Extending Web Apps using Visual FoxPro

Eric Selje
Salty Dog Solutions, LLC
2505 Commonwealth Ave.
Madison, WI 53711
Voice: 608/213-9567
Email: Eric@SaltyDogLLC.com
WebSite/Blog: SaltyDogLLC.com
LinkedIn/Facebook/Twitter:EricSelje

From CRM and bookkeeping to email and your backups, many applications are now on-line, but that doesn't mean they're beyond your control. In this session you'll learn how to use a web application's API using tools from within Visual FoxPro in order to extend, exploit, and exchange information with that application. We'll use the FreshBooks API as an example and learn about cURL and LibCurl class library.

Down the Rabbit Hole

Like many things, this all started out of frustration: How could an application be so good, yet omit such obvious (to me) functionality?

The application is FreshBooks, an online bookkeeping system that is used by thousands of users, from single independent consultants like me all the way up to medium-sized businesses with multiple users. FreshBooks is very good, but there were things I wanted it to do that weren't there yet.

FreshBooks is a web app, which means we can't get at the source code. It's "out there," beyond our reach. But FreshBooks, like many web applications, *does* publish an "API," a way to get at its data. The trick is knowing how to do that, and that's what this session is about.

Start at the Beginning – What are Web Applications?

At its essence, a web application is essentially a website that you log into through your PC's web browser to run. Nothing runs on your local PC except your browser.

A web application is different than a mere website in that in includes security to show you only the information that is pertinent to you, and it usually includes interactivity to allow you to manipulate data that is stored on the site.

Examples of popular web applications include

- email: Gmail, Yahoo Mail, Hotmail (or whatever Microsoft is calling it now)
- Social Stuff: Twitter, Facebook, MySpace, LinkedIn, Orkut
- Business Stuff: Salesforce.com, Zoho, MS Dynamics, Highrise
- Office Suites: Google Apps, LiveOffice, Central Desktop
- Developer Tools: Beanstalk, Basecamp, Github, Codeplex
- Accounting: FreshBooks, Mint, Shoeboxed

There are many, many more but I think you get the idea.

*Note: We're **not** talking about embedding a web browser control within your VFP application and manipulating that (see Bo Durban's session for that). Nor are we talking about making your VFP data available on the web (see Paul Mrozowski or Venelina Jordanova's session for that).*

There are lots of advantages to companies who write web applications rather than distribute client (Windows) applications that need to be installed on each user's PC, but from a user's perspective one big *disadvantage* of web applications is that, well, they're online. That means if you're off-line you can't get at your data; and if the web application

goes away tomorrow…well shudder to think. But if we could get at the data that the web app is storing, well then we've got something!

But how do we get at that data? The web app developer must provide some sort of way, some sort of "interface" for us programmers to access their application: an application programmer's interface, if you will. Well that's what an API is.

There are a wide variety of methods that different APIs handle the details. Some might have you authenticate yourself with the newer oAuth way, while others may use the tried-and-true SOAP style *ws-security* protocols. When making requests, some might use a REST-ful style URL, maybe because they've implemented oData on their site which gives them that functionality, but others may use SOAP (simple object-access protocol). Many others will have a proprietary protocol.

We can't look at every possibility that exists, so I've settled on one particular API to use as an example: FreshBooks.

What is FreshBooks?

In this session we'll use FreshBooks for our ~~guineau pig~~ demonstration. I have used FreshBooks for my bookkeeping solution for a few years now, and I can honestly say that it makes me not hate time tracking and invoicing. It's very good and continues to improve.

This is the "FreshBooks Manifesto," from their site:

FreshBooks Manifesto

Our mission is to deliver fast and simple invoicing and time tracking services that help you manage your business. We call these Unaccounting™ services and they will:

- Save you time
- Be easy to use
- Make you look professional (think Fortune 500)
- Let you manage your books without an accountant
- Secure, encrypt, and back up your data
- Be available 24/7 from anywhere with any computer

The best things about FreshBooks are:

- It really does make me appear like a larger company to my clients. When I send them a link to their invoice, they don't get a simple PDF they can print. Rather, they log into my website and can view their statement, individual invoices, confirm payments, and even submit trouble tickets.
- It has various ways to track time. I normally use the link from my site, but I sometimes use the iPhone app as well. There's also a Windows "Widget" I could use, a BlackBerry or Android app, and probably more ways.
- It makes invoicing dead simple and postage-free, and that gets me paid quicker.

In the live session we'll now go through setting up a demo company in FreshBooks, but for you reading this whitepaper I recommend going to www.FreshBooks.com/Tour.php and walking through its features.

What FreshBooks Lacks

Ok, did you take the tour? Good stuff, isn't it? You saw a lot of what FreshBooks can do and it's a pretty impressive set of features. However, once you've been using FreshBooks for a while you start wishing it could do a few more things. And this goes for any web app.

For the purposes of this session, I set up a free account in FreshBooks under the name "Grotto Gatherings." As far as I know, Grotto Gatherings is not a real company and you should not confuse it with a real company. Just to get things seeded a bit, I've also added a client and a bit of time into Grotto Gatherings as well.

After doing that, I can run some of the reports that FreshBooks gives me or send my fictitious client an invoice via email.

Here are a few features that I would love to see in FreshBooks:

- A report that shows me all unbilled time, summarized by client. I don't know why this isn't in there as it would seem to me to be something everyone would want. Currently the only way to know if you need to invoice a client is by creating an invoice. And I don't like to bill my clients for small amounts, as it's a pain for both of us.
- A client statement that only shows unpaid invoices. Currently the client statements show a complete history of invoices and payments, which may be overwhelming and confusing. Ideally the statement would only show unpaid invoices. Again, this seems like a no-brainer.
- A way to export a client's unbilled time into a custom CSV form that my client has specified that they'd like in order to import it into their payment system. At the same time it might be useful to export to XML for even greater flexibility. I guess I can understand why FreshBooks hasn't implemented this, since it's a bit specific to my needs.

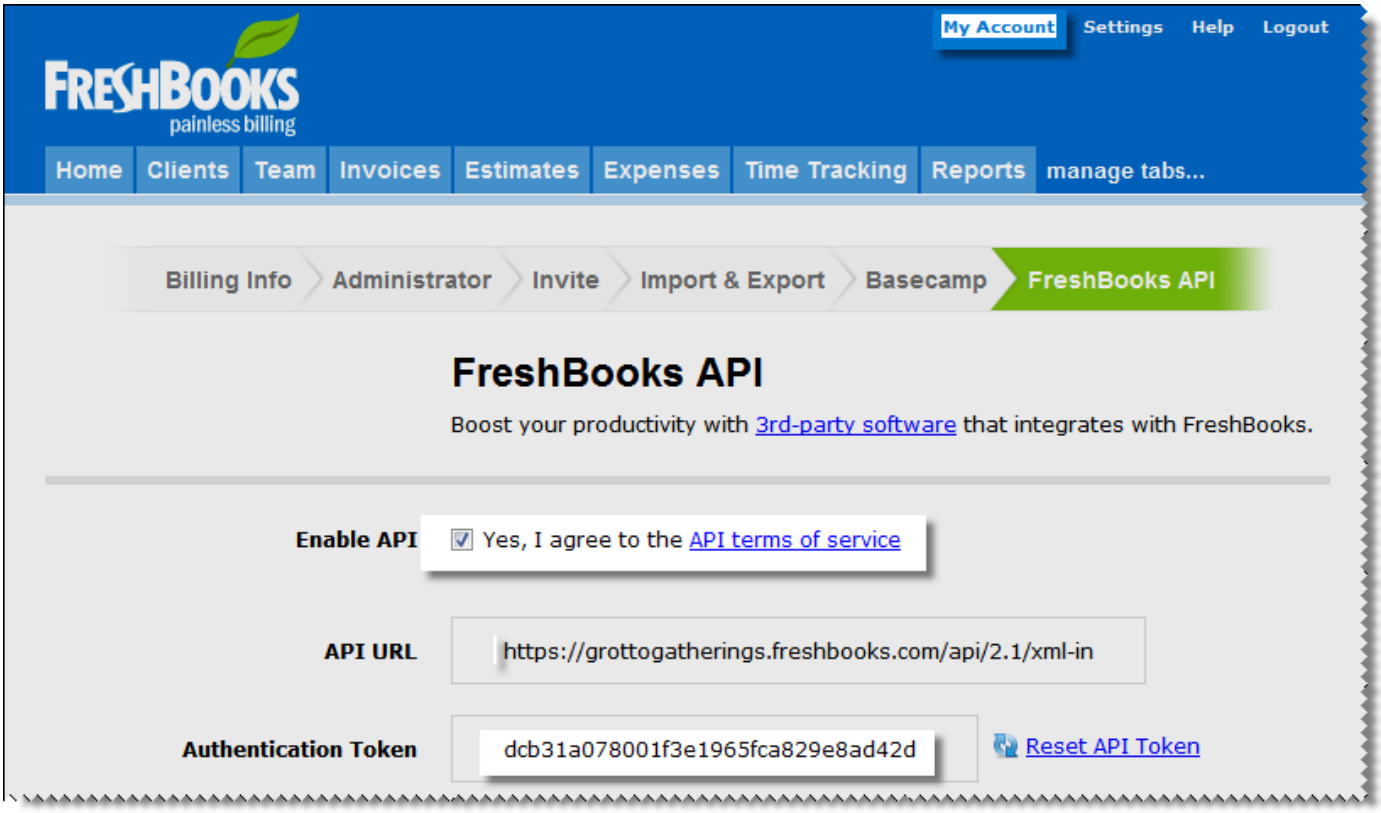
You can certainly submit a ticket to FreshBooks and hope they'll implement whatever feature you ask for into the next "version" (and I use the term version lightly, because web apps can be updated overnight without you even being aware of it). But we're programmers! We should be able to code whatever features we want without relying on other developers, right? Right. If you know how. Let's figure out how to do that.

The FreshBooks API

The key to all of this is understanding and using the FreshBooks API. Thankfully FreshBooks' API is well-documented and well-supported. It's also very straightforward which makes it very easy to use.

To get started using the API from an external application, you first have to "enable" it in FreshBooks. By default interaction with your FreshBooks account via the API is disabled. This is a great security feature because it removes a potential hole for an account that will never need to use the API.

To enable the API for your FreshBooks account, click "My Account" at the top of the FreshBooks page, and then click on the "FreshBooks API" tab:



Click the checkbox that say you agree to the Terms of Service (be sure to read it first!), and you'll get the Authentication Token.

That "Authentication Token" is your user id that we'll need later. *[And if you're curious, yes, I've clicked the "Reset API Token" link after creating this image, so this token will no longer work. Now I have to go back to my code and put that token in again. More on that later]*. Most APIs don't require you to give out the username and password that you yourself use to log into the web app, but instead they use this long string of characters. Knowing this token doesn't give a hacker the keys to your kingdom: they can only use it to do what the API makes available; but there's no API function for, say, changing your username and password.

A Word about "oAuth"

For your private application's interface with FreshBooks, using an authentication token may work just fine because you've got control. But imagine if you were developing a commercial or open-source application to integrate with FreshBooks and didn't want to have the user entering their token. That's a scenario where the new "oAuth" authentication protocol would be preferable.

With oAuth, your application must explicitly request permission to access the web application. When the web app approves your request, it calls the callback function that you left it, and send along an oAuth token, along with other information such as an expiration date. You can then use that oAuth token to access the API just like the regular token, up until it expires. It's analogous to the requirement of explicitly inviting a vampire into your home.

A big advantage to using oAuth is that the token doesn't change if you change your username or password – you never gave that out. If you use Facebook, Mint.com, Picasa, or any of a number of web apps, you know that the 3rd party add-ons break if you change your password, and you have to authenticate them. This doesn't happen with oAuth.

Now that the API is enabled for your account, let's try it out and make sure it's working.

Introducing CURL[1]

We're going to start by using a command-line tool called "curl" to access the FreshBooks API. Curl is a "tool for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and a busload of other useful tricks.")[2]

For our purposes dealing with the FreshBooks API we really only care about the HTTP part of all of those features, although it's reassuring to know all of that other stuff is there too and we may definitely want to use some of that in our apps as well.

