CURL

TUTORIAL

A SmlCodes.Com Small presentation
In Association with Idleposts.com

For more tutorials & Articles visit SmlCodes.com

Copyright © 2016 SmlCodes.com All rights Reserved.
Copyright © 2016 Smlcodes.com

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, SmlCodes.com, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Smlcodes.com has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, SmlCodes.com Publishing cannot guarantee the accuracy of this information.

If you discover any errors on our website or in this tutorial, please notify us at support@smlcodes.com or smlcodes@gmail.com

First published on Mar 2017, Published by SmlCodes.com

Author Credits

Name : Satya Kaveti
Email : ceo@smlcodes.com
Website : smlcodes.com

Digital Partners
1. INTRODUCTION

2. CURL INSTALLATION & CONFIGURATION

3. CURL COMMANDS
   3.1 CURL –OPTIONS
   3.2 CURL –ENVIRONMENT VARIABLES
   3.3 EXIT CODES

4. CURL EXAMPLES
   4.1 SIMPLE USAGE
   4.2 USING PASSWORDS
   4.3 RANGES
   4.4 UPLOADING
   4.5 POST (HTTP)

5. CURL REAL-TIME OPERATIONS

REFERENCES
1. Introduction

Curl is a command line tool for doing all sorts of URL manipulations and transfers, but this particular document will focus on how to use it when doing HTTP requests for fun and profit. I'll assume that you know how to invoke 'curl --help' or 'curl --manual' to get basic information about it.

This tool is used for sending or receiving files from a server. The different protocols supported by curl are HTTP, HTTPS, FTP, TELNET, LDAP and FILE etc.,

`curl` offers proxy support, user authentication, FTP uploading, HTTP posting, SSL connections, cookies, file transfer resume, Metalink, and many other features

2. CURL Installation & Configuration

Curl is available in two formats.

1. **Source code formate(.zip)**
2. **Binary formate(.exe)**

You can download these from the [Official website](https://curl.haxx.se/download.html):

After downloading/Installing you can directly go to the curl folder where curl.exe exists through windows command prompt to run curl commands.you can find this in src folder. For example - curl -7.x.x/src/curl.exe
3. CURL Commands

Curl syntax

curl [options] [URL...]

