Visual Basic Concepts

Visual Studio 6.0 29 out of 42 rated this helpful

Using the Internet Transfer Control

The Internet Transfer control implements two widely-used Internet protocols: the HyperText Transfer Protocol (HTTP) and the File Transfer Protocol (FTP). Using the Internet Transfer control, you can connect to any site that uses one of these protocols, and retrieve files using either the OpenURL or Execute method.

Possible Uses

- To add an FTP browser to any application.
- To create an application that automatically downloads files from a public FTP site.
- To parse a World Wide Web site for graphics references and download the graphics only.
- To present a custom display of dynamic data retrieved from a Web page.

Basic Operation

The functionality of the Internet Transfer control depends on the protocol you wish to use. Because the two supported protocols work differently, the operations you can perform depend on which protocol you are using. For example, the GetHeader method only works with HTTP (HTML documents).

However, there are a few procedures that are common to both protocols. Basically, in order to use either protocol, you must:

1. Set the AccessType property to a valid proxy server.
2. Invoke the OpenURL method with a valid URL.
3. Invoke the Execute method with a valid URL and command appropriate to the protocol.
4. Use the GetChunk method to retrieve data from the

buffer.

Setting the AccessType Property: Using a Proxy Server

In order to make any kind of connection to the Internet, you must determine how your computer is connected to the Internet. If you are on an intranet, you will probably be connected to the Internet via a proxy server.

In short, a *proxy server* is an intermediary between your computer and the Internet. All computers on an intranet that need to connect to the Internet must do so through a proxy server. Thus the proxy functions as a *firewall* between the intranet and the Internet, discarding invalid end-user and external requests, thereby protecting the intranet from hostile actions.

To find the proxy settings on your computer

Note The following steps apply only to Windows 95, Windows NT® 4.0, or later systems.

1. On the **Taskbar** of your computer, click **Start**.
2. On the **Settings** item, click the **Control Panel**.
3. Double-click the **Internet** icon.
4. On the **Internet Properties** dialog box, click **Connection**.
5. Under **Proxy Server**, confirm that the **Connect Through a Proxy Server** check box is selected.
6. If it is selected, click **Settings**. The name of proxy servers you use for various protocols will be found in the dialog box. If no proxy is defined, contact your system administrator for available proxy servers.

If you intend to use a proxy other than that named in the dialog box, set the AccessType property to icNamedProxy (2). Then set the Proxy property to the name of the proxy, as shown in the code below:

```
Inet1.Proxy = "myProxyName"
Inet1.AccessType = icNamedProxy
```

On the other hand, if you are content to use the default proxy (as determined by your computer's registry), ignore the Proxy property, and simply set the AccessType to icUseDefault (0).

The settings for AccessType are shown in the following table:

table.

Constant	Value	Description
icUseDefault	0	(Default) Use Defaults. The control uses default settings found in the registry to access the Internet.
icDirect	1	Direct to Internet. The control has a direct connection to the Internet.
icNamedProxy	2	Named Proxy. Instructs the control to use the proxy server specified in the Proxy property.

Invoke the OpenURL Method

After you set the AccessType property, the most basic operation is to use the OpenURL method with a valid URL. When you use the OpenURL method, the result will depend on the target URL. For example, the following URL will return the HTML document found at www.microsoft.com:

```
' A TextBox control named Text1 contains th
e
' result of the method. The Internet Transf
er
' control is named Inet1.
Text1.Text = Inet1.OpenURL("http://www.micr
osoft.com")
```

As a result, the TextBox control is filled with the HTML source, which may resemble the figure below:

```
<HTML>
<HEAD>
<TITLE>Microsoft Corporation</TITLE>
<STYLE>
<!-- ###### defines styles ###### -->
<!--
          A:link {color: 000000; font-weight:bold}
          A:visited {font: 9pt Arial; color: 0099cc; font-weight:bold}
          STRONG {font: 16pt Arial; color: 990000; text-decoration:none}
          BIG {font: 10pt Arial; background: cccc66}
          H1 {font: 24pt Arial; color: 000000}
-->
</STYLE>
<!-- ######  end defines styles ###### -->
```