I chose curl for this demonstration for five reasons, not necessarily sorted in order of importance:

- It may be new to you
- It's fairly simple to implement
- The FreshBooks API documentation uses curl for its examples
- It's free
- There is a VFP class library that implements it, LibCurl.vcx

I am certain I could have done this session using a proven VFP utility like West Wind's IPStuff library. I believe most VFP developers are well aware of Rick Strahl's excellent tools, and by using curl I'm hoping to introduce something unfamiliar to the arsenal.

I might have also tried accessing the API just by making use of the Net methods available in the Windows API, but I am not a glutton for such punishment.

Installing Curl

To start with curl, first download it from http://curl.haxx.se/download.html. Be sure to get the binaries, not the source code (unless you want to compile it yourself) for Win32 or Win64, depending on your machine. As of this writing, the most current version is 7.21.1 for Win32, and 7.21.0 for Win64. You also want to be sure you get the version that includes SSL, as we'll need that to connect to our web apps securely.

You now have one file, *curl.exe*, on your hard drive. Fortuitously, as previously mentioned, the documentation on FreshBooks' API page[3] uses curl.exe.

Testing FreshBooks Connectivity with Curl

---

[1] In typical UNIX fashion, the name "curl" is a pun because the function of curl is to See URLs. Get it? You'll even often see it spelled cURL to really emphasize this pun.
[2] http://curl.haxx.se
[3] http://developers.freshbooks.com/

When you access your FreshBooks, you use a URL based on the name of your company that you used when you set up FreshBooks.. So for our sample company, Grotto Gatherings, the URL is https://GrottoGatherings.freshbooks.com/api/2.1/xml-in (note, that's **https** as in secure which is reassuring).

To that URL you will post an XML payload that makes the request. For example,

```
<?xml version="1.0" encoding="utf-8"?>
<request method="system.current">
</request>
```

This is the "Help, About" request that tells you about your company's FreshBooks setup. Take that XML and put it in a file, such as *FBRequest.xml.* Now issue this from the Command Window:

```
Curl –u dcb31a078001f3e1965fca829e8ad42d:X
https://GrottoGatherings.freshbooks.com/api/2.1/xml-in -d @FBRequest.xml -k
```

- The *–u* parameter is the username and password. We used our authentication token from FreshBooks as the username, and followed that up with any password because with the token, you don't actually need a password
- After that is our URL, followed up with the *–d* parameter for our data. Because we put our data into a file we precede the file name with '@'. Supposedly you can put the XML right here on the command line, but I could never get that to parse correctly with the quotes and XML.
- That last parameter, *-k*, is telling curl to connect to the Secure site, but ignore the fact that I don't have a certificate on my machine. You may not need this.
- For the complete list of curl command line arguments, you can type in curl --help. For convenience, I've included it here in Appendix A. You can get a comprehensive curl manual by type curl --manual (you may want to redirect this to a text file because it scrolls by pretty fast.)

If everything went well, you will get this result:

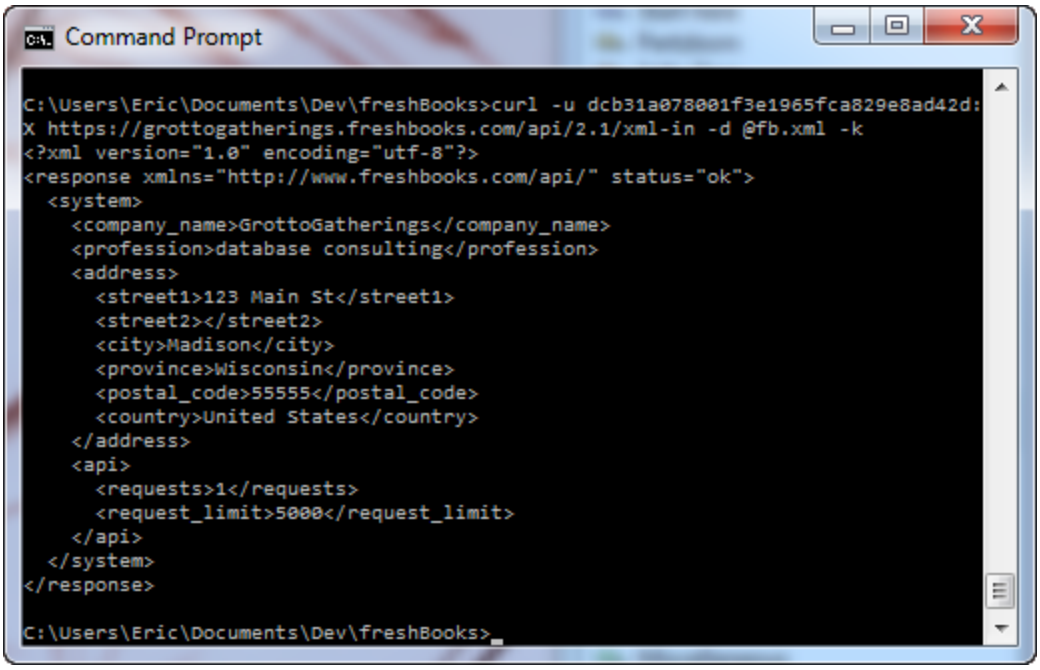


Figure 1 Command line curl.exe

Now by simply changing the XML file *FBRequest.xml*, we can get at most of the data in our FreshBooks account. Here are a few sample requests:

1. List of all of my clients:

```
<?xml version="1.0" encoding="utf-8"?>
<request method="client.list"></request>
```

2. List all payments I received in August:

```
<?xml version="1.0" encoding="utf-8"?>
<request method="payment.list">
  <date_from>2010-08-01</date_from>
  <date_to>2010-08-30</date_to>
</request>
```

3. List my time from last week:

```
<?xml version="1.0" encoding="utf-8"?>
<request method="time.list">
  <date_from>2010-09-12</date_from>
  <date_to>2010-09-18</date_to>
</request>
```

Isn't that simple? In fact, the hardest thing is knowing how to get at it from within Visual FoxPro, which we'll take a look at in the next section.

Enter LibCurl

The command-line interface is easy, but it's not as useful to Windows programmers as, say, a DLL would be. This is where LibCurl comes in.

LibCurl is a cross-platform tool that wraps all of curl's functionality into an easy to use package. For Windows users like us, that package is a DLL. We could directly interact with this DLL ourselves, but much like VFP2C32[4] did for the WindowsAPI, Carlos Alloatti has wrapped LibCurl.dll into a class library for Visual FoxPro called, appropriately, LibCurl.vcx.

To make installation dead simple, Carlos has put together a zip file that will install the LibCurl.dll, LibCurl.vcx, and all other necessary files to make this work. You can get these at his website, http://www.ctl32.com.ar/libcurl.asp[5]. Simply unzip this into a folder to install: there are no DLLs to register.

With the LibCurl.vcx class library you now have most of the functionality of curl available to you within Visual FoxPro. All of the options that you would have to set with command-line switches in curl.exe (See Appendix A) are now merely properties of the object. You can upload and download files via FTP or HTTP, retrieve web pages, post information to websites, and lookup directory information (including directory or Active Directory if you've got LDAP enabled on it).

Now that we've got LibCurl installed, let's simulate some of what we did with curl.exe from the Command Window within Visual FoxPro.

Let's try using libCurl from the Visual FoxPro's command line window in order to retrieve a webpage and do a little "screen scraping" to see if a certain somebody is coming to the conference. Create this program, and save it as IsAttending.prg:

[4] [You may even notice that VFP2C32.dll is included in this Zip file. This is a FoxPro Library created by Christian Ehlscheid that wraps Windows API functions to make calling them from VFP easier. LibCurl.vcx uses VFP2C32 to set up "callback" functions, which we'll discuss later, but if you're interested in VFP2C32 be sure to see my other session.]

[5] The most recent version, as of this writing, is LibCurl 20081013

```
LPARAMETER cUserName
ASSERT(NOT EMPTY(cUserName)) MESSAGE "Usage: IsComing('cUserName')"

LOCAL lReturn, cTempFile, cAttendingPage, oCurl
SET CLASSLIB TO libCurl.vcx  ADDITIVE
oCurl = CREATE("libCurl")
cTempFile = FORCEEXT(ADDBS(SYS(2023))+SYS(2015),'txt')

iResult = oCurl.httpDownloadFile("http://swfox.net/whoscoming.aspx",
cTempFile)
IF iResult = 0  && Everything is ok
   cAttendingPage=FILETOSTR(cTempFile)
   oCurl = .NULL.
   lReturn = ATC(cUserName, cAttendingPage)>0
   ERASE (cTempFile)
ENDIF

RETURN lReturn
```

Now we have a quick way to check who's coming to this year's Southwest Fox conference. Try it out:

```
? IsAttending("Henzten, Whil")
```

Figure 2 IsAttending.prg

Or

```
? IsAttending(Yourname)
```

It'd be pretty easy to extend this example to do a lot more, and swfox.net isn't even a web "app," it's just a website. Imagine all the sites you could use this for! For example, I signed up for my radio station's contest line and every day they post three winners on their website. But if they think I'm going to check their slow, ad-ridden website every day they are mistaking. I wrote a little script in VFP that uses LibCurl to download the page and scan for my last name.

Another example of something you could do with LibCurl is backup some file, such as a log file or database file, to an ftp site. To do this you'd simply call LibCurl's UploadFTPFile function, like this:

```
LPARAMETER cFileName

IF NOT EMPTY(cFileName) AND FILE(cFileName)
   LOCAL iResult, oCurl AS libCurl OF LibCurl.vcx
   SET CLASSLIB TO LibCurl.vcx  ADDITIVE
   oCurl = CREATE("libCurl")
   cRemoteFile = "ftp://UserName:Pwd@Site/" + Justfname(cFileName)
   iResult = oCurl.ftpuploadfile(cRemoteFile, cFileName)
   IF iResult <> 0
      MESSAGEBOX(oCurl.curleasystrerror(iResult),0,"FTP Upload")
   ENDIF
   oCurl = .NULL.
ENDIF
```

Try it with

```
UploadFile("libCurl.log")
```

Check your ftp site and you should see your file is now uploaded.

We could add a lot more robustness to this example by doing some error checking, moving the authentication parts out of the source code, making directories, and verifying the file made it up there, but this session isn't mean to be a comprehensive LibCurl tutorial. I hope you're getting the idea of what LibCurl can do for you, and you can get even more examples from the *example_project* folder under the installation folder of LibCurl.

Accessing FreshBooks from Visual FoxPro

Figure 3 UploadFile.prg

Everything we need is now in place for us to access our FreshBooks account from within Visual FoxPro, where we can do just about anything including plugging up those shortcomings we pointed out in the beginning.

Let's get our FreshBooks system information using LibCurl. Here are the commands in Visual FoxPro that get the system information:

```
#INCLUDE LibCurl.h
LOCAL lReturn, cRequest, cLastResult, oCurl AS libCurl OF LIBCURL.VCX
IF ATC("libcurl.vcx", SET("ClassLib"))=0
	SET CLASSLIB TO LIBCURL.VCX ADDI
ENDIF

cLastResult = ""
oCurl = CREATEOBJECT("libCurl")
lReturn = VARTYPE(oCurl)='O'
IF lReturn
	WITH oCurl
		.curleasyreset()
		* Log into the sample GrottoGatherings site
		.curlOptUserPwd= This.cUserPwd
		.curlOptUrl= This.APIUrl

		* Set authentication options
		.curlOptUseSSL= CURLUSESSL_ALL
		.curlOptSSLVersion = CURL_SSLVERSION_DEFAULT
		.CurlOptHttpAuth = CURLAUTH_BASIC

		* Using examples from http://invoicesync.com/blog/?p=36
		.curlOptSslVerifyHost = 2
		.curlOptSslVerifyPeer = .f.
		.curlOptPort=443
		.curlOptPost=.t.

		.curlOptHttpHeader="Content-Type: text/xml"
		.curlOptUseragent="Visual FoxPro Agent/LibCurl"
```

```
            cRequest=[<request method="system.current"></request>]

            .curlOptPostFieldsize=-1      && Let curl figure this out
            .curlOptCopyPostfields= cRequest
            iResult = .curlEasyPerform()
            IF iResult != 0
                  cLastResult = oCurl.curleasystrerror(iResult)
            ENDIF
            cLastResult = cLastResult + ._databuffer

      ENDWITH
      oCurl = .null.
ENDIF
RETURN cLastResult
```

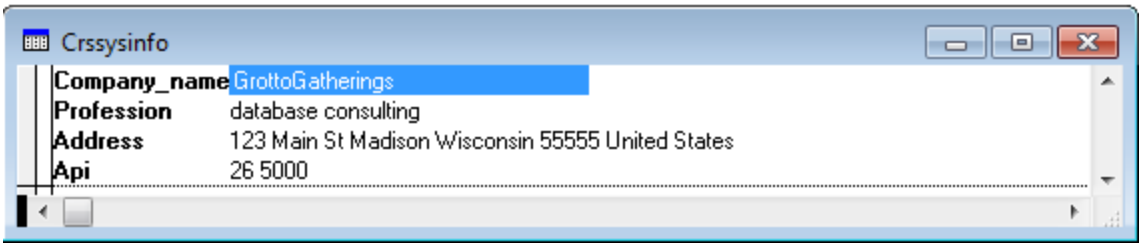
Save this code as getSystemInfo.prg. Then issue a `? getSystemInfo()` from the command line and you will get back the results as an XML string:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://www.freshbooks.com/api/" status="ok">
  <system>
    <company_name>GrottoGatherings</company_name>
    <profession>database consulting</profession>
    <address>
      <street1>123 Main St</street1>
      <street2></street2>
      <city>Madison</city>
      <province>Wisconsin</province>
      <postal_code>55555</postal_code>
      <country>United States</country>
    </address>
    <api>
      <requests>26</requests>
      <request_limit>5000</request_limit>
    </api>
  </system>
</response>
```

Dealing with XML in Visual FoxPro

Our results from FreshBooks come back as XML, which Visual FoxPro has limited support for. If you try to use XMLToCursor() on the results of the System.Current request in the previous example, you'll be disappointed in the results.

