is an Long variable which specifies the size of the structure. You must always fill in this value before passing the structure.

fReserved is reserved. Set to False.

pszKeywords is a string containing the keywords to search for. Separate multiple keywords with simicolons.

pszURL is a string containing the name of the topic file to show if no matches are found. Use the following format for this string:

```
Helpfile.chm[::/Topic.htm][>Window name]
```

pszMsgText is a string containing the message box text to display if *fIndexOnFail* is False and *pszURL* is an empty string.

pszMsgTitle is a string containing the caption of the message box which will be displayed if *fIndexOnFail* is False and *pszURL* is an empty string.

pszWindow is a string containing the name of the secondary help window in which to display one of the following:

- The selected topic if the lookup yields one or more matching topics.
- The topic specified in *pszURL* if the lookup fails and a topic is specified in *pszURL*.
- The **Index** tab if the lookup fails and *fIndexOnFail* is True.

fIndexOnFail is True if you want to display the keyword in the Index tab of the HTML Help viewer (tri-pane window) if the lookup fails

The HH_FTS_QUERY structure is defined as follows:

```
Public Type HH_FTS_QUERY
  cbStruct          As Long
  fUniCodeStrings   As Long
  pszSearchQuery    As String
  iProximity        As Long
  fStemmedSearch    As Long
  fTitleOnly        As Long
  fExecute          As Long
  pszWindow         As String
End Type
```

cbStruct is an Long variable which specifies the size of the structure. You must always fill in this value before passing the structure.

fUniCodeStrings is a Long variable which is non-zero if all strings are Unicode.

pszSearchQuery is a string containing the search query.

iProximity is a Long variable which specifies word proximity.

fStemmedSearch is a Long variable which is non-zero if you want a stemmed search.

fTitleOnly is a long variable which is non-zero if you want to search titles only.

fExecute is a long variable which is non-zero to initiate the search.

pszWindow is a string which specifies the window in which to display the results.

HTML Help API Examples

In the examples below:

- "frmMain.hWnd" is the hWnd for the main form in your program.
- Its assumed that you have set the full path and file name for the help file in App.HelpFile.
- "TopicID" is the ID for the desired topic.

To display the Contents tab in the navigation pane and open the default topic:

```
HtmlHelp(frmMain.hWnd, App.HelpFile, HH_DISPLAY_TOC, ByVal 0&)
```

To display the Contents tab in the navigation pane and open a specific topic:

```
HtmlHelp(frmMain.hWnd, App.HelpFile & \
        "::/TopicFileName.htm", HH_DISPLAY_TOC, ByVal 0&)
```

-OR-

```
HtmlHelp(frmMain.hWnd, App.HelpFile, \
        HH_DISPLAY_TOC, ByVal "TopicFileName.htm")
```

To display a specific topic in a standard window:

```
HtmlHelp(frmMain.hWnd, App.HelpFile, HH_HELP_CONTEXT, \
        ByVal CLng(TopicID))
```

To display a specific topic in a popup:

```
HtmlHelp(frmMain.hWnd, App.HelpFile, HH_HELP_CONTEXTPOPUP, \
        ByVal CLng(TopicID))
```

To display the Index tab in the navigation pane:

```
HtmlHelp(frmMain.\
         hWnd, App.HelpFile, HH_DISPLAY_INDEX, "")
```

To display the Search tab in the navigation pane:

```
dim searchIt as HH_FTS_QUERY
With searchIt
  .cbStruct = len(searchIt)
  .fUniCodeStrings = 0&
  .pszSearchQuery = ""
  .iProximity = 0&
  .fStemmedSearch = 0&
  .fTitleOnly = 0&
  .fExecute = 0&
  .pszWindow = ""
End With
HtmlHelp(hWnd, App.HelpFile, HH_DISPLAY_SEARCH, searchIt)
```

To display a topic or topics using a K Keyword:

```
dim keyData as HH_AKLINK
With keyData
  .cbStruct = len(keyData)
  .fReserved = 0&
  .pszKeywords = "cats"
  .pszUrl = ""
  .pszMsgText = "No such keyword found"
  .pszMsgTitle = "Keyword Not Found"
  .pszWindow = ""
  .fIndexOnFail = 0&
End With
HtmlHelp(hWnd, App.HelpFile, HH_KEYWORD_LOOKUP, keyData)
```

To display a topic or topics using an A Keyword:

```
dim keyData as HH_AKLINK
With keyData
  .cbStruct = len(keyData)
  .fReserved = 0&
  .pszKeywords = "cats"
  .pszUrl = ""
  .pszMsgText = "No such associative link found"
  .pszMsgTitle = "Link Not Found"
  .pszWindow = ""
  .fIndexOnFail = 0&
End With
HtmlHelp(hWnd, App.HelpFile, HH_ALINK_LOOKUP, keyData)
```

To force all HTML Help windows closed that were opened by this app:

```
HtmlHelp(0&, "", HH_CLOSE_ALL, ByVal 0&)
```

Using the Common Dialog Control to Call HTML Help

You cannot use the Common Dialog control to call HTML Help in any Visual Basic version. It does not recognize .chm files as a valid help file format.

Training Cards from Visual Basic

You cannot implement training cards from a Visual Basic application in its native form, since it does not support message loops. However, David Liske has created a subclassing module that will let you accomplish this.

Resources

All of the following plus links to other reference sites about online user assistance are available through the Shadow Mountain Tech Web site (www.smountain.com).

On the Help/Connecting page:

- Latest update of this document and other documents about connecting online help to your program
- *Programmer's Reference to WinHelp* (by Don Lammers and Paul O'Rear)
- Sample code, including API definition modules.
- David Liske's help subclassing tutorial and modules

On the Help/Bugs & FAQ page:

- *WinHelp 4.0 Unofficial Bug and Anomaly List* (currently maintained by Don Lammers)
- *WinHelp FAQ File* (by Charlie Munro)