In this case, the default action was to return the HTML document located at the URL. However, if the URL is modified to target a specific text file, the actual file would be retrieved. For example, the following code:

```
Text1.Text = Inet1. _
OpenURL("ftp://ftp.microsoft.com/disclaimer
.txt")
```

would result in the actual text of the file, as shown below:

```
THE INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
ANY
KIND. MICROSOFT DISCLAIMS ALL WARRANTIES, EITHER
EXPRESSED
OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY
AND
FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL
MICROSOFT CORPORATION OR ITS SUPPLIERS BE LIABLE FOR AN
DAMAGES WHATSOEVER INCLUDING DIRECT, INDIRECT,
INCIDENTAL,
CONSEQUENTIAL, LOSS OF BUSINESS PROFITS OR SPECIAL
DAMAGES,
EVEN IF MICROSOFT CORPORATION OR ITS SUPPLIERS HAVE BEE
ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES
DO
```

Tip When you use either the OpenURL or Execute method, you need not set the Protocol property. The Internet Transfer control will automatically set itself to the correct protocol, as determined by the protocol portion of the URL.

Finally, you can use the OpenURL method with a URL that includes appended data. For example, many Web sites offer the ability to search a database. To search, send a URL that includes the search criteria. For example, the following code would use a search engine named "search.exe" with the criteria "find=Maui."

```
Dim strURL As String
strURL = _
"http://www.megaphone43.com/cgi-bin/search.
exe?find=maui
Text1.Text = Inet1.OpenURL(strURL)
```

If the search engine finds a match for the criteria, an HTML document would be assembled and returned with the appropriate information.

Saving to a File Using the OpenURL Method

If you wish to save the data retrieved through the OpenURL method to a file, use the Open, Put, and Close statements, as shown in the code below. This example streams a binary file into a Byte array before saving the data to disk:

```
Dim strURL As String
Dim bData() As Byte      ' Data variable
Dim intFile As Integer   ' FreeFile variabl
e
strURL = _
"ftp://ftp.microsoft.com/Softlib/Softlib.ex
e"
```

```
e
intFile = FreeFile()       ' Set intFile to
an unused
                           ' file.
' The result of the OpenURL method goes int
o the Byte
' array, and the Byte array is then saved t
o disk.
bData() = Inet1.OpenURL(strURL, icByteArray
)
Open "C:\Temp\Softlib.exe" For Binary Acces
s Write _
As #intFile
Put #intFile, , bData()
Close #intFile
```

A similar procedure can be used to write a text file to disk, except no Byte array is needed; the data is saved directly to the file:

```
Dim strURL As String       ' URL string
Dim intFile As Integer    ' FreeFile variabl
e
IntFile = FreeFile()
strURL = "http://www.microsoft.com"
Open "c:\temp\MSsource.txt" For Output _
As #IntFile
Write #IntFile, Inet1.OpenURL(strURL)
Close #IntFile
```

Synchronous and Asynchronous Transmission

The OpenURL method results in a *synchronous* transmission of data. In this context, synchronous means that the transfer operation occurs before any other procedures are executed. Thus the data transfer must be completed before any other code can be executed.

On the other hand, the Execute method results in an *asynchronous* transmission. When the Execute method is invoked, the transfer operation occurs independently of other procedures. Thus, after invoking the Execute method, other code can execute while data is received in the background.

What does this mean for the user of the Internet Transfer control? In short, using the OpenURL method results in a direct stream of data that you can save to disk (as shown above), or view directly in a TextBox control (if the data was text). On the other hand, if you use the Execute method to retrieve data, you must monitor the control's connection state using the StateChanged event. When the appropriate state is reached, invoke the GetChunk method to retrieve data from the control's buffer. This operation is discussed in greater detail below.

greater detail below.