```
cXML = getSystemInfo()
XMLToCursor(cXML, "crsSysInfo")
EDIT
```



XMLToCursor only parses the top-level children on the XML tree, so the Address and API nodes, which have children, are not broken down into their component parts. The XMLAdapter class doesn't help either.

For more precise parsing of XML, you'll want to use a better tool, such is Microsoft's XML parser, MSXML, which is probably already installed on your machine. For example, in order to get the city name from the System.Current response, you would parse it like this:

```
oXML=CREATEOBJECT("MSXML2.DOMDocument")
oXML.loadXML(cXML)
cCity =oXML.selectSingleNode("//response/system/address/city").text
```

Extending FreshBooks

Now that we're able to use LibCurl to communicate with We're now ready to extend the functionality of FreshBooks.

To start doing this I've designed a class library named FBApi.vcx to interface with FreshBooks (a copy of which is available with the session materials). Its main class is FBConnection, which includes the connectivity functionality we need to access FreshBooks. It also functions as a "Class Factory" for the entity objects.

Another class, FBEntity, is an abstract superclass of which all the entities in FreshBooks (clients, projects, time_entries, etc.) are subclassed. These entity classes have all the "CRUD" functionality, as well as a "Sync()" method to replicate the FreshBooks data into local FoxPro tables (which it will create on the fly if they don't exist) that we can use to create custom reports and other functionality.

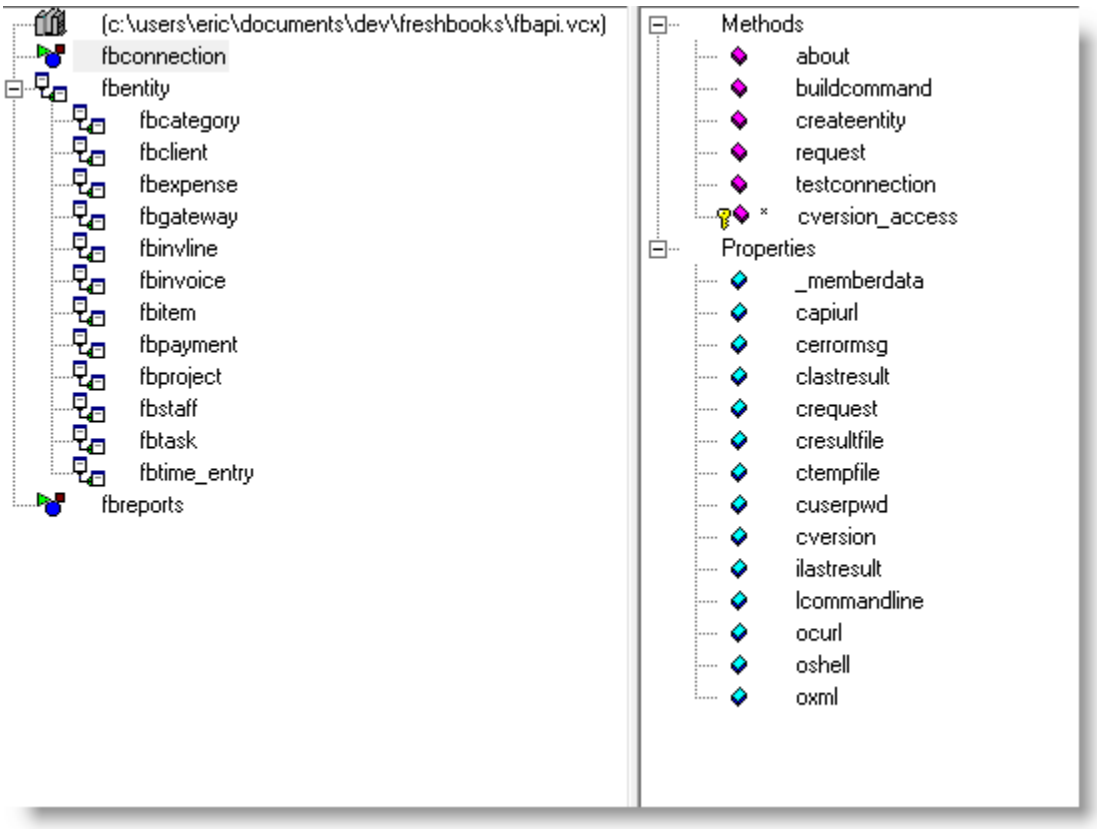


Figure 4 The FBConnection Class

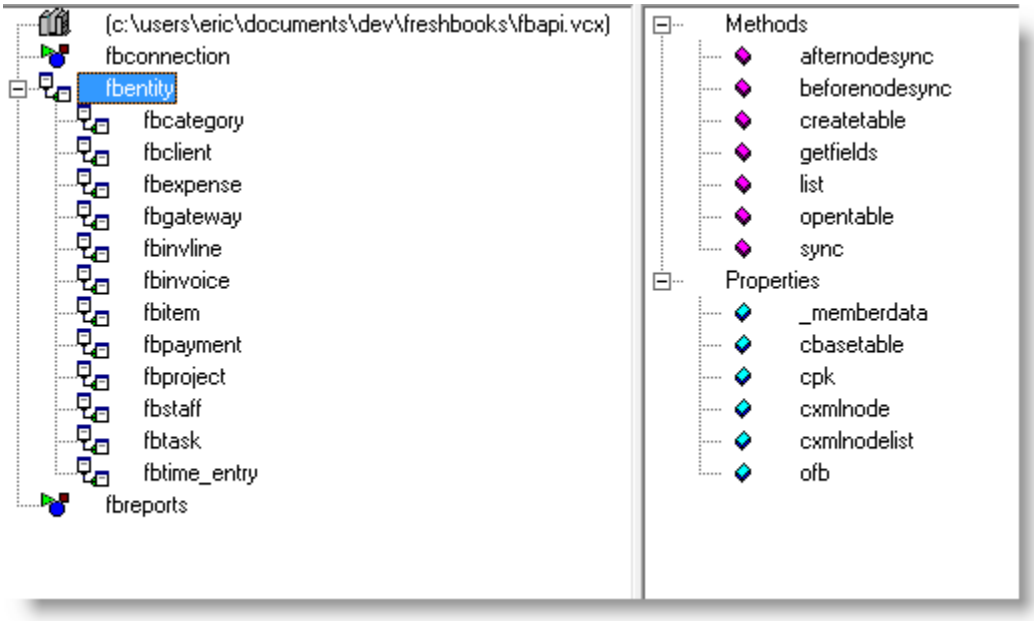


Figure 5 FBEntity Class and its descendants

Example: Listing the clients

In order to Sync the clients, you first create an instance of the FBConnection object:

```
SET CLASSLIB TO FBApi
oFBAPI = createObject("FBConnection")
```

Then you create a client entity object, which is of type fbClient though you don't have to know that:

```
oClient = oFBApi.createEntity("client")
```

And then invoke the client's Sync() method:

```
oClient.sync()
```

Doing something similar for any of the other entities follows a parallel

```
oTime = oFBApi.createEntity("time_entry")
oTime.sync()

oTask = oFBApi.createEntity("task")
oTask.sync()

oProject = oFBApi.createEntity("project")
oProject.sync()
```

The naming conventions used throughout the FreshBooks API is so consistent that we are able to use the abstract method in the FBEntity class for almost all entity's synching. The fields that are different for the different entities are placed in the class's properties rather than getting hardcoded into the method itself:

```
* Sync
* Abstract method Synchronize the entities

LOCAL cXML, iNodeId, cPk
LOCAL oXML as MSXML2.DOMDocument, oNodeList as MSXML2.IXMLDOMNodeList, oNode as
MSXML2.IXMLDOMNode

cXML = This.List()  && Gets all the data for this entity from FreshBooks
```

```
* Parse the XML and ensure each is in our FoxPro table
This.openTable(This.cBaseTable)
This.oFb.oXML.LoadXML(cXML)
oNodeList=This.oFb.oXML.selectNodes(This.cXMLNodeList)
iNodeCount = oNodeList.length
* Go through each child, see if it's in our FoxPro table, and add it if not
iChild = 0
SELECT (This.cBaseTable)
cPk = This.cPk
DO WHILE iChild < iNodeCount
   oNode = oNodeList.item(iChild)
   iNodeId = VAL(oNode.selectSingleNode(This.cXMLNode+'/'+This.cPk).text)
   IF This.beforeNodeSync(iNodeId, oNode)
      IF NOT SEEK(iNodeId) && This is a new entity
         INSERT INTO (This.cBaseTable) ((cPk)) VALUES (iNodeId)
      ENDIF
      SELECT (This.cBaseTable)
      * Update fields
      This.getFields(This.cBaseTable, oNode)
      This.afterNodeSync(iNodeId, oNode)
   ENDIF
   iChild = iChild + 1
ENDDO
USE IN SELECT(This.cBaseTable)
```

I added some hooks before and after the node is imported, just in case there is some entity-specific functionality that needs to go in there.

One key method here is getFields(), which takes advantage of the fact that we named our local FoxPro table's fields with the exact same name as the XML element that is returned from FreshBooks. That allows us to write this generic routine that can parse the XML and stick the data into the fields. It has to do a little massaging of the XML, since everything that comes back in the results is a string, but we know the data type which makes the conversion easy.

