Using HTML Help With Microsoft Access

Updated February 26, 2003

Dave Liske
Delmar Computing Services, Tecumseh, Michigan
http://www.mvps.org/htmlhelpcenter

Access 97 was released about the same time as HTML Help, so there is zero support for this particular help system from Access 97. For example, you won't be able to use the help functionality of the MsgBox function with HTML Help as it only works with WinHelp. The way to fix this particular problem is to use Sönke Huckfeldt's techniques for creating custom MsgBox and InputBox functions. These techniques can be found in a sample database which is available from the same page you retrieved this article from.

You can snag code out of the HTML Help class module to call topics from a menu, command buttons, etc., in Access 97 and 2000. However, there's a problem with this in that Access doesn't use App.Path to derive the path to the application as in the class module code. Microsoft Access MVP Doug Steele suggests using the following to replace App.Path:

CurrentDB.Name will give you the fully qualified path to the .MDB: to convert that to the path only, use

```
Left$(CurrentDB.Name, Len(CurrentDB.Name) - Len(Dir(CurrentDB.Name)))
```

or, more efficiently:

```
Dim strFullPath As String
Dim strPathOnly As String

strFullPath = CurrentDB.Name
strPathOnly = Left$(strFullPath, Len(strFullPath) -
Len(Dir(strFullPath)))
```

Another method is to register your HTML Help file name and path in HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\HTML Help. Once you do this, calling a help topic uses the name of the file only. The HTML Help engine will check the registry for the path. Here's how to open an HTML Help file to a topic this way, where 1030 is the topics mapped ID:

```
Private Const HH_HELP_CONTEXT = &HF

Private Declare Function HTMLHelpStdCall Lib "hhctrl.ocx" _
    Alias "HtmlHelpA" (ByVal hwnd As Long, _
    ByVal lpHelpFile As String, _
    ByVal wCommand As Long, _
    ByVal dwData As Long) As Long

Private Sub cmdTopicID_Click()

  HTMLHelpStdCall 0, "myhelp.chm", HH_HELP_CONTEXT, 1030

End Sub
```

Creating menus isn't so easy. Drop the following code into a module:

```
Private Const HH_DISPLAY_TOC = &H1
Private Const HH_DISPLAY_INDEX = &H2
Private Const HH_DISPLAY_SEARCH = &H3

' UDT for accessing the Search tab
Private Type tagHH_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

Private Declare Function HTMLHelpStdCall Lib "hhctrl.ocx" _
    Alias "HtmlHelpA" (ByVal hwnd As Long, _
    ByVal lpHelpFile As String, _
    ByVal wCommand As Long, _
    ByVal dwData As Long) As Long

Private Declare Function HTMLHelpCallSearch Lib "hhctrl.ocx" _
    Alias "HtmlHelpA" (ByVal hwnd As Long, _
    ByVal lpHelpFile As String, _
    ByVal wCommand As Long, _
    ByRef dwData As tagHH_FTS_QUERY) As Long

Public Function ShowContents() As Long

  ShowContents = HTMLHelpStdCall(0, "myhelp.chm", HH_DISPLAY_TOC, 0)

End Function

Public Function ShowIndex() As Long

  ShowIndex = HTMLHelpStdCall(0, "myhelp.chm", HH_DISPLAY_INDEX, 0)

End Function

Public Function ShowSearch() As Long

  Dim HH_FTS_QUERY As tagHH_FTS_QUERY

  With HH_FTS_QUERY
    .cbStruct = Len(HH_FTS_QUERY)
    .fUniCodeStrings = 0&
    .pszSearchQuery = ""
    .iProximity = 0&
    .fStemmedSearch = 0&
    .fTitleOnly = 0&
    .fExecute = 1&
    .pszWindow = ""
  End With
```

```
  ShowSearch = HTMLHelpCallSearch(0, _
      "myhelp.chm", _
      HH_DISPLAY_SEARCH, HH_FTS_QUERY)

End Function
```

Create three new RunCode macros to call these three procedures. These will open the help file to the Contents, Index, and Search tabs respectively. In the toolbar customize dialog, create a new toolbar named HTMLHelp, then drop a new menu on it to turn it into a menu bar. Rename this to &Help. Then, using the same toolbar customize dialog, drop each of these macro commands onto the menu and rename them to &Contents, &Index, and &Search. They'll then work as expected.

