NAME

CURLINFO_TLS_SSL_PTR − TLS session info

SYNOPSIS

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

CURLcode curl_easy_getinfo(CURL *handle, CURLINFO_TLS_SSL_PTR,
                           struct curl_tlssessioninfo **session);
```

DESCRIPTION

Pass a pointer to a *struct curl_tlssessioninfo* *. The pointer is initialized to refer to a *struct curl_tlssessioninfo* * that contains an enum indicating the SSL library used for the handshake and a pointer to the respective internal TLS session structure of this underlying SSL library.

This option may be useful for example to extract certificate information in a format convenient for further processing, such as manual validation. Refer to the **LIMITATIONS** section.

```
struct curl_tlssessioninfo {
  curl_sslbackend backend;
  void *internals;
};
```

The *backend* struct member is one of these defines: CURLSSLBACKEND_NONE (when built without TLS support), CURLSSLBACKEND_WOLFSSL, CURLSSLBACKEND_SECURETRANSPORT, CURLSSLBACKEND_GNUTLS, CURLSSLBACKEND_MBEDTLS, CURLSSLBACKEND_NSS, CURLSSLBACKEND_OPENSSL or CURLSSLBACKEND_SCHANNEL. (Note that the OpenSSL forks are all reported as OpenSSL here.)

The *internals* struct member points to a TLS library specific pointer for the active ("in use") SSL connection, with the following underlying types:

GnuTLS
: **gnutls_session_t**

OpenSSL
: *CURLINFO_TLS_SESSION(3)*: **SSL_CTX** *

: *CURLINFO_TLS_SSL_PTR(3)*: **SSL** *

mbedTLS
: **mbedTLS_ssl_context** *

Secure Channel
: **CtxtHandle** *

wolfSSL
: **SSL** *

If the *internals* pointer is NULL then either the SSL backend is not supported, an SSL session has not yet been established or the connection is no longer associated with the easy handle (e.g. *curl_easy_perform(3)* has returned).

LIMITATIONS

This option has some limitations that could make it unsafe when it comes to the manual verification of certificates.

This option only retrieves the first in−use SSL session pointer for your easy handle, however your easy handle may have more than one in−use SSL session if using FTP over SSL. That is because the FTP protocol has a control channel and a data channel and one or both may be over SSL. Currently there is no way to retrieve a second in−use SSL session associated with an easy handle.

This option has not been thoroughly tested with clear text protocols that can be upgraded/downgraded to/from SSL: FTP, SMTP, POP3, IMAP when used with *CURLOPT_USE_SSL(3)*. Though you can to retrieve the SSL pointer, it is possible that before you can do that, data (including auth) may have already been sent over a connection after it was upgraded.

Renegotiation. If unsafe renegotiation or renegotiation in a way that the certificate is allowed to change is allowed by your SSL library this may occur and the certificate may change, and data may continue to be sent or received after renegotiation but before you are able to get the (possibly) changed SSL pointer, with the (possibly) changed certificate information.

Instead of using this option to poll for certificate changes use *CURLOPT_SSL_CTX_FUNCTION(3)* to set a verification callback, if supported. That is safer and does not suffer from any of the problems above.

How are you using this option? Are you affected by any of these limitations? Please let us know by making a comment at https://github.com/curl/curl/issues/685

PROTOCOLS

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

This option works only with the following TLS backends: GnuTLS, OpenSSL, Schannel, mbedTLS and wolfSSL

EXAMPLE

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

CURL *curl;
static size_t wf(char *ptr, size_t size, size_t nmemb, void *stream)
{
 const struct curl_tlssessioninfo *info = NULL;
 CURLcode result = curl_easy_getinfo(curl, CURLINFO_TLS_SSL_PTR, &info);
 if(info && !result) {
  if(CURLSSLBACKEND_OPENSSL == info->backend) {
   printf("OpenSSL ver. %s\n", SSL_get_version((SSL*)info->internals));
  }
 }
 return size * nmemb;
}

int main(int argc, char **argv)
{
 CURLcode result;
 curl = curl_easy_init();
 if(curl) {
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");
  curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, wf);
  result = curl_easy_perform(curl);
  curl_easy_cleanup(curl);
 }
 return result;
}
```

HISTORY

This option supersedes *CURLINFO_TLS_SESSION(3)* which was added in 7.34.0. This option is exactly the same as that option except in the case of OpenSSL.

Non−OpenSSL support was added in 7.48.0.

AVAILABILITY

Added in curl 7.48.0

RETURN VALUE

curl_easy_getinfo(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

CURLINFO_TLS_SESSION(3), **curl_easy_getinfo**(3), **curl_easy_setopt**(3)