```
LPARAMETERS cAlias, oNode as MSXML2.IXMLDOMNode
LOCAL iFields, iField, oEx as Exception, cField, cFieldType, cValue

iFields = AFIELDS(aClientFields, cAlias)
* Skip the ID field
FOR iField = 2 TO iFields
   cField = aClientFields[iField,1]
   cFieldType = aClientFields[iField,2]
   TRY
      cValue = oNode.getElementsByTagName(LOWER(cField)).item[0].text
```

```
        DO CASE
            CASE INLIST(cFieldType, "Y", "B", "F", "N", "I")
                cValue = VAL(cValue)
            CASE cFieldType = "D"
                * In form yyyy-mm-dd
                cValue =
DATE(VAL(LEFT(cValue,4)),VAL(SUBSTR(cValue,6,2)),VAL(SUBSTR(cValue,9,2)))
            CASE cFieldType = "T"
                * In form yyyy-mm-dd 99:99:99
                cValue =
DATETIME(VAL(LEFT(cValue,4)),VAL(SUBSTR(cValue,6,2)),VAL(SUBSTR(cValue,9,2)), ;
                    VAL(SUBSTR(cValue,12,2)), VAL(SUBSTR(cValue,15,2)),
VAL(SUBSTR(cValue,18,2)))
        ENDCASE

        REPLACE (cField) WITH (cValue) IN (cAlias)

    CATCH TO oEx
        * Ignore the fact that some fields from the table will be buried deeper
    ENDTRY

NEXT
```

Reports

My biggest need is to see all of the time that's in FreshBooks that has not yet been billed, summarized by client, so I know which clients are ready to invoice. I don't want to invoice if I've only got an hour or two of time in for a client. One of these clients wants the report in a specific CSV format, so I created a subclass of FBReports.UnbilledTime just for them.

To create my custom reports, I first pull all of my FreshBooks data into my local Visual FoxPro tables, using the FBConnection.syncAll() method. This method calls the Sync() method for each type of entity.

Then I can use my FoxPro skills to analyze and report on the data just like any other FoxPro data. One particularly tricky aspect of the "Unbilled Hours" report is that the FreshBooks API does not have a "Billed" flag on time entries; in order to find out if a particular time entry has been billed, you actually have to compare it with the description on existing invoices' line items!

I created another classlibrary for reports, FBReports:

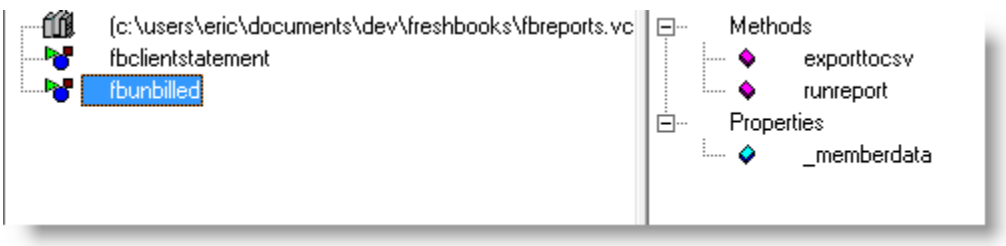


Figure 6 fbUnbilled class in FBReports class library

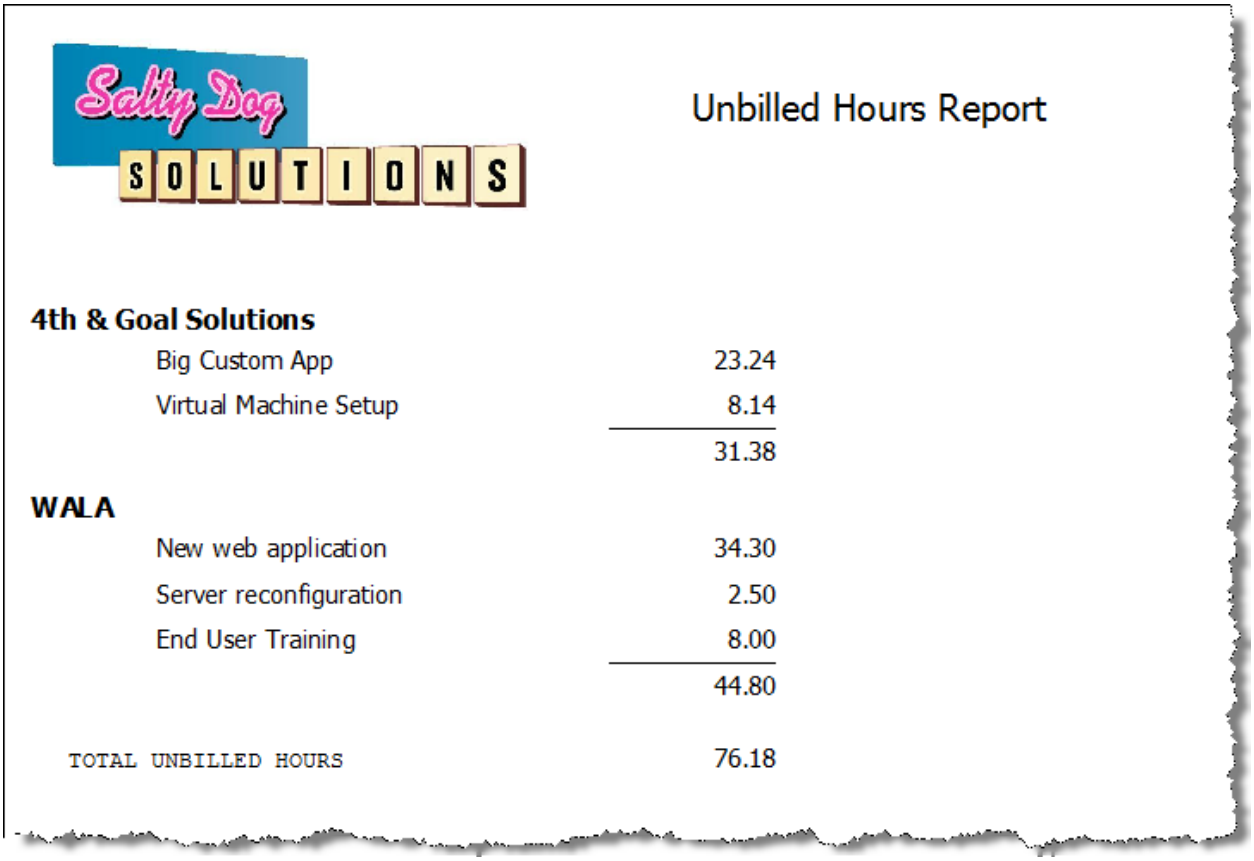


Figure 7 I now have a custom report from FreshBooks! Time to send invoices…

The source for these class libraries are available in the session materials. Please peruse that code to get the details about to integrate Visual FoxPro with your web app via LibCurl.

Summary

This session was intended to turn your thinking upside-down about the way Visual FoxPro interacts with the web. For a long time we've taken advantage of web services to extend the functionality of our Visual FoxPro applications. Now we can think of ways to extend the functionality of our web applications using Visual FoxPro.

In this session we covered what web apps are, took a look at various way to authenticate and use a web app's APIs, learned about curl.exe, libCurl.dll, and libCurl.vcx to facilitate accessing web apps from within Visual FoxPro.

The world is moving to web apps, but that doesn't mean you have to give up control of your data. Hopefully you will start to think about the web apps you use not as monolith islands

of impenetrable data, but will now consider ways to integrate the features of those web apps into your own Visual FoxPro applications, perhaps even extending their functionality.

Eric Selje

Appendix A
Curl.exe command line arguments

```
Usage: curl [options...] <url>
Options: (H) means HTTP/HTTPS only, (F) means FTP only
    --anyauth       Pick "any" authentication method (H)
 -a/--append        Append to target file when uploading (F/SFTP)
    --basic         Use HTTP Basic Authentication (H)
    --cacert <file> CA certificate to verify peer against (SSL)
    --capath <directory> CA directory to verify peer against (SSL)
 -E/--cert <cert[:passwd]> Client certificate file and password (SSL)
    --cert-type <type> Certificate file type (DER/PEM/ENG) (SSL)
    --ciphers <list> SSL ciphers to use (SSL)
    --compressed    Request compressed response (using deflate or gzip)
 -K/--config <file> Specify which config file to read
    --connect-timeout <seconds> Maximum time allowed for connection
 -C/--continue-at <offset> Resumed transfer offset
 -b/--cookie <name=string/file> Cookie string or file to read cookies from (H)
 -c/--cookie-jar <file> Write cookies to this file after operation (H)
    --create-dirs   Create necessary local directory hierarchy
    --crlf          Convert LF to CRLF in upload
    --crlfile <file> Get a CRL list in PEM format from the given file
 -d/--data <data>   HTTP POST data (H)
    --data-ascii <data>  HTTP POST ASCII data (H)
    --data-binary <data> HTTP POST binary data (H)
    --data-urlencode <name=data/name@filename> HTTP POST data url encoded (H)
    --digest        Use HTTP Digest Authentication (H)
    --disable-eprt  Inhibit using EPRT or LPRT (F)
    --disable-epsv  Inhibit using EPSV (F)
 -D/--dump-header <file> Write the headers to this file
    --egd-file <file> EGD socket path for random data (SSL)
    --engine <eng>  Crypto engine to use (SSL). "--engine list" for list
 -f/--fail          Fail silently (no output at all) on HTTP errors (H)
 -F/--form <name=content> Specify HTTP multipart POST data (H)
    --form-string <name=string> Specify HTTP multipart POST data (H)
```

```
   --ftp-account <data> Account data to send when requested by server (F)
   --ftp-alternative-to-user <cmd> String to replace "USER [name]" (F)
   --ftp-create-dirs Create the remote dirs if not present (F)
   --ftp-method [multicwd/nocwd/singlecwd] Control CWD usage (F)
   --ftp-pasv      Use PASV/EPSV instead of PORT (F)
-P/--ftp-port <address> Use PORT with address instead of PASV (F)
   --ftp-skip-pasv-ip Skip the IP address for PASV (F)
   --ftp-pret      Send PRET before PASV (for drftpd) (F)
   --ftp-ssl-ccc   Send CCC after authenticating (F)
   --ftp-ssl-ccc-mode [active/passive] Set CCC mode (F)
   --ftp-ssl-control Require SSL/TLS for ftp login, clear for transfer (F)
-G/--get           Send the -d data with a HTTP GET (H)
-g/--globoff       Disable URL sequences and ranges using {} and []
-H/--header <line> Custom header to pass to server (H)
-I/--head          Show document info only
-h/--help          This help text
   --hostpubmd5 <md5> Hex encoded MD5 string of the host public key. (SSH)
-0/--http1.0       Use HTTP 1.0 (H)
   --ignore-content-length  Ignore the HTTP Content-Length header
-i/--include       Include protocol headers in the output (H/F)
-k/--insecure      Allow connections to SSL sites without certs (H)
   --interface <interface> Specify network interface/address to use
-4/--ipv4          Resolve name to IPv4 address
-6/--ipv6          Resolve name to IPv6 address
-j/--junk-session-cookies Ignore session cookies read from file (H)
   --keepalive-time <seconds> Interval between keepalive probes
   --key <key>     Private key file name (SSL/SSH)
   --key-type <type> Private key file type (DER/PEM/ENG) (SSL)
   --krb <level>   Enable Kerberos with specified security level (F)
   --libcurl <file> Dump libcurl equivalent code of this command line
   --limit-rate <rate> Limit transfer speed to this rate
-J/--remote-header-name Use the header-provided filename (H)
-l/--list-only     List only names of an FTP directory (F)
   --local-port <num>[-num] Force use of these local port numbers
-L/--location      Follow Location: hints (H)
   --location-trusted Follow Location: and send auth to other hosts (H)
```

```
-M/--manual        Display the full manual
   --mail-from <from> Mail from this address
   --mail-rcpt <to> Mail to this receiver(s)
   --max-filesize <bytes> Maximum file size to download (H/F)
   --max-redirs <num> Maximum number of redirects allowed (H)
-m/--max-time <seconds> Maximum time allowed for the transfer
   --negotiate     Use HTTP Negotiate Authentication (H)
-n/--netrc         Must read .netrc for user name and password
   --netrc-optional Use either .netrc or URL; overrides -n
-N/--no-buffer     Disable buffering of the output stream
   --no-keepalive  Disable keepalive use on the connection
   --no-sessionid  Disable SSL session-ID reusing (SSL)
   --noproxy       Comma-separated list of hosts which do not use proxy
   --ntlm          Use HTTP NTLM authentication (H)
-o/--output <file> Write output to <file> instead of stdout
   --pass  <pass>  Pass phrase for the private key (SSL/SSH)
   --post301       Do not switch to GET after following a 301 redirect (H)
   --post302       Do not switch to GET after following a 302 redirect (H)
-#/--progress-bar  Display transfer progress as a progress bar
   --proto <protocols>       Enable/disable specified protocols
   --proto-redir <protocols> Enable/disable specified protocols on redirect
-x/--proxy <host[:port]> Use HTTP proxy on given port
   --proxy-anyauth Pick "any" proxy authentication method (H)
   --proxy-basic   Use Basic authentication on the proxy (H)
   --proxy-digest  Use Digest authentication on the proxy (H)
   --proxy-negotiate Use Negotiate authentication on the proxy (H)
   --proxy-ntlm    Use NTLM authentication on the proxy (H)
-U/--proxy-user <user[:password]> Set proxy user and password
   --proxy1.0 <host[:port]> Use HTTP/1.0 proxy on given port
-p/--proxytunnel   Operate through a HTTP proxy tunnel (using CONNECT)
   --pubkey <key>  Public key file name (SSH)
-Q/--quote <cmd>   Send command(s) to server before file transfer (F/SFTP)
   --random-file <file> File for reading random data from (SSL)
-r/--range <range> Retrieve only the bytes within a range
   --raw           Pass HTTP "raw", without any transfer decoding (H)
-e/--referer       Referer URL (H)
```