Using the Execute Method with the FTP Protocol

The Execute method has four arguments: *url*, *operation*, *data*, and *requestHeaders*. FTP operations take only the *operation* argument and the *url* argument, which is optional. For example, to get a file from a remote computer, you could use the following code:

```
Inet1.Execute "FTP://ftp.microsoft.com", _
"GET disclaimer.txt C:\Temp\Disclaimer.txt"
```

If you are used to using FTP to retrieve files from anonymous FTP servers, you will be familiar with certain commands used to navigate through server trees, and to retrieve files to a local hard disk. For example, to change directory with the FTP protocol, you would use the command "CD" with the path of the directory you wish to change to.

For the most common operations, such as putting a file on a server and retrieving a file from a server, the Internet Transfer control uses the same or a similar command with the Execute method. For example, the following code uses the "CD" command as an argument of the Execute method to change directory:

```
' The txtURL textbox contains the path to o
pen. The
' txtRemotePath textbox contains the path t
o change to.
Inet1.Execute txtURL.Text, "CD " & txtRemot
ePath.Text
```

Note When using the Execute method with FTP commands, the *data* and *requestHeaders* arguments are not used. Instead, all of the operations and their parameters are passed as a single string in the *operation* argument; parameters are separated by a space. In the descriptions below, do not confuse the terms "file1" and "file2" with the *data* and *requestHeaders* arguments.

The syntax for FTP operations is:

```
operationName file1 file2
```

For example, to get a file, the following code includes the operation name ("GET"), and the two file names required by the operation:

```
' Get the file named Disclaimer.txt and cop
y it to the
```

```
' location C:\Temp\Disclaimer.txt
Inet1.Execute, _
"GET Disclaimer.txt C:\Temp\Disclaimer.txt"
```

The following table lists the supported FTP commands of the control:

Operation	Description	Example
CD *file1*	Change Directory. Changes to the directory specified in *file1*.	`Execute , "CD docs\myd`
CDUP	Change to Parent. Same as "CD .."	`Execute , "CDUP"`
DELETE *file1*	Deletes the file specified in *file1*.	`Execute , "DELETE disc`
DIR [*file1*]	Searches the directory specified in *file1*. If file1 isn't supplied, the current working directory is searched. Use the GetChunk method to return the data.	`Execute , "DIR /mydocs`
GET *file1* *file2*	Retrieves the remote file specified in *file1*, and creates a new local file specified in *file2*.	`Execute , _` `"GET getme.txt C:\gotm`
MKDIR	Creates a	

MKDIR *file1*	Creates a directory as specified in *file1*. Success is dependent on user privileges on the remote host.	`Execute , "MKDIR /myDi`
PUT *file1 file2*	Copies a local file specified in *file1* to the remote host specified in *file2*.	`Execute , _` `"PUT C:\putme.txt /put`
PWD	Print Working Directory. Returns the current directory name. Use the GetChunk method to return the data.	`Execute , "PWD"`
QUIT	Terminate current connection	`Execute , "QUIT"`
RECV *file1 file2*	Same as GET.	`Execute , _` `"RECV getme.txt C:\got`
RENAME *file1 file2*	Renames a file. Success is dependent on user privileges on the remote host.	`Execute ,` `"RENAME old.txt new.tx`
RMDIR *file1*	Remove directory. Success is	`Execute , "RMDIR oldDi`

	dependent on user privileges on the remote host.	
SEND *file1*	Copies a file to the FTP site. (same as PUT.)	`Execute , _` `"SEND C:\putme.txt /pu`
SIZE *file1*	Returns the size of the file specified in *file1*.	`Execute "SIZE /largefi`

Important If your proxy server is a CERN proxy server, direct FTP connections (using the Execute method) are disallowed. In that case, to get a file, use the OpenURL method with the Open, Put, and Close statements, as shown earlier in "Saving to a File Using the OpenURL Method." You can also use the OpenURL method to get a directory listing by invoking the method and specifying the target directory as the URL.

