NAME

CURLOPT_SSLVERSION − preferred TLS/SSL version

SYNOPSIS

#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_SSLVERSION, long version);

DESCRIPTION

Pass a long as parameter to control which version range of SSL/TLS versions to use.

The SSL and TLS versions have typically developed from the most insecure version to be more and more secure in this order through history: SSL v2, SSLv3, TLS v1.0, TLS v1.1, TLS v1.2 and the most recent TLS v1.3.

Use one of the available defines for this purpose. The available options are:

CURL_SSLVERSION_DEFAULT
: The default acceptable version range. The minimum acceptable version is by default TLS v1.2 since 8.16.0 (unless the TLS library has a stricter rule).

CURL_SSLVERSION_TLSv1
: TLS v1.0 or later

CURL_SSLVERSION_SSLv2
: SSL v2 − refused

CURL_SSLVERSION_SSLv3
: SSL v3 − refused

CURL_SSLVERSION_TLSv1_0
: TLS v1.0 or later

CURL_SSLVERSION_TLSv1_1
: TLS v1.1 or later

CURL_SSLVERSION_TLSv1_2
: TLS v1.2 or later

CURL_SSLVERSION_TLSv1_3
: TLS v1.3 or later

The maximum TLS version can be set by using *one* of the CURL_SSLVERSION_MAX_ macros below. It is also possible to OR *one* of the CURL_SSLVERSION_ macros with *one* of the CURL_SSLVERSION_MAX_ macros.

CURL_SSLVERSION_MAX_DEFAULT
: The flag defines the maximum supported TLS version by libcurl, or the default value from the SSL library is used. libcurl uses a sensible default maximum, which was TLS v1.2 up to before 7.61.0 and is TLS v1.3 since then − assuming the TLS library support it.

CURL_SSLVERSION_MAX_TLSv1_0
: The flag defines maximum supported TLS version as TLS v1.0.

CURL_SSLVERSION_MAX_TLSv1_1
: The flag defines maximum supported TLS version as TLS v1.1.

CURL_SSLVERSION_MAX_TLSv1_2
: The flag defines maximum supported TLS version as TLS v1.2.

CURL_SSLVERSION_MAX_TLSv1_3
: The flag defines maximum supported TLS version as TLS v1.3.

DEFAULT

CURL_SSLVERSION_DEFAULT

PROTOCOLS

This functionality affects all TLS based protocols: HTTPS, FTPS, IMAPS, POP3S, SMTPS etc.

All TLS backends support this option.

EXAMPLE

```
int main(void)
{
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLcode result;
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");

  /* ask libcurl to use TLS version 1.0 or later */
  curl_easy_setopt(curl, CURLOPT_SSLVERSION, CURL_SSLVERSION_TLSv1);

  /* Perform the request */
  result = curl_easy_perform(curl);
  curl_easy_cleanup(curl);
 }
}
```

HISTORY

SSLv2 is disabled by default since 7.18.1. Other SSL versions availability may vary depending on which backend libcurl has been built to use.

SSLv3 is disabled by default since 7.39.0.

SSLv2 and SSLv3 are refused completely since curl 7.77.0

Since 8.10.0 wolfSSL is fully supported. Before 8.10.0 the MAX macros were not supported with wolfSSL and the other macros did not set a minimum, but restricted the TLS version to only the specified one.

Rustls support added in 8.10.0.

CURL_SSLVERSION_* macros became *long* types in 8.16.0, prior to this version a *long* cast was necessary when passed to *curl_easy_setopt(3)*.

AVAILABILITY

Added in curl 7.1

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_HTTP_VERSION(3), **CURLOPT_IPRESOLVE**(3), **CURLOPT_PROXY_SSLVERSION**(3), **CURLOPT_USE_SSL**(3)