```
-O/--remote-name   Write output to a file named as the remote file
   --remote-name-all Use the remote file name for all URLs
-R/--remote-time   Set the remote file's time on the local output
-X/--request <command> Specify request command to use
   --retry <num>   Retry request <num> times if transient problems occur
   --retry-delay <seconds> When retrying, wait this many seconds between each
   --retry-max-time <seconds> Retry only within this period
-S/--show-error    Show error. With -s, make curl show errors when they occur
-s/--silent        Silent mode. Don't output anything
   --socks4 <host[:port]> SOCKS4 proxy on given host + port
   --socks4a <host[:port]> SOCKS4a proxy on given host + port
   --socks5 <host[:port]> SOCKS5 proxy on given host + port
   --socks5-hostname <host[:port]> SOCKS5 proxy, pass host name to proxy
   --socks5-gssapi-service <name> SOCKS5 proxy service name for gssapi
   --socks5-gssapi-nec  Compatibility with NEC SOCKS5 server
-Y/--speed-limit   Stop transfer if below speed-limit for 'speed-time' secs
-y/--speed-time    Time needed to trig speed-limit abort. Defaults to 30
   --ssl           Try SSL/TLS (FTP, IMAP, POP3, SMTP)
   --ssl-reqd      Require SSL/TLS (FTP, IMAP, POP3, SMTP)
-2/--sslv2         Use SSLv2 (SSL)
-3/--sslv3         Use SSLv3 (SSL)
   --stderr <file> Where to redirect stderr. - means stdout
   --tcp-nodelay   Use the TCP_NODELAY option
-t/--telnet-option <OPT=val> Set telnet option
   --tftp-blksize <value> Set TFTP BLKSIZE option (must be >512)
-z/--time-cond <time> Transfer based on a time condition
-1/--tlsv1         Use TLSv1 (SSL)
   --trace <file>  Write a debug trace to the given file
   --trace-ascii <file> Like --trace but without the hex output
   --trace-time    Add time stamps to trace/verbose output
-T/--upload-file <file> Transfer <file> to remote site
   --url <URL>     Set URL to work with
-B/--use-ascii     Use ASCII/text transfer
-u/--user <user[:password]> Set server user and password
-A/--user-agent <string> User-Agent to send to server (H)
-v/--verbose       Make the operation more talkative
```