Using the Execute Method with the HTTP Protocol

The HTTP protocol allows client machines to request data from the server using the GET, HEAD, POST, and PUT commands. These operations are shown in the following table:

Operation	Description	Example
GET	Retrieves the file named in *url*.	`Execute "http://www.mi` `"/default.htm", "GET"`
HEAD	Retrieves only the headers of the file named in the URL property.	`Execute , "HEAD"`

POST	Provides additional data to support a request to the remote host.	`Execute , "POST", strF`
PUT	Replaces data at the specified URL.	`Execute , "PUT", "repl`

The Common Gateway Interface and the Execute Method

Many World Wide Web sites offer the ability to search a database. This is accomplished by using the HTTP protocol's ability to send queries using the Common Gateway Interface (CGI).

It is not in the scope of this topic to explain the CGI; however, if you are familiar with the CGI, you can use the Execute method to construct an application that simulates the behavior of World Wide Web sites. For example, the code below shows a typical CGI query string:

```
http://www.findThis2490.com/cgi-bin/find.ex
e?find=Hangzhou
```

This same query could be sent using the Execute method as shown below:

```
Dim strURL As String, strFormData As String
strURL = "//www.findThis2490.com/cgi-bin/fi
nd.exe"
strFormData = "find=Hangzhou"
Inet1.Execute strURL, "POST", strFormData
```

If you are expecting a result back from a server (as in the example above), you must use the GetChunk method to retrieve the resulting HTML document.

Using the State Event with the GetChunk Method

When you are downloading data from a remote computer, an asynchronous connection will be made. For example, using the Execute method with the operation "GET", will cause the server to retrieve the requested file. When the entire file has been retrieved, the State argument will return icResponseCompleted (12). At that point, you can use the GetChunk method to retrieve the data from the buffer. This

GetChunk method to retrieve the data from the buffer. This is shown in the example below:

```
Private Sub Inet1_StateChanged(ByVal State
As Integer)
   Dim vtData As Variant ' Data variable.
   Select Case State
   ' ... Other cases not shown.
   Case icResponseCompleted ' 12
      ' Open a file to write to.
      Open txtOperation For Binary Access _
      Write As #intFile

      ' Get the first chunk. NOTE: specify
a Byte
      ' array (icByteArray) to retrieve a b
inary file.
      vtData = Inet1.GetChunk(1024, icStrin
g)

      Do While LenB(vtData) > 0
         Put #intFile, , vtData
         ' Get next chunk.
         vtData = Inet1.GetChunk(1024, icSt
ring)
      Loop
      Put #intFile, , vtData
      Close #intFile
   End Select
End Sub
```

Logging on to FTP Servers

FTP servers come in two flavors: public and private. Public servers, as suggested by the name, are open to anyone. Private servers, on the other hand, won't let you log on unless you are a bona fide user of the server. In either case, the FTP protocol demands that you supply a user name and a password. The two are used to authenticate a user and allow (or disallow) subsequent actions.

To log on to public servers the common practice is to log in as "anonymous," (UserName = "anonymous") and send your e-mail name as the password. However this process is simplified even further with the Internet Transfer control. By default, if you do not supply UserName and Password property values, the control sends "anonymous" as your UserName, and your e-mail name for the Password.

If you are logging on to a private server, simply set the UserName, Password, and URL properties as appropriate, and invoke the Execute method, as shown in the example below:

```
With Inet1
   .URL = "ftp://ftp.someFTPSite1020.com"
```

```
    .UserName = "John Smith"
    .Password = "mAuI&9$6"
    .Execute ,"DIR"   ' Returns the director
y.
    .Execute ,"CLOSE" ' Close the connection
.
End With
```

After you have invoked the Execute method, the FTP connection will remain open. You can then continue to use the Execute method to perform other FTP operations such as CD and GET. When you have completed the session, close the connection using the Execute method with the CLOSE operation. You can also close the connection automatically by changing the URL property, and invoking either the OpenURL or Execute method; such action will close the current FTP connection, and open the new URL.