3.1 CURL – Options

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>One or multiple URLs that will be fetched in sequence. Multiple URLs or parts of URLs can be specified by writing part sets within braces as in: <a href="http://site.%7Bone,two,three%7D.com">http://site.{one,two,three}.com</a> or get sequences of alphanumeric series by using [] as in: ftp://ftp.numericals.com/file[1-100].txt ftp://ftp.numericals.com/file[001-100].txt (with leading zeros) ftp://ftp.letters.com/file[a-z].txt</td>
</tr>
<tr>
<td>-A &quot;agent string&quot; --user-agent &quot;agent string&quot;</td>
<td>Specify the User-Agent string to send to the HTTP server. To encode blanks in the string, surround the string with single quote marks. This can also be set with -H, --header option. (HTTP)</td>
</tr>
<tr>
<td>-b name=data --cookie name=data</td>
<td>Send the data to the HTTP server as a cookie. It is supposedly the data previously received from the server in a &quot;Set-Cookie:&quot; line. The data should be in the format &quot;NAME1=VALUE1; NAME2=VALUE2&quot;.</td>
</tr>
<tr>
<td>-c filename --cookie-jar file name</td>
<td>Save cookies to file after a completed operation. Curl writes all cookies previously read from a specified file as well as all cookies received from remote server(s). If no cookies are known, no file will be written. To write to stdout, set the file name to a single dash, &quot;-&quot;.</td>
</tr>
<tr>
<td>--compressed</td>
<td>Request a compressed response using one of the algorithms curl supports (gzip), and save the uncompressed document. If this option is used and the server sends an unsupported encoding, curl will report an error. (HTTP)</td>
</tr>
<tr>
<td>-d @file -d &quot;string&quot; --data &quot;string&quot;</td>
<td>Send the specified data in an (HTTP) POST request, in the same way that a web browser does. This will pass the data using the content-type application/x-www-form-urlencoded. Compare to -F, --form.</td>
</tr>
<tr>
<td>-d, --data is the same as --data-ascii. To post data in pure binary, use --data-binary.</td>
<td>To URL-encode the value of a form field you may use --data-urlencode. Multiple date options will be merged together. Thus, using '-d name=daniel -d skill=lousy' would generate a post that looks like 'name=daniel&amp;skill=lousy'. If</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-F name=@file</td>
<td>Emulate a filled-in form in which a user has pressed the submit button. This will POST data using the Content-Type multipart/form-data according to RFC 2388. This enables uploading of binary files etc. If the data starts with @, the rest should be a filename containing the data.</td>
</tr>
<tr>
<td>-F name=content</td>
<td>--form name=content</td>
</tr>
<tr>
<td>-k</td>
<td>This option explicitly allows curl to perform &quot;insecure&quot; SSL connections and transfers. All SSL connections are attempted in secure mode using the CA certificate bundle installed by default. This makes all connections considered &quot;insecure&quot; fail unless -k, --insecure is used. (SSL).</td>
</tr>
<tr>
<td>--limit-rate</td>
<td>Specify the maximum transfer rate. This feature is useful if you have a limited pipe and you’d like your transfer not to use your entire bandwidth. The given speed is measured in bytes/second, unless a suffix is appended. Appending ‘k’ or ‘K’ will count the number as kilobytes/sec, ‘m’ or M’ megabytes, while ‘g’ or ‘G’ makes it gigabytes/sec. Eg: 200K, 3m, 1G.</td>
</tr>
<tr>
<td>speed</td>
<td>Specify the maximum transfer rate. This feature is useful if you have a limited pipe and you’d like your transfer not to use your entire bandwidth. The given speed is measured in bytes/second, unless a suffix is appended. Appending ‘k’ or ‘K’ will count the number as kilobytes/sec, ‘m’ or M’ megabytes, while ‘g’ or ‘G’ makes it gigabytes/sec. Eg: 200K, 3m, 1G.</td>
</tr>
<tr>
<td>-m seconds</td>
<td>Write output to a local file named like the remote file we get. (Only the file part of the remote file is used, the path is cut off.) The remote file name to use for saving is extracted from the given URL, nothing else. Consequently, the file will be saved in the current working directory.</td>
</tr>
<tr>
<td>--max-time</td>
<td>Write output to a local file named like the remote file we get. (Only the file part of the remote file is used, the path is cut off.) The remote file name to use for saving is extracted from the given URL, nothing else. Consequently, the file will be saved in the current working directory.</td>
</tr>
<tr>
<td>seconds</td>
<td>Write output to file instead of stdout. If you are using {} or [] to fetch multiple documents, you can use ‘#’ followed by a number in the file specifier. That variable will be replaced with the current string for the URL being fetched. Like in: curl http://{one,two}.site.com -o “file_#1.txt” or use several variables like: curl http://[site,host].host[1-5].com -o “#1_#2” You may use this option as many times as the number of URLs you have. See also --create-dirs option to create the local directories dynamically. Specify ‘-’ to force the output to stdout.</td>
</tr>
<tr>
<td>-O</td>
<td>Write output to a local file named like the remote file we get. (Only the file part of the remote file is used, the path is cut off.) The remote file name to use for saving is extracted from the given URL, nothing else. Consequently, the file will be saved in the current working directory.</td>
</tr>
<tr>
<td>--remote-name</td>
<td>Enable a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file.</td>
</tr>
<tr>
<td>-s</td>
<td>Silent or quiet mode. Don’t show progress meter or error messages.</td>
</tr>
<tr>
<td>--silent</td>
<td>Silent or quiet mode. Don’t show progress meter or error messages.</td>
</tr>
<tr>
<td>--trace-ascii</td>
<td>Enable a full trace dump of all incoming and outgoing data, including descriptive information, to the given output file.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>-T file</code></td>
<td>Use &quot;-&quot; as filename to have the output sent to stdout. This option overrides previous uses of <code>-v</code>, --verbose or --trace-ascii. If this option is used several times, the last one will be used.</td>
</tr>
<tr>
<td><code>--upload-file file</code></td>
<td>Transfer the specified local file to the remote URL. PUT If there is no file part in the specified URL, Curl will append the local file name. You must use a trailing / on the last directory to really prove to Curl that there is no file name or curl will think that the last directory name is the remote file name to use. Use the file name &quot;-&quot; to use stdin. You can specify one <code>-T</code> for each URL on the command line. Each <code>-T</code> + URL pair specifies what to upload and to where. curl also supports &quot;globbing&quot; of the <code>-T</code> argument, meaning that you can upload multiple files to a single URL like this: curl -T &quot;[file1,file2]&quot; <a href="http://www.uploadtothissite.com">http://www.uploadtothissite.com</a> or even curl -T &quot;img[1-1000].png&quot; ftp://ftp.picturemania.com/upload/</td>
</tr>
<tr>
<td><code>-I</code></td>
<td>Fetch the HTTP-header only! (HTTP/FTP/FILE) HTTP-servers feature the command HEAD which this uses to get nothing but the header of a document. When used on an FTP or FILE file, curl displays the file size and last modification time only.</td>
</tr>
<tr>
<td><code>--user user:password</code></td>
<td>The username and password to use for server authentication. Overrides -n, --netrc and --netrc-optional. If you just give the user name (without entering a colon) curl will prompt for a password. If you use an SSPI-enabled curl binary and do NTLM authentication, you can force curl to pick up the username and password from your environment by specifying a single colon with this option: &quot;-u: &quot;. If this option is used several times, the last one will be used.</td>
</tr>
<tr>
<td><code>-w</code></td>
<td>Define extra info to display on stdout after a completed and successful operation. The format is a string that may contain plain text mixed with any number of variables. The format string can be specified as &quot;string&quot;, or to read from a file specify &quot;@[filename]&quot; to read the format from stdin use &quot;@-&quot;. Various variables may be included in the format and will be substituted by curl (file size, ip address etc see man curl for details). variables are specified as %{variable_name} Output a newline using \n, a carriage return with \r and a tab space with \t.</td>
</tr>
<tr>
<td><code>-x host:port</code></td>
<td>Use the specified HTTP proxy. If the port number is not specified, it is assumed at port 1080.</td>
</tr>
</tbody>
</table>
3.2 CURL – Environment Variables