```
 -V/--version       Show version number and quit
 -w/--write-out <format> What to output after completion
 -q                 If used as the first parameter disables .curlrc
```

Appendix B – LibCurl.vcx Quick Guide

LibCurl Features:

- Wraps most of the libcurl functions and properties, so you can have the full power of the libcurl library available, if you want to use it. This requires some understanding of how libcurl works.
- Provides FTP methods like FtpUploadFile, FtpDownloadFile, FtpListDirectory, etc., so you can use the class in a simple way without the need to understand how libcurl works.
- FTP uses passive mode by default.
- Provides HTTP methods like HttpUploadFile, HttpDownloadFile, etc.
- Provides generic methods to use with all protocols, like UploadFile, DownloadFile, etc.

In the class Init() event the libcurl library is initialized and a curl easy handle is created, so there is no need to do that manually. In the Destroy event the curl easy handle is closed and the libcurl library is released.

The class wraps most of the libcurl API calls, for example:
The **CurlEasyPerform** method, among other things, calls the **curl_easy_perform** function in libcurl.
Most of the configuration of the libcurl handle is done with calls to the curl_easy_setopt function in libcurl. The most common setup options are already wrapped in class properties, for example:
CurlOptUserPwd
CurlOptUseSsl
CurlOptVerbose
For the options not wrapped, you can use the libcurl API CurlEasySetOptInteger, CurlEasySetOptInt64, and CurlEasySetOptString dll declares aliases.
Really Quick Guide
Forget about libcurl, just drag and drop the class into your form and then use the class methods, like FtpDownloadFile, FtpUploadFile, etc. You will have to set some optional properties that are not set by those methods. If you need them, like CurlOptUserPwd for example.

Examples

Properties, Events and Methods


CancelTransfer
Cancels the current active transfer.
PARAMETERS
None.
DESCRIPTION
-
RETURN VALUE
None.
REMARKS
-


CurlEasyEscape(cString)
URL encodes the given string.
PARAMETERS

cString. Character. Specifies the string to escape.

DESCRIPTION

This method converts the given input string to an URL encoded string and returns that as a new allocated string. All input characters that are not a-z, A-Z or 0-9 are converted to their "URL escaped" version (%NN where NN is a two-digit hexadecimal number).

RETURN VALUE

Character.

REMARKS

Use CurlEasyUnescape to do the reverse.

 CurlEasyHandle

This property stores the curl easy handle that is created internally in the INIT method.

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE

Read only.

REMARKS

Under normal circumstances, you should not touch this property.

 CurlEasyPerform()

Performs a transfer.

PARAMETERS

None. Internally the CurlEasyHandle property is used in the call to curl_easy_perform.

DESCRIPTION

This method will perform the transfer as described in the options.

You can do any amount of calls to CurlEasyPerform. If you intend to transfer more than one file, you are even encouraged to do so. libcurl will then attempt to re-use the same connection for the following transfers, thus making the operations faster, less CPU intense and using less network resources. Just note that you will have to use the **CtlOpt...** properties between the invokes to set options for the following CurlEasyPerform.

RETURN VALUE

nResultCode. Numeric. 0 means everything was ok, non-zero means an error occurred. You can use **CurlEasyStrError**(nResultCode) to get a readable error message.

REMARKS

Under normal circumstances you will not need to call this method directly, but instead use the **Ftp/Http/File** methods.

 CurlEasyReset()

Resets all options of a libcurl session handle.

PARAMETERS

None. Internally the CurlEasyHandle property is used in the call to curl_easy_reset.

DESCRIPTION

Re-initializes all options previously set on a specified CURL handle to the default values. This puts back the handle to the same state as it was in when it was just created.

It does not change the following information kept in the handle: live connections, the Session ID cache, the DNS cache, the cookies and shares.

RETURN VALUE

None.

REMARKS

Internally the class sets again the callbacks and the location of the certificates file.

CurlEasyStrError([nErrorNum])

Returns a string describing the numeric error code passed.

PARAMETERS

nErrorNum. Numeric. Optional. A curl error code obtained by calling **CurlEasyPerform** or by calling **CurlInfoPerformCode**

DESCRIPTION

Returns a string describing the error code passed in the argument nErrorNum.

RETURN VALUE

Character.

REMARKS

If no error code is passed, the last return code from CurlEasyPerform is used (CurlInfoPerformCode)

CurlEasyUnescape(cString)

URL decodes the given string.

PARAMETERS

cString. Character. Specifies the string to url decode.

DESCRIPTION

This function converts the given URL encoded input string to a "plain string" and returns that in an allocated memory area. All input characters that are URL encoded (%XX where XX is a two-digit hexadecimal number) are converted to their binary versions.

RETURN VALUE

Character.

REMARKS

Use CurlEasyEscape to do the reverse.

CurlInfoEffectiveUrl()

Returns the last used effective URL.

PARAMETERS

None.

DESCRIPTION

RETURN VALUE

Character.

REMARKS

CurlInfoPerformCode()

The last result code returned by CurlEasyPErform

PARAMETERS

None.

DESCRIPTION

RETURN VALUE

Numeric.
REMARKS
This method does not wrap a libcurl function, it was added as a convenience to use with functions that do not return the result code from CurlEasyPerform. For example, FtpGetFileSize returns the size of the remote file or -1 if it fails, you can then get the result code from FtpGetFileSize with this method.
See also CurlEasyStrError

 CurlInfoResponseCode()
The last received HTTP or FTP code. This will be zero if no server response code has been received.
PARAMETERS
None
DESCRIPTION
RETURN VALUE
Numeric.
REMARKS
-

 CurlOptAppend
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Set to TRUE to append to the remote file instead of overwriting it. This is only useful when uploading to an ftp site.
There is no need to set this property directly, use the FtpAppendFile and FtpAppendString methods.

 CurlOptAutoReferer
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
When enabled, libcurl will automatically set the Referrer: field in requests where it follows a Location: redirect.

 CurlOptCookie
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes

REMARKS
It will be used to set a cookie in the http request. The format of the string should be NAME=CONTENTS, where NAME is the cookie name and CONTENTS is what the cookie should contain.
If you need to set multiple cookies, you need to set them all using a single option and thus you need to concatenate them all in one single string. Set multiple cookies in one string like this: "name1=content1; name2=content2;" etc.
Note that this option sets the cookie header explicitly in the outgoing request(s). If multiple requests are done due to authentication, followed redirections or similar, they will all get this cookie passed on.
Using this option multiple times will only make the latest string override the previous ones.

 CurlOptCookieFile
Specifies a file that holds cookie data to read.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
The cookie data may be in Netscape / Mozilla cookie data format or just regular HTTP-style headers dumped to a file.
Given an empty or non-existing file or by passing the empty string (""), this option will enable cookies for this curl handle, making it understand and parse received cookies and then use matching cookies in future request.
If you use this option multiple times, you just add more files to read. Subsequent files will add more cookies.

 CurlOptCookieJar
Specifies a file where libcurl will store known cookies when curl_easy_cleanup is called.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
libcurl will write all internally known cookies to the specified file when curl_easy_cleanup is called. If no cookies are known, no file will be created. Specify "-" to instead have the cookies written to stdout.
Using this option also enables cookies for this session, so if you for example follow a location it will make matching cookies get sent accordingly.

If the cookie jar file can't be created or written to (when the curl_easy_cleanup(3) is called), libcurl will not and cannot report an error for this. Using CURLOPT_VERBOSE or CURLOPT_DEBUGFUNCTION will get a warning to display, but that is the only visible feedback you get about this possibly lethal situation.

 CurlOptCopyPostFields
Specifies the full data to post in an HTTP POST operation.

VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
Use the HttpPostFile and HttpPostString methods.
It behaves as the CURLOPT_POSTFIELDS option, but the original data is copied by the library, allowing the application to overwrite the original data after setting this option.
Because data is copied, care must be taken when using this option in conjunction with CurlOptPostFieldSize or CURLOPT_POSTFIELDSIZE_LARGE: If the size has not been set prior to CurlOptCopyPostFields, the data are assumed to be a NUL-terminated string; else the stored size informs the library about the data byte count to copy. In any case, the size must not be changed after CurlOptCopyPostFields, unless another CURLOPT_POSTFIELDS or CurlOptCopyPostFields option is issued. (Added in 7.17.1)

 CurlOptCrlf
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Convert Unix newlines to CRLF newlines on transfers. It seems that this is not working in libcurl.

 CurlOptCustomRequest
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
It will be used instead of GET or HEAD when doing an HTTP request, or instead of LIST or NLST when doing an ftp directory listing. This is useful for doing DELETE or other more or less obscure HTTP requests. Don't do this at will, make sure your server supports the command first.
Note that libcurl will still act and assume the keyword it would use if you didn't set your custom one is the one in use and it will act according to that.
Restore to the internal default by setting this to "".

 CurlOptDirListOnly
VALUE TYPE
Logical.
DEFAULT VALUE
FALSE

READ/WRITE
Yes
REMARKS
Used by FtpListDirectory, no need to use directly.
Set to TRUE to just list the names of files in a directory, instead of doing a full directory listing that would include file sizes, dates etc. This works for FTP and SFTP URLs.
This causes an FTP NLST command to be sent on an FTP server. Beware that some FTP servers list only files in their response to NLST; they might not include subdirectories and symbolic links.

 CurlOptFileTime
VALUE TYPE
Logical.
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Used by FtpGetDateTimeStamp, no need to use directly.

 CurlOptFollowLocation
VALUE TYPE
Logical.
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
If TRUE, tells the library to follow any Location: header that the server sends as part of an HTTP header. This means that the library will re-send the same request on the new location and follow new Location: headers all the way until no more such headers are returned. CurlOptMaxRedirs can be used to limit the number of redirects libcurl will follow.
Not used in FTP protocol.

 CurlOptFtpAccount
Specifies the account data for FTP servers.
VALUE TYPE
Character.
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
When an FTP server asks for "account data" after user name and password has been provided, this data is sent off using the ACCT command.

 CurlOptFtpCreateMissingDirs
VALUE TYPE

Logical.
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
If TRUE, libcurl will attempt to create any remote directory that it fails to CWD into. CWD is the command that changes working directory. (Added in 7.10.7)
This setting also applies to SFTP connections. libcurl will attempt to create the remote directory if it can't obtain a handle to the target-location. The creation will fail if a file of the same name as the directory to create already exists or lack of permissions prevents creation.

 CurlOptFtpFileMethod
VALUE TYPE
Numeric.
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
This option controls what method libcurl should use to reach a file on a FTP(S) server. The argument should be one of the following alternatives:
CURLFTPMETHOD_MULTICWD (1)
libcurl does a single CWD operation for each path part in the given URL. For deep hierarchies this means very many commands. This is how RFC1738 says it should be done. This is the default but the slowest behavior.
CURLFTPMETHOD_NOCWD (2)
libcurl does no CWD at all. libcurl will do SIZE, RETR, STOR etc. and give a full path to the server for all these commands. This is the fastest behavior.
CURLFTPMETHOD_SINGLECWD (3)
libcurl does one CWD with the full target directory and then operates on the file "normally" (like in the multicwd case). This is somewhat more standards compliant than 'nocwd' but without the full penalty of 'multicwd'.

 CurlOptFtpPort
Specifies the IP address to use for the ftp PORT instruction.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
The PORT instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a host name, an network interface name (under Unix) or just a '-' letter to let the library use your systems default IP address. Default FTP operations are passive, and thus won't use PORT.
You disable PORT again and go back to using the passive version by setting this option to "".

CurlOptFtpResponseTimeOut

VALUE TYPE

Numeric.

DEFAULT VALUE

0

READ/WRITE

Yes

REMARKS

Specifies the timeout period (in seconds) on the amount of time that the server is allowed to take in order to generate a response message for a command before the session is considered hung. While curl is waiting for a response, this value overrides CurlOptTimeOut. It is recommended that if used in conjunction with CurlOptTimeOut, you set CurlOptFtpResponseTimeOut to a value smaller than CurlOptTimeOut. (Added in 7.10.8)

CurlOptFtpSslAuth

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE

Yes

REMARKS

Set to one of the values from below, to alter how libcurl issues "AUTH TLS" or "AUTH SSL" when FTP over SSL is activated (see CurlOptUseSsl CURLOPT_FTP_SSL).

CURLFTPAUTH_DEFAULT	0	Allow libcurl to decide.
CURLFTPAUTH_SSL	1	Try "AUTH SSL" first, and only if that fails try "AUTH TLS"
CURLFTPAUTH_TLS	2	Try "AUTH TLS" first, and only if that fails try "AUTH SSL"

CurlOptFtpSslCcc

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE

Yes

REMARKS

If enabled, this option makes libcurl use CCC (Clear Command Channel). It shuts down the SSL/TLS layer after authenticating. The rest of the control channel communication will be unencrypted. This allows NAT routers to follow the FTP transaction. Pass a long using one of the values below. (Added in 7.16.1)

CURLFTPSSL_CCC_NONE	0	Don't attempt to use CCC.
CURLFTPSSL_CCC_PASSIVE	1	Do not initiate the shutdown, but wait for the server to do it. Do not send a reply.
CURLFTPSSL_CCC_ACTIVE	2	Initiate the shutdown and wait for a reply.

 CurlOptHeader
VALUE TYPE
Logical.
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
If TRUE, it tells the library to include the header in the body output. This is only relevant for protocols that actually have headers preceding the data (like HTTP).

 CurlOptHttpAuth
VALUE TYPE
Numeric
DEFAULT VALUE
1
READ/WRITE
Yes
REMARKS
Tells libcurl what authentication method(s) you want it to use. The available bits are listed below. If more than one bit is set, libcurl will first query the site to see what authentication methods it supports and then pick the best one you allow it to use. For some methods, this will induce an extra network round-trip. Set the actual name and password with the CurlOptUserPwd option. (Added in 7.10.6)
CURLAUTH_BASIC (1)
HTTP Basic authentication. This is the default choice, and the only method that is in wide-spread use and supported virtually everywhere. This is sending the user name and password over the network in plain text, easily captured by others.
CURLAUTH_DIGEST (2)
HTTP Digest authentication. Digest authentication is defined in RFC2617 and is a more secure way to do authentication over public networks than the regular old-fashioned Basic method.
CURLAUTH_GSSNEGOTIATE (4)
HTTP GSS-Negotiate authentication. The GSS-Negotiate (also known as plain "Negotiate") method was designed by Microsoft and is used in their web applications. It is primarily meant as a support for Kerberos5 authentication but may be also used along with another authentication methods. For more information see IETF draft draft-brezak-spnego-http-04.txt.
You need to build libcurl with a suitable GSS-API library for this to work.
CURLAUTH_NTLM (8)
HTTP NTLM authentication. A proprietary protocol invented and used by Microsoft. It uses a challenge-response and hash concept similar to Digest, to prevent the password from being eavesdropped.
CURLAUTH_ANY (15)
This is a convenience macro that sets all bits and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure.
CURLAUTH_ANYSAFE (14)
This is a convenience macro that sets all bits except Basic and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure.

 CurlOptHttpGet

VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