One of the strangest issues I've found is that, while Access uses What's This popups correctly, it won't let you do the same. For example, in Access 2000, if you set the HelpFile property of a form to the name of an HTML Help file, and set the HelpContextID of the same form to a known context integer, pressing F1 when the form is open will open the HTML Help file to that topic. However, the whole of it gets opened into the Office 2000 help system instead of any window you've specified.

The same thing happens if you set the HelpContextID of a control, even if it's implemented via the What's This Button on the top right of the form. The What's This button is described in the Access 2000 help as follows:

"With the question-mark pointer, you can click any control to access its custom Help topic specified by the control's HelpContextID property. If the control doesn't have a custom Help topic, the form's custom Help topic is displayed. If neither the form or the control has a custom Help topic, Microsoft Access Help is displayed."

Looking into this further, clicking the What's This pointer on the titlebar on the form doesn't fire a WM_HELP message as in other applications. Looking at the WinHelp messages in Microsoft's WinHelp Help Compiler Workshop shows that it's calling one of Access's help files:

```
HELP_SETPOPUP_POS: 611 251
HELP_CONTEXTPOPUP: 3763 -- C:\Program Files\Microsoft
Office\Office\1033\actip9.hlp
```

Public speculation is that this was done so the Office 2000 help authors could still use the WinHelp-style formatting, which isn't possible with the HTML Help text-only popups yet. Now, clicking the What's This pointer on a control as placed on a form by an Access developer like myself opens the Office 2000 help window. However, this occurs without a call being made to either the WinHelp or the HTML Help API:

```
HELP_CONTEXT: 10040 -- D:\ProAccess\sample\context.hlp>MAIN
Processing D:\ProAccess\sample\context.GID
```

This indicates that the opening of the Office 2000 help windows isn't using either the WinHelp or HTML help systems, but is bypassing it to create the custom Office help window. It appears they're calling WinHelp directly instead of firing a WM_HELP message so they can use HELP_CONTEXT for your own help topics vs. the default WM_HELP mechanism.

Notice that the controls in Access have a HelpContextID property but not a WhatsThisID, as the controls in Visual Basic do. What this means is that Access 2000 doesn't really have any workable What's This Help as users have come to expect it. What's This in Access 2000 opens a topic in a standard help window rather than as a popup. This is true even if the What's This source file is HTML Help or WinHelp.

I played around with the RoboHELP 2000 What's This Help composer and Access 2000 one weekend, and ended up with a very strange result. When I had the What's This Help Composer look at an Access 2000 database, I ended up with a message that read: "Could not load this program file. It may be a 16-bit file. What's This? Help is not supported by 16-bit programs." Access 2000 is certainly not a 16-bit program, and looking at this message, it appears the Composer probably may not work with any version of Access.

A member of the Access team suggested creating a class module to accomplish this, and I fully understand what he meant. The HTML Help class module provides VB5 and VB6 developers with the capability to subclass VB forms in order to use the HH_DISPLAY_TEXT_POPUP API call for What's This popups. When I was asked to write the HTML help chapter for Wrox's "Professional Access 2000 Programming" earlier this year, I took a good hard look at how to accomplish the same thing for Access 2000 as I'd done for VB. While Spy++ shows the Access form window, but it's not possible to subclass it properly using code in order to intercept Help messages. Even if I were able to do so, once again, clicking the What's This pointer on a control as defined by an Access developer like myself opens the Office 2000 help window without a call being made to either the WinHelp or the HTML Help API. Here are the details:

I first attempted to use Mabry's MsgHook OCX control. In VB5 and VB6 you can drop the OCX onto a form and have a nice way to subclass the form to intercept the WM_HELP message that's generated when the What's This cursor is clicked on a control. Well, it turns out the MsgHook OCX isn't compatible with Access ...

I headed over to Dev Ashish's site and took a look at some of the subclassing functionality he describes there. I was then able to subclass the Access form to see some things. However, when I used the What's This button on the top-right of an Access form, there was no response from the WM_HELP message. Thinking it may be a custom cursor function, I implemented the What's This cursor from a menu item. The strangest thing happened ... the pointer, complete with the question-mark cursor, pressed the command button when I did a left-button click! In What's This mode, this shouldn't happen. I then intercepted the messages for the Access form using Spy++ ... and whenever I clicked on something on the form with the What's This cursor, no WM_HELP message was sent.

So, when it comes to Access of any variety, use menus and call custom topics from command buttons to your heart's content. But you'll just have to leave What's This help alone.