The environment variables can be specified in lower case or upper case. The lower case version has precedence. `http_proxy` is an exception as it is only available in lower case.

Using an environment variable to set the proxy has the same effect as using the `--proxy` option.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http_proxy</code> [protocol://] &lt;host&gt;[;port]</td>
<td>Sets the proxy server to use for HTTP.</td>
</tr>
<tr>
<td><code>HTTPS_PROXY</code> [protocol://] &lt;host&gt;[;port]</td>
<td>Sets the proxy server to use for HTTPS.</td>
</tr>
<tr>
<td><code>[url-protocol]_PROXY</code> [protocol://] &lt;host&gt;[;port]</td>
<td>Sets the proxy server to use for <code>[url-protocol]</code>, where the protocol is a protocol that curl supports and as specified in a URL: FTP, FTPS, POP3, IMAP, SMTP, LDAP etc.</td>
</tr>
<tr>
<td><code>ALL_PROXY</code> [protocol://] &lt;host&gt;[;port]</td>
<td>Sets the proxy server to use if no protocol-specific proxy is set.</td>
</tr>
<tr>
<td><code>NO_PROXY</code> &lt;comma-separated list of hosts&gt;</td>
<td>List of host names that shouldn’t go through any proxy. If set to a asterisk * only, it matches all hosts.</td>
</tr>
</tbody>
</table>

3.3 Exit Codes

There are a bunch of different error codes and their corresponding error messages that may appear during bad conditions. We have 90+ exit codes, listed from 0,1,2,...88,89,90,91..

We can find these all these Exit code from Official website:
https://curl.haxx.se/libcurl/c/libcurl-errors.html
4. CURL Examples

4.1 Simple Usage

1. Get the main page from any Website/WebServer

```
curl www.google.com
```

2. Get a directory listing of an FTP site:

```
curl ftp://servername/
```

3. Fetch two documents at once:

```
```
4.2 Using Passwords

Curl supports user and password in HTTP URLs, thus you can pick a file like:

```
curl http://name:passwd@machine.domain/full/path/to/file
```

or specify user and password separately like in

```
curl -u name:passwd http://machine.domain/full/path/to/file
```

4.3 Ranges

HTTP 1.1 introduced byte-ranges. Using this, a client can request to get only one or more subparts of a specified document. Curl supports this with the `-r` flag.

**Get the first 100 bytes of a document:**

```
curl -r 0-99 http://localhost:9090
```

**Get the last 500 bytes of a document:**

```
curl -r -500 http://localhost:9090
```
## 4.4 Uploading

Upload all data on stdin to a specified server

```
```

Upload data from a specified file, login with user and password

```
```

## 4.5 POST (HTTP)

It's easy to post data using curl. This is done using the `-d <data>` option. The post data must be urlencoded.