This forces the HTTP request to get back to GET. usable if a POST, HEAD, PUT or a custom request have been used previously using the same curl handle.
When set to TRUE, it will automatically set CurlOptNoBody to FALSE.

 CurlOptHttpHeader
List of HTTP headers to pass to the server in your HTTP request. (Separated by the "|" character)
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
If you add a header with no contents as in 'Accept:' (no data on the right side of the colon), the internally used header will get disabled. Thus, using this option you can add new headers, replace internal headers and remove internal headers. To add a header with no contents, make the contents be two quotes: "".
Pass a "" to this to reset back to no custom headers.
The most commonly replaced headers have "shortcuts" in the options CurlOptCookie, CurlOptUserAgent and CurlOptReferer

---

 CurlOptHttpProxyTunnel
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Set the parameter to TRUE to make the library tunnel all operations through a given HTTP proxy. There is a big difference between using a proxy and to tunnel through it. If you don't know what this means, you probably don't want this tunneling option.

CurlOptInterface
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes

REMARKS
This sets the interface name to use as outgoing network interface. The name can be an interface name, an IP address or a host name.

 CurlOptLocalPort
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
This sets the local port number of the socket used for connection. This can be used in combination with CurlOptInterface and you are recommended to use CurlOptLocalPortRange as well when this is set. Note that port numbers are only valid 1 - 65535.

 CurlOptLocalPortRange
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
This is the number of attempts libcurl should do to find a working local port number. It starts with the given CurlOptLocalPort and adds one to the number for each retry. Setting this to 1 or below will make libcurl do only one try for the exact port number. Note that port numbers by nature are scarce resources that will be busy at times so setting this value to something too low might cause unnecessary connection setup failures.

 CurlOptMaxRecvSpeed
Specifies the maximum download speed.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
If a download exceeds this speed on cumulative average during the transfer, the transfer will pause to keep the average rate less than or equal to the parameter value. Defaults to unlimited speed.

 CurlOptMaxRedirs
Specifies the limit of redirections.
VALUE TYPE
Numeric.
DEFAULT VALUE

-1
READ/WRITE
Yes
REMARKS
The set number will be the redirection limit. If that many redirections have been followed, the next redirect will cause an error (CURLE_TOO_MANY_REDIRECTS). This option only makes sense if the CurlOptFollowLocation is used at the same time. Added in 7.15.1: Setting the limit to 0 will make libcurl refuse any redirect. Set it to -1 for an infinite number of redirects (which is the default)

 CurlOptMaxSendSpeed
Specifies the maximum upload speed.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
If an upload exceeds this speed on cumulative average during the transfer, the transfer will pause to keep the average rate less than or equal to the parameter value. Defaults to unlimited speed.

 CurlOptNoBody
If TRUE, tells the library to not include the body-part in the output.
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
This is only relevant for protocols that have separate header and body parts. On HTTP(S) servers, this will make libcurl do a HEAD request.

 CurlOptPort
Specifies the remote port number to connect to.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
Specifies the remote port number to connect to, instead of the one specified in the URL or the default port for the used protocol.

 CurlOptPost
If TRUE, tells the library to do a regular HTTP post.

VALUE TYPE
Logical
DEFAULT VALUE
TRUE
READ/WRITE
Yes
REMARKS
Use the HttpPostString and HttpPostFile methods instead of using this property directly.
If TRUE, tells the library to do a regular HTTP post. This will also make the library use a "Content-Type: application/x-www-form-urlencoded" header. (This is by far the most commonly used POST method).
Use one of CURLOPT_POSTFIELDS or CurlOptCopyPostFields options to specify what data to post and CurlOptPostFieldSize or CURLOPT_POSTFIELDSIZE_LARGE to set the data size.
Optionally, you can provide data to POST using the CURLOPT_READFUNCTION and CURLOPT_READDATA options but then you must make sure to not set CURLOPT_POSTFIELDS to anything but NULL. When providing data with a callback, you must transmit it using chunked transfer-encoding or you must set the size of the data with the CurlOptPostFieldSize or CURLOPT_POSTFIELDSIZE_LARGE option. To enable chunked encoding, you simply pass in the appropriate Transfer-Encoding header, see the post-callback.c example.
You can override the default POST Content-Type: header by setting your own with CurlOptHttpHeader.
Using POST with HTTP 1.1 implies the use of a "Expect: 100-continue" header. You can disable this header with CurlOptHttpHeader as usual.
If you use POST to a HTTP 1.1 server, you can send data without knowing the size before starting the POST if you use chunked encoding. You enable this by adding a header like "Transfer-Encoding: chunked" with CurlOptHttpHeader. With HTTP 1.0 or without chunked transfer, you must specify the size in the request.
When setting CurlOptPost to TRUE, it will automatically set CurlOptNoBody to FALSE (since 7.14.1).
If you issue a POST request and then want to make a HEAD or GET using the same re-used handle, you must explicitly set the new request type using CurlOptNoBody or CurlOptHttpGet or similar.

CurlOptPost301
Specifies if RFC 2616/10.3.2 should be followed in redirections.
VALUE TYPE
Logical
DEFAULT VALUE
TRUE
READ/WRITE
Yes
REMARKS
If TRUE tells the library to respect RFC 2616/10.3.2 and not convert POST requests into GET requests when following a 301 redirection. The non-RFC behavior is ubiquitous in web browsers, so the library does the conversion by default to maintain consistency. However, a server may requires a POST to remain a POST after such a redirection. This option is meaningful only when setting CurlOptFollowLocation .

CurlOptPostFieldSize
Specifies the length of the CurlOptCopyPostFields property.
VALUE TYPE
Numeric

DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
If you want to post data to the server without letting libcurl do a strlen() to measure the data size, this option must be used. When this option is used you can post fully binary data, which otherwise is likely to fail. If this size is set to -1, the library will use strlen() to get the size.

 CurlOptPostQuote
List of FTP or SFTP commands to pass to the server after your ftp transfer request.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
The commands should be separated by the ASCII character 124 "|". Disable this operation again by setting an empty string to this option.

 CurlOptPreQuote
List of FTP commands to pass to the server after the transfer type is set.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
The commands should be separated by the ASCII character 124 "|". Disable this operation again by setting an empty string to this option.

 CurlOptProgress
Specifies if the TransferProgress event should be fired.
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Set to TRUE to use the TransferProgress event. The original libcurl option is called CURLOPT_NOPROGRESS. I removed the negation to make things easier.

 CurlOptProxy
Specifies the HTTP proxy to use.

VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
Set HTTP proxy to use. The parameter should be a string holding the host name or dotted IP address. To specify port number in this string, append :[port] to the end of the host name. The proxy string may be prefixed with [protocol]:// since any such prefix will be ignored. The proxy's port number may optionally be specified with the separate option CurlOptProxyPort.
When you tell the library to use an HTTP proxy, libcurl will transparently convert operations to HTTP even if you specify an FTP URL etc. This may have an impact on what other features of the library you can use, such as CurlOptQuote and similar FTP specifics that don't work unless you tunnel through the HTTP proxy. Such tunneling is activated with CurlOptHttpProxyTunnel.
libcurl respects the environment variables http_proxy, ftp_proxy, all_proxy etc., if any of those is set. The CurlOptProxy option does however override any possibly set environment variables.
Setting the proxy string to "" (an empty string) will explicitly disable the use of a proxy, even if there is an environment variable set for it.
Since 7.14.1, the proxy host string given in environment variables can be specified the exact same way as the proxy can be set with CurlOptProxy, include protocol prefix (http://) and embedded user + password.

 CurlOptProxyAuth
Specifies the proxy authentication method(s).
VALUE TYPE
Numeric
DEFAULT VALUE
?
READ/WRITE
Yes
REMARKS
Pass along as parameter, which is set to a bitmask, to tell libcurl what authentication method(s) you want it to use for your proxy authentication. If more than one bit is set, libcurl will first query the site to see what authentication methods it supports and then pick the best one you allow it to use. For some methods, this will induce an extra network round-trip. Set the actual name and password with the CurlOptProxyUserPwd option. The bitmask can be constructed by or'ing together the bits listed. As of this writing, only Basic, Digest and NTLM work.

CURLAUTH_NONE	0	None
CURLAUTH_ANY	15	This is a convenience macro that sets all bits and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure.
CURLAUTH_ANYSAFE	14	This is a convenience macro that sets all bits except Basic and thus makes libcurl pick any it finds suitable. libcurl will automatically select the one it finds most secure.

CURLAUTH_BASIC	1	HTTP Basic authentication. This is the default choice, and the only method that is in wide-spread use and supported virtually everywhere. This is sending the user name and password over the network in plain text, easily captured by others.
CURLAUTH_DIGEST	2	HTTP Digest authentication. Digest authentication is defined in RFC2617 and is a more secure way to do authentication over public networks than the regular old-fashioned Basic method.
CURLAUTH_GSSNEGOTIATE	4	HTTP GSS-Negotiate authentication. The GSS-Negotiate (also known as plain "Negotiate") method was designed by Microsoft and is used in their web applications. It is primarily meant as a support for Kerberos5 authentication but may be also used along with another authentication methods. For more information see IETF draft draft-brezak-spnego-http-04.txt.
CURLAUTH_NTLM	8	HTTP NTLM authentication. A proprietary protocol invented and used by Microsoft. It uses a challenge-response and hash concept similar to Digest, to prevent the password from being eavesdropped.

 CurlOptProxyPort
Specifies the proxy port to connect to unless it is specified in the proxy string CurlOptProxy.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
-

 CurlOptProxyType
Specifies the type of the proxy.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE
Yes
REMARKS
Available options for this are

CURLPROXY_HTTP	0
CURLPROXY_SOCKS4	4
CURLPROXY_SOCKS5	5
CURLPROXY_SOCKS4A	6

CURLPROXY_SOCKS5_HOSTNAME 7

The HTTP type is default.

 CurlOptProxyUserPwd

Specifies the user name and password for the HTTP proxy. Should be in the form "user:password"

VALUE TYPE

Character

DEFAULT VALUE

None

READ/WRITE

Yes

REMARKS

String, which should be "user name:password" to use for the connection to the HTTP proxy. Use CurlOptProxyAuth to decide authentication method.

 CurlOptQuote

Commands to execute before any other commands are issued.

VALUE TYPE

Character

DEFAULT VALUE

None

READ/WRITE

Yes

REMARKS

Pass a string list of FTP or SFTP commands to pass to the server prior to your ftp request. This will be done before any other commands are issued (even before the CWD command for FTP).
The commands should be separated by the ASCII character 124 "|"
The set of valid FTP commands depends on the server (see RFC959 for a list of mandatory commands).
The valid SFTP commands are: chgrp, chmod, chown, ln, mkdir, pwd, rename, rm, rmdir, symlink (see curl (1)) (SFTP support added in 7.16.3)

 CurlOptRange

Specifies the range in bytes to transfer.

VALUE TYPE

Character

DEFAULT VALUE

None

READ/WRITE

Yes

REMARKS

String, which should contain the specified range you want. It should be in the format "X-Y", where X or Y may be left out. HTTP transfers also support several intervals, separated with commas as in "X-Y,N-M". Using this kind of multiple intervals will cause the HTTP server to send the response document in pieces (using standard MIME separation techniques). Pass a "" to this option to disable the use of ranges. Ranges work on HTTP, FTP and FILE (since 7.18.0) transfers only.

 CurlOptReferer

Specifies the Referrer: header in the http request sent to the remote server.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
This can be used to fool servers or scripts. You can also set any custom header with CurlOptHttpHeader.

 CurlOptResumeFrom
Offset in number of bytes that you want the transfer to start from.
VALUE TYPE
Numeric
DEFAULT VALUE
0
READ/WRITE: Yes
REMARKS
Set this option to 0 to make the transfer start from the beginning of the file(effectively disabling resume). For FTP, set this option to -1 to make the transfer start from the end of the target file (useful to continue an interrupted upload).

 CurlOptSslVerifyHost
Specifies if the server certificate Common Name field should match the host name in the URL.
VALUE TYPE
Numeric
DEFAULT VALUE
2
READ/WRITE
Yes
REMARKS
This option determines whether libcurl verifies that the server cert is for the server it is known as. When negotiating an SSL connection, the server sends a certificate indicating its identity. When CurlOptSslVerifyHost is 2, that certificate must indicate that the server is the server to which you meant to connect, or the connection fails.
Curl considers the server the intended one when the Common Name field or a Subject Alternate Name field in the certificate matches the host name in the URL to which you told Curl to connect.
When the value is 1, the certificate must contain a Common Name field, but it doesn't matter what name it says. (This is not ordinarily a useful setting).
When the value is 0, the connection succeeds regardless of the names in the certificate.
The DEFAULT VALUE: is 2.
The checking this option controls is of the identity that the server claims. The server could be lying. To control lying, see CurlOptSslVerifyPeer.
NOTE: If using a self-signed certificate, just set this to 0 if you are not sure if the host name matches the name in the certificate.

 CurlOptSslVerifyPeer

Basically specifies if the server certificate should be verified. Set to FALSE for self-signed certificates.

VALUE TYPE

Logical

DEFAULT VALUE

TRUE

READ/WRITE

Yes

REMARKS

This option determines whether curl verifies the authenticity of the peer's certificate. A value of TRUE means curl verifies; FALSE means it doesn't. The default is TRUE.

When negotiating an SSL connection, the server sends a certificate indicating its identity. Curl verifies whether the certificate is authentic, i.e. that you can trust that the server is who the certificate says it is. This trust is based on a chain of digital signatures, rooted in certification authority (CA) certificates you supply.

When CurlOptSslVerifyPeer is TRUE, and the verification fails to prove that the certificate is authentic, the connection fails. When the option is FALSE, the connection succeeds regardless.

Authenticating the certificate is not by itself very useful. You typically want to ensure that the server, as authentically identified by its certificate, is the server you mean to be talking to. Use CurlOptVerifyHost to control that.

NOTE: If using a self-signed certificate, set this to FALSE

 CurlOptSslVersion

Specifies the version of SSL/TLS to attempt to use.

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE

Yes

REMARKS

The available options are:

CURL_SSLVERSION_DEFAULT	0	The default action. This will attempt to figure out the remote SSL protocol version, i.e. either SSLv3 or TLSv1 (but not SSLv2, which became disabled by default with 7.18.1).
CURL_SSLVERSION_TLSv1	1	Force TLSv1
CURL_SSLVERSION_SSLv2	2	Force SSLv2
CURL_SSLVERSION_SSLv3	3	Force SSLv3

 CurlOptTimeOut

Specifies the maximum time in seconds that you allow the libcurl transfer operation to take.

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE: Yes

REMARKS
Pass along as parameter containing the maximum time in seconds that you allow the libcurl transfer operation to take. Normally, name lookups can take a considerable time and limiting operations to less than a few minutes risk aborting perfectly normal operations. This option will cause curl to use the SIGALRM to enable time-outing system calls.

 CurlOptTransferText
Specifies if ASCII mode or BINARY mode should be used for FTP transfers
VALUE TYPE
Logical
DEFAULT VALUE: FALSE
READ/WRITE
Yes
REMARKS
