NAME

CURLOPT_HTTPAUTH − HTTP server authentication methods to try

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_HTTPAUTH, long bitmask);

DESCRIPTION

Pass a long as parameter, which is set to a bitmask, to tell libcurl which authentication method(s) you want it to use speaking to the remote server.

The available bits are listed below. If more than one bit is set, libcurl first queries the host to see which authentication methods it supports and then picks the best one you allow it to use. For some methods, this induces an extra network round−trip. Set the actual name and password with the *CURLOPT_USERPWD(3)* option or with the *CURLOPT_USERNAME(3)* and the *CURLOPT_PASSWORD(3)* options.

For authentication with a proxy, see *CURLOPT_PROXYAUTH(3)*.

CURLAUTH_BASIC
: HTTP Basic authentication. This is the default choice, and the only method that is in wide−spread use and supported virtually everywhere. This sends the username and password over the network in plain text, easily captured by others.

CURLAUTH_DIGEST
: HTTP Digest authentication. Digest authentication is defined in RFC 2617 and is a more secure way to do authentication over public networks than the regular old−fashioned Basic method.

CURLAUTH_DIGEST_IE
: HTTP Digest authentication with an IE flavor. Digest authentication is defined in RFC 2617 and is a more secure way to do authentication over public networks than the regular old−fashioned Basic method. The IE flavor means that libcurl uses a special "quirk" that IE is known to have used before version 7 and that some servers require the client to use.

CURLAUTH_BEARER
: HTTP Bearer token authentication, used primarily in OAuth 2.0 protocol.

  You can set the Bearer token to use with *CURLOPT_XOAUTH2_BEARER(3)*.

CURLAUTH_NEGOTIATE
: HTTP Negotiate (SPNEGO) authentication. Negotiate authentication is defined in RFC 4559 and is the most secure way to perform authentication over HTTP.

  You need to build libcurl with a suitable GSS−API library or SSPI on Windows for this to work.

CURLAUTH_NTLM
: 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.

  NTLM uses weak cryptographic algorithms and is not considered secure.

CURLAUTH_NTLM_WB
: Support for this is removed since libcurl 8.8.0.

  NTLM delegating to winbind helper. Authentication is performed by a separate binary application that is executed when needed. The name of the application is specified at compile time but is typically **/usr/bin/ntlm_auth**.

  Note that libcurl forks when necessary to run the winbind application and kill it when complete,

calling **waitpid()** to await its exit when done. On POSIX operating systems, killing the process causes a SIGCHLD signal to be raised (regardless of whether *CURLOPT_NOSIGNAL(3)* is set), which must be handled intelligently by the application. In particular, the application must not unconditionally call wait() in its SIGCHLD signal handler to avoid being subject to a race condition. This behavior is subject to change in future versions of libcurl.

CURLAUTH_ANY
: This is a convenience macro that sets all bits and thus makes libcurl pick any it finds suitable. libcurl automatically selects the one it finds most secure.

CURLAUTH_ANYSAFE
: This is a convenience macro that sets all bits except Basic and thus makes libcurl pick any it finds suitable. libcurl automatically selects the one it finds most secure.

CURLAUTH_ONLY
: This is a meta symbol. OR this value together with a single specific auth value to force libcurl to probe for unrestricted auth and if not, only that single auth algorithm is acceptable.

CURLAUTH_AWS_SIGV4
: provides AWS V4 signature authentication on HTTPS header see *CURLOPT_AWS_SIGV4(3)*.

DEFAULT

CURLAUTH_BASIC

PROTOCOLS

This functionality affects http only

EXAMPLE

```
int main(void)
{
  CURL *curl = curl_easy_init();
  if(curl) {
    CURLcode result;
    curl_easy_setopt(curl, CURLOPT_URL, "https://example.com/");
    /* allow whatever auth the server speaks */
    curl_easy_setopt(curl, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
    curl_easy_setopt(curl, CURLOPT_USERPWD, "james:bond");
    result = curl_easy_perform(curl);
  }
}
```

HISTORY

CURLAUTH_DIGEST_IE was added in 7.19.3

CURLAUTH_ONLY was added in 7.21.3

CURLAUTH_NTLM_WB was added in 7.22.0

CURLAUTH_* macros became *long* types in 7.26.0, prior to this version a *long* cast was necessary when passed to *curl_easy_setopt(3)*.

CURLAUTH_BEARER was added in 7.61.0

CURLAUTH_AWS_SIGV4 was added in 7.74.0

AVAILABILITY

Added in curl 7.10.6

RETURN VALUE

curl_easy_setopt(3) returns a CURLcode indicating success or error.

CURLE_OK (0) means everything was OK, non−zero means an error occurred, see *libcurl−errors(3)*.

SEE ALSO

CURLOPT_PASSWORD(3), **CURLOPT_PROXYAUTH**(3), **CURLOPT_USERNAME**(3)