**curl post request with no data:**

```
curl -X POST http://URL/example.php
```

**curl post request with data:**

```
curl -d "data=example1&data2=example2" http://URL/example.cgi
```

**curl POST to a form:**

```
curl -X POST -F "name=user" -F "password=test" http://URL/example.php
```

**curl POST with a file:**

```
curl -X POST -F "image=@/path/example.gif" http://URL/uploadform.cgi
```
5. CURL Real-Time Operations

1. How to Download a Single File using CURL

The following command will get the content of the URL and display it in the STDOUT.

```bash
curl http://localhost:9090/
```

To store the output in a file, you can redirect it as shown below. This will also display some additional download statistics.

```bash
curl http://localhost:9090/ > stats.html
```

2. How to save the CURL Output to a file

We can save the result of the curl command to a file by using `-o/-O` options.

- `-o` (lowercase o) the result will be saved in the filename provided in the command line
- `-O` (uppercase O) the filename in the URL will be taken and it will be used as the filename to store the result

```bash
curl -o outputfile.html http://localhost:9090/
```

Now the output data will be stored in `outputfile.html`
3. How to Fetch Data from Multiple Files at a time
We can download multiple files in a single shot by specifying the URLs on the command line.
```
curl -O URL1 -O URL2
```

```
```

4. How to Get More Information using Verbose and Trace Option
You can get to know what is happening using the `-v` option. `-v` option enable the verbose mode and it will print the details
```
curl -v http://hostname/
```

```
C:\\DevOps\\curl\\src\\curl \-v http://localhost:8080/
 * timeout on name lookup is not supported
 * Trying ::1...
 * TCP_NODELAY set
 * Connected to localhost (::1) port 8080 (\W)
 > GET / HTTP/1.1
 > Host: localhost:8080
 > User-Agent: curl/7.53.1
 > Accept: */*
 >
 < HTTP/1.1 403 Forbidden
 < Date: Fri, 17 Mar 2017 12:20:02 GMT
 < X-Content-Type-Options: nosniff
 < Set-Cookie: JSESSIONID.34deab8b=14406sagr4ldw8rn8nngta38fg;
 < Expires: Thu, 01 Jan 1970 00:00:00 GMT
 < Content-Type: text/html;charset=UTF-8
 < X-Hudson: 1.395
 < X-Jenkins: 2.50
 < X-Jenkins-Session: a6a9dcf9
 < X-Hudson-CLI-Port: 56296
 < X-Jenkins CLI-Port: 56296
 < X-Jenkins: 1.56
 <
```

5. How to use Proxy to Download a File
We can specify cURL to use proxy to do the specific operation using `-x` option. We need to specify the host and port of the proxy.
```
$ curl -x proxyserver.test.com:3128 http://google.co.in
```

6. How to Send a Mail using SMTP Protocol
CURL can also be used to send mail using the SMTP protocol. You should specify the from-address, to-address, and the mailserver ip-address as shown below.
```
$ curl --mail-from ceo@smlcodes.com --mail-rcpt smlcodes@gmail.com smtp://mailserver.com
```
7. Download Files and Resume Downloads

I want to download a sample PDF file whose URL is http://www.pdf995.com/samples/pdf.pdf

To Download PDF file with Default name(CAPITAL ‘O’ – default name)

```
```

To Download PDF file with Custom name(small ‘o’ – Custom name)

```
```

To download multiple files at once, i.e. one after the other:

```
curl -O file1.txt -O file2.pdf -O file3.zip
```

8. How to Check If a Website Is Down or Not

To check website down or not by using below command

```
curl -Is https://www.domain.com -L | grep HTTP/
```

9. How to Get HTTP header information from a website

You can easily get HTTP header information from any website you want by adding the -I option (capital ‘i’) to cURL.

```
curl -I http://domain.com
```

```
C:\DevOps\curl\src>curl -I http://google.com
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.co.in/?gfe_rd=cr&ei=C9vLUMbj7HV8uf4u4LoCg
Content-Length: 261
Date: Fri, 17 Mar 2017 12:48:11 GMT
Proxy-Connection: Keep-Alive
Connection: Keep-Alive
```

10. How to Test URL by injecting header

You can use curl by injecting header with your data to test or troubleshoot the specific issue. Let’s see following example to request with Content-Type

```
curl --header 'Content-Type: application/json' http://domain.com
```
11. How to Connect HTTPS/SSL URL and ignore any SSL certificate error

If you are trying to access https URL and that is having certificate error due to host name mismatch or others, you can ignore them by using the following syntax.

```
curl --insecure https://domain.com
```

References

http://www.computerhope.com/unix/curl.htm

https://curl.haxx.se/docs/manual.html