NAME

CURLOPT_PROXY_SSL_VERIFYPEER − verify the proxy's SSL certificate

SYNOPSIS

```
#include <curl/curl.h>

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_PROXY_SSL_VERIFYPEER,
                          long verify);
```

DESCRIPTION

Pass a long as parameter set to 1L to enable or 0L to disable.

This option tells curl to verify the authenticity of the HTTPS proxy's certificate. A value of 1 means curl verifies; 0 (zero) means it does not.

This is the proxy version of *CURLOPT_SSL_VERIFYPEER(3)* that is used for ordinary HTTPS servers.

When negotiating a TLS or 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. curl uses a default bundle of CA certificates (the path for that is determined at build time) and you can specify alternate certificates with the *CURLOPT_PROXY_CAINFO(3)* option or the *CURLOPT_PROXY_CAPATH(3)* option.

When *CURLOPT_PROXY_SSL_VERIFYPEER(3)* is enabled, and the verification fails to prove that the certificate is authentic, the connection fails. When the option is zero, the peer certificate verification succeeds regardless.

Authenticating the certificate is not enough to be sure about the server. You typically also want to ensure that the server is the server you mean to be talking to. Use *CURLOPT_PROXY_SSL_VERIFYHOST(3)* for that. The check that the hostname in the certificate is valid for the hostname you are connecting to is done independently of the *CURLOPT_PROXY_SSL_VERIFYPEER(3)* option.

WARNING: disabling verification of the certificate allows bad guys to man−in−the−middle the communication without you knowing it. Disabling verification makes the communication insecure. Having encryption on a transfer is not enough as you cannot be sure that you are communicating with the correct end−point.

DEFAULT

1

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");

  /* Set the default value: strict certificate check please */
  curl_easy_setopt(curl, CURLOPT_PROXY_SSL_VERIFYPEER, 1L);
```

```
    result = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
  }
}
```

AVAILABILITY

Added in curl 7.52.0

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_PROXY_SSL_VERIFYHOST(3), **CURLOPT_SSL_VERIFYHOST**(3), **CURLOPT_SSL_VERIFYPEER**(3)