If TRUE, tells the library to use ASCII mode for ftp transfers, instead of the default binary transfer. libcurl does not do a complete ASCII conversion when doing ASCII transfers over FTP. This is a known limitation/flaw that nobody has rectified. libcurl simply sets the mode to ascii and performs a standard transfer.

 CurlOptUnrestrictedAuth
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
-

 CurlOptUpload
Specifies if the connection should attempt an upload.
VALUE TYPE: Logical
DEFAULT VALUE: FALSE
READ/WRITE: Yes
REMARKS
A parameter set to TRUE tells the library to prepare for an upload. Under normal circumstances, there is no need to set this property, since each method will set it to the desired value. For example FtpDownloadFile, FtpUploadFile, etc.

 CurlOptUrl
Specifies the URL to use.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE: Yes
REMARKS

The actual URL to deal with. The parameter should be a string.
If the given URL lacks the protocol part ("http://" or "ftp://" etc.), it will attempt to guess which protocol to use based on the given host name. If the given protocol of the set URL is not supported, libcurl will return on error (CURLE_UNSUPPORTED_PROTOCOL) when you call CurlEasyPerform . Use curl_version_info(3) for detailed info on which protocols that are supported.
The string given to CurlOptUrl must be url-encoded and following the RFC 2396 (http://curl.haxx.se/rfc/rfc2396.txt).
CurlOptUrl is the only option that must be set before CurlEasyPerform is called.

 CurlOptUserAgent
Specifies the string to use in the User-Agent header.
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
String. It will be used to set the User-Agent: header in the http request sent to the remote server. This can be used to fool servers or scripts. You can also set any custom header with CURLOPT_HTTPHEADER. This is an example User Agent string: "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

 CurlOptUserPwd
Specifies the user name and password to use for the connection. The format is "user:password"
VALUE TYPE
Character
DEFAULT VALUE
None
READ/WRITE
Yes
REMARKS
String, which should be "user name:password" to use for the connection. Use CURLOPT_HTTPAUTH to decide authentication method.
When using NTLM, you can set domain by prepending it to the user name and separating the domain and name with a forward (/) or backward slash (\). Like this: "domain/user:password" or "domain\user:password". Some HTTP servers (on Windows) support this style even for Basic authentication.
When using HTTP and CURLOPT_FOLLOWLOCATION, libcurl might perform several requests to possibly different hosts. libcurl will only send this user and password information to hosts using the initial host name (unless CURLOPT_UNRESTRICTED_AUTH is set), so if libcurl follows locations to other hosts it will not send the user and password to those. This is enforced to prevent accidental information leakage.

 CurlOptUseSsl
VALUE TYPE
Numeric
DEFAULT VALUE
0

READ/WRITE
Yes
REMARKS
Pass a number using one of the values from below, to make libcurl use your desired level of SSL for the ftp transfer.

CURLUSESSL_NONE	0	Don't attempt to use SSL.
CURLUSESSL_TRY	1	Try using SSL, proceed as normal otherwise.
CURLUSESSL_CONTROL	2	Require SSL for the control connection or fail with CURLE_USE_SSL_FAILED (64)
CURLUSESSL_ALL	3	Require SSL for all communication or fail with CURLE_USE_SSL_FAILED (64).

 CurlOptVerbose
VALUE TYPE
Logical
DEFAULT VALUE
FALSE
READ/WRITE
Yes
REMARKS
Set the parameter to TRUE to get the library to display a lot of verbose information about its operations. Very useful for libcurl and/or protocol debugging and understanding. The verbose information will be sent to the CurlDebugBuffer property
You hardly ever want this set in production use, you will almost always want this when you debug/report problems.

 CurlVersion()
Returns the libcurl version in a human readable form.
PARAMETERS
None.
DESCRIPTION
-
RETURN VALUE
Character.
REMARKS
-

 FileDownloadFile(cRemoteFile, cLocalFile, [nResumeFrom], [cRange])
Downloads a remote file to a local file using the FILE protocol.
REMARKS
See FtpDownloadFile

 FileDownloadString(cRemoteFile, [nResumeFrom], [cRange])
Downloads a remote file, returns the contents as a character string using the FILE protocol.
REMARKS
See FtpDownloadString

FileUploadFile(cRemoteFile, cLocalFile)
Uploads a local file to a remote file using the FILE protocol.
REMARKS
See FtpUploadFile

FileUploadString(cRemoteFile, cString)
Uploads a string to a remote file.
REMARKS
See FtpUploadString

FormatByteSize(nBytes)
Pass a numeric value of Bytes, returns a character string formatted as Bytes, KB, MB, GB, TB
PARAMETERS
nBytes. Numeric.
DESCRIPTION
-
RETURN VALUE
Character.
REMARKS
Can be used to show info to the user, in the FtpProgressEvent.

FormatTimeSeconds(nSeconds)
Pass a numeric value in seconds, returns a character string in the format 99 h 99 m 99 s
PARAMETERS
nSeconds. Numeric.
DESCRIPTION
-
RETURN VALUE
Character.
REMARKS
Can be used to show info to the user, in the FtpProgressEvent.

FtpAppendFile(cRemoteFile, cLocalFile)
Represents the FTP command APPE that is used to append a file to an existing remote file.
PARAMETERS
cRemoteFile. Character. Specifies the full URL of the remote file.
cLocalFile. Character. Specifies the name of the local file.
DESCRIPTION
This method always appends the local file to the end of the remote file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
Specify the full URL of the remote file, i.e. ftp://ftp.example.com/directory/file.txt

---

FtpAppendString(cRemoteFile, cString)
Represents the FTP command APPE that is used to append a file to an existing remote file.

PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
cString. Character. Specifies the data to append.
DESCRIPTION
Append the character string cString to the end of the remote file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
-

FtpDeleteFile(cRemoteFile)
Represents the FTP command DELE used to delete files on an FTP server.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
DESCRIPTION
Deletes a remote file from an FTP server. The full URL to the file should be passed.
RETURN VALUE
nResultCode. Numeric.
REMARKS
-

FtpDownloadFile(cRemoteFile, cLocalFile, [nResumeFrom], [cRange])
Downloads a remote file to a local file using the FTP/FTPS protocol.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
cLocalFile. Character. Specifies the name of the local file.
nResumeFrom. Numeric. Optional. Specifies the offset in number of bytes that you want the transfer to start from. Set this option to 0 to make the transfer start from the beginning (effectively disabling resume). For FTP, set this option to -1 to make the transfer start from the end of the target file.
cRange. Character. Optional. This should be in the format "X-Y", where X or Y may be left out. HTTP transfers also support several intervals, separated with commas as in "X-Y,N-M". Using this kind of multiple intervals will cause the HTTP server to send the response document in pieces (using standard MIME separation techniques).
DESCRIPTION
Downloads a file from an FTP server to a local file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
You can use the CurlOptResumeFrom property instead of the nResumeFrom parameter.
While the file is downloading, you can call FtpCancel to cancel the transfer in progress.
When using the FileDownloadFile method, there is no need to specify the file protocol in the url (file://)

FtpDownloadString(cRemoteFile, [nResumeFrom], [cRange])
Downloads a remote file, returns the contents as a character string. using the FTP/FTPS protocol.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.

nResumeFrom. Numeric. Optional. Specifies the offset in number of bytes that you want the transfer to start from. Set this option to 0 to make the transfer start from the beginning (effectively disabling resume). For FTP, set this option to -1 to make the transfer start from the end of the target file.
cRange. Character. Optional. This should be in the format "X-Y", where X or Y may be left out. HTTP transfers also support several intervals, separated with commas as in "X-Y,N-M". Using this kind of multiple intervals will cause the HTTP server to send the response document in pieces (using standard MIME separation techniques).

DESCRIPTION

Downloads a remote file and returns the downloaded file data as a string.

RETURN VALUE

String

REMARKS

Do not use with files larger than 16777184 bytes due to a VFP limitation. In fact, with files larger than a few KB FtpDownloadFile is the way to go.
When using the FileDownloadString method, there is no need to specify the file protocol in the url (file://)

 FtpFileCount

VALUE TYPE

Numeric

DEFAULT VALUE

0

READ/WRITE

Read only.

REMARKS

Specifies the number of files and directories returned by the last FtpList method call.

 FtpFiles(FtpFileCount)

VALUE TYPE

Array

DEFAULT VALUE

?

READ/WRITE

Read only.

REMARKS

This array has FtpFileCount rows and four columns, contains information on the last call to FtpList:
Column 1 File name, Character
Column 2 File size, Numeric
Column 3 DateTime last modified, DateTime
Column 4 File attributes, Character, "A" for file or "D" for directory

---

 FtpGetDateTimeStamp(cRemoteFile)

Attempts to retrieve the date-time stamp of a remote file using the FTP/FTPS protocol.

PARAMETERS

cRemoteFile. Character. Specifies the URL of the remote file.

DESCRIPTION

Gets the date and time of a remote file. This is GMT time.

RETURN VALUE
DateTime.
REMARKS
On failure, returns an empty DateTime value.

FtpGetFeatures(cUrl)
Represents the FTP command FEAT that is used to retrieve the features of an FT server.
PARAMETERS
cURL. Character.
DESCRIPTION
Returns a list of "|" separated features, for example:
"MDTM|REST STREAM|SIZE|MLST type*;size*;modify*;|MLSD|AUTH SSL|AUTH TLS|UTF8|CLNT|MFMT"
RETURN VALUE
Character.
REMARKS
Used by the class itself to find out if the remote server supports MLST.

FtpGetFileSize(cRemoteFile)
Attempts to get the size of a remote file using the FTP/FTPS protocol.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
DESCRIPTION
Get the size of a remote file.
RETURN VALUE
Numeric
REMARKS
Returns -1 on failure.

FtpList(cUrl)
Fills the FtpFiles array property with information about the contents of an FTP directory
PARAMETERS
String URL
DESCRIPTION
This method first calls the FtpGetFeatures method to find out if the FTP server supports the MLSD command. If it does, it then sends an MLSD command and parses the result, filling the FtpFiles array property.
If the FTP server does not support the MLSD command, then the FtpListDirectory method is called to get the list of files and folders, and then FtpListDirectoryDetails is called, just to identify files from directories. Then FtpGetFileSize and FtpGetDateTimeStamp is called for each file/directory, and the FtpFiles array property is filled. All this can take quite some time.
RETURN VALUE
Numeric. The total number of files and directories found.
REMARKS
The class makes some assumptions about the format of the replies of the MLST, LIST and NLST commands. This may be wrong.

FtpListDirectory(cUrl)
Represents the FTP command NLIST that gets a short listing of the files.
PARAMETERS
String URL
DESCRIPTION
This command just returns the raw directory listing sent from the FTP server as a character string. No parsing is done by the class.
RETURN VALUE
Character.
REMARKS
The URL should point to a directory, not a file.

FtpListDirectoryDetails(cUrl)
Represents the FTP command LIST that gets a detailed listing of the files.
PARAMETERS
String URL
DESCRIPTION
This command just returns the raw directory listing sent from the FTP server as a character string. No parsing is done by the class.
RETURN VALUE
Character.
REMARKS
The URL should point to a directory, not a file.

FtpMakeDirectory(cRemoteDir)
Represents the FTP command MKD that creates a directory.
PARAMETERS
cRemoteDir. Character. Specifies the URL of the remote directory.
DESCRIPTION
Creates a remote directory in an FTP server.
RETURN VALUE
nResultCode. Numeric.
REMARKS
-

FtpNoop(cUrl)
Represents the FTP command NOOP
PARAMETERS
String
DESCRIPTION
Just sends a NOOP command to an FTP server.
RETURN VALUE
nResultCode. Numeric.
REMARKS
Can be used to keep a connection alive.

FtpRemoveDirectory(cRemoteDir)

Represents the FTP command RMD that removes a directory
PARAMETERS
String
DESCRIPTION
Deletes a remote directory in an FTP server.
RETURN VALUE
nResultCode. Numeric.
REMARKS
-

FtpRename(cRemoteFileOldName, cRemoteFileNewName)
Represents the FTP commands RNFR and RNTO that rename a file.
PARAMETERS
cRemoteFileOldName. Character. Specifies the old URL of the remote file.
cRemoteFileNewName. Character. Specifies the new URL of the remote file.
DESCRIPTION
Renames a remote FTP server file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
-

FtpUploadFile(cRemoteFile, cLocalFile, [nResumeFrom])
Uploads a local file to a remote file using the FTP/FTPS protocol.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
cLocalFile. Character. Specifies the name of the local file.
nResumeFrom. Numeric. Optional. Specifies the offset in number of bytes that you want the transfer to start from. Set this option to 0 to make the transfer start from the beginning (effectively disabling resume). For FTP, set this option to -1 to make the transfer start from the end of the target file.
DESCRIPTION
Uploads a local file to an FTP server file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
For FTP, pass -1 in nResumeFrom to make the transfer start from the end of the target file (useful to continue an interrupted upload).
When using the FileUploadFile method, there is no need to specify the file protocol in the url (file://)

FtpUploadString(cRemoteFile, cString, [nResumeFrom])
Uploads a string to a remote file using the FTP/FTPS protocol.
PARAMETERS
cRemoteFile. Character. Specifies the URL of the remote file.
cString. Character. Specifies the data to append.
nResumeFrom. Numeric. Optional. Specifies the offset in number of bytes that you want the transfer to start from. Set this option to 0 to make the transfer start from the beginning (effectively disabling resume). For FTP, set this option to -1 to make the transfer start from the end of the target file.

DESCRIPTION
Uploads a string to an FTP server file.
RETURN VALUE
nResultCode. Numeric.
REMARKS
When using the FileUploadString method, there is no need to specify the file protocol in the url (file://)

HttpDownloadFile(cRemoteFile, cLocalFile, [nResumeFrom], [cRange])
Downloads a remote file to a local file using the HTTP/HTTPS protocol.
REMARKS
See FtpDownloadFile

HttpDownloadString(cRemoteFile, [nResumeFrom], [cRange])
Downloads a remote file using the HTTP/HTTPS protocol, returns the contents as a character string.
REMARKS
See FtpDownloadString

HttpGetDateTimeStamp(cRemoteFile)
Attempts to retrieve the date-time stamp of a remote file using the HTTP/HTTPS protocol.
REMARKS
See FtpGetDateTimeStamp

HttpGetFileSize(cRemoteFile)
Attempts to get the size of a remote file using the HTTP/HTTPS protocol.
REMARKS
See FtpGetFileSize

HttpPostFile(cUrl, cLocalFile)
Posts a local file to a remote server using the HTTP/HTTPS protocol.
REMARKS
This is not tested, since I don´t have the need for it. If you need to use POST, contact me to explain your needs so I can finish this.

HttpPostString(cUrl, cString)
Posts a string to a remote server using the HTTP/HTTPS protocol.
REMARKS
This is not tested, since I don´t have the need for it. If you need to use POST, contact me to explain your needs so I can finish this.

HttpUploadFile(cRemoteFile, cLocalFile)
Uploads a local file to a remote file using the HTTP/HTTPS protocol PUT method.
REMARKS
See FtpUploadFile

HttpUploadString(cRemoteFile, cString)
Uploads a string to a remote file using the HTTP/HTTPS protocol PUT method.
REMARKS

See FtpUploadString

LogFileName
This property stores the name of the file where verbose information about libcurl operations will be stored. The DEFAULT VALUE: is "libcurl.log"
CurlOptVerbose must be TRUE to output debug info to this file. If empty, the DEFAULT VALUE is libcurl.log
VALUE TYPE
Character.
DEFAULT VALUE
None
READ/WRITE
Yes.
REMARKS
The log file is never deleted by the class, and debug information is always appended to it. Set CurlOptVerbose to TRUE to log debug information to this file.
Each line starts with a code that indicates the type of information of the line. Possible values are:

CURLINFO_TEXT	0
CURLINFO_HEADER_IN	1
CURLINFO_HEADER_OUT	2
CURLINFO_DATA_IN	3
CURLINFO_DATA_OUT	4
CURLINFO_SSL_DATA_IN	5
CURLINFO_SSL_DATA_OUT	6
CURLINFO_END	7

When the info type is CURLINFO_DATA_IN, CURLINFO_DATA_OUT, CURLINFO_SSL_DATA_IN or CURLINFO_SSL_DATA_OUT, the actual information is not stored in the line to prevent cluttering the debug buffer with useless data.
If the line contains the PASS ftp command, the password is masked.
You hardly ever want **CurlOptVerbose** set to TRUE in production use, you will almost always want this when you debug/report problems.

LogFileReset()
Deletes the log file.
PARAMETERS
None
DESCRIPTION
-
REMARKS
-

TransferProgress

This event gets fired about once every second when CurlOptProgress is TRUE and a transfer is in progress.

PARAMETERS

nBytesTotal, nBytesDone, nBytesLeft, nSpeed, nTimeTotal, nTimeDone, nTimeLeft

DESCRIPTION

-

REMARKS

Use this event to update the user interface with info about the current process.