NAME

CURLOPT_SSL_CTX_FUNCTION − SSL context callback for OpenSSL or wolfSSL/CyaSSL

SYNOPSIS

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

CURLcode ssl_ctx_callback(CURL *curl, void *ssl_ctx, void *userptr);

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_SSL_CTX_FUNCTION,
                          ssl_ctx_callback);
```

DESCRIPTION

This option only works for libcurl powered by OpenSSL or wolfSSL/CyaSSL. If libcurl was built against another SSL library this functionality is absent.

Pass a pointer to your callback function, which should match the prototype shown above.

This callback function gets called by libcurl just before the initialization of an SSL connection after having processed all other SSL related options to give a last chance to an application to modify the behaviour of the SSL initialization. The *ssl_ctx* parameter is actually a pointer to the SSL library's *SSL_CTX*. If an error is returned from the callback no attempt to establish a connection is made and the perform operation will return the callback's error code. Set the *userptr* argument with the *CURLOPT_SSL_CTX_DATA(3)* option.

This function will get called on all new connections made to a server, during the SSL negotiation. The SSL_CTX pointer will be a new one every time.

To use this properly, a non-trivial amount of knowledge of your SSL library is necessary. For example, you can use this function to call library-specific callbacks to add additional validation code for certificates, and even to change the actual URI of a HTTPS request.

DEFAULT

NULL

PROTOCOLS

All TLS based protocols: HTTPS, FTPS, IMAPS, POP3S, SMTPS etc.

EXAMPLE

```
/* OpenSSL specific */

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

static CURLcode sslctx_function(CURL *curl, void *sslctx, void *parm)
{
  X509_STORE *store;
  X509 *cert=NULL;
  BIO *bio;
  char *mypem = /* example CA cert PEM - shortened */
    "-----BEGIN CERTIFICATE-----0
    "MIIHPTCCBSWgAwIBAgIBADANBgkqhkiG9w0BAQQFADB5MRAwDgYDVQQKEwdSb2900
    "IENBMR4wHAYDVQQLExVodHRwOi8vd3d3LmNhY2VydC5vcmcxIjAgBgNVBAMTGUNB0
    "IENlcnQgU2lnbmluZyBBdXRob3JpdHkxITAfBgkqhkiG9w0BCQEWEnN1cHBvcnRA0
    "Y2FjZXJ0Lm9yZzAeFw0wMzAzMzAxMjI5NDlaFw0zMzAzMjkxMjI5NDlaMHkxEDAO0
    "GCSNe9FINSkYQKyTYOGWhlC0elnYjyELn8+CkcY7v2vcB5G5l1YjqrZslMZIBjzk0
    "zk6q5PYvCdxTby78dOs6Y5nCpqyJvKeyRKANihDjbPIky/qbn3BHLt4Ui9SyIAmW0
    "omTxJBzcoTWcFbLUvFUufQb1nA5V9FrWk9p2rSVzTMVD0    "-----END CERTIFICATE-----0;
  /* get a BIO */
  bio=BIO_new_mem_buf(mypem, -1);
  /* use it to read the PEM formatted certificate from memory into an X509
   * structure that SSL can use
   */
  PEM_read_bio_X509(bio, &cert, 0, NULL);
  if(cert == NULL)
    printf("PEM_read_bio_X509 failed...0);

  /* get a pointer to the X509 certificate store (which may be empty!) */
  store=SSL_CTX_get_cert_store((SSL_CTX *)sslctx);

  /* add our certificate to this store */
  if(X509_STORE_add_cert(store, cert)==0)
    printf("error adding certificate0);

  /* decrease reference counts */
  X509_free(cert);
  BIO_free(bio);

  /* all set to go */
  return CURLE_OK;
}

int main(void)
{
  CURL * ch;
  CURLcode rv;

  rv=curl_global_init(CURL_GLOBAL_ALL);
  ch=curl_easy_init();
  rv=curl_easy_setopt(ch, CURLOPT_SSLCERTTYPE, "PEM");
  rv=curl_easy_setopt(ch, CURLOPT_SSL_VERIFYPEER, 1L);
  rv=curl_easy_setopt(ch, CURLOPT_URL, "https://www.example.com/");

  /* Retrieve page using cacerts' certificate -> will succeed
   * load the certificate by installing a function doing the nescessary
   * "modifications" to the SSL CONTEXT just before link init
   */
  rv=curl_easy_setopt(ch, CURLOPT_SSL_CTX_FUNCTION, *sslctx_function);
  rv=curl_easy_perform(ch);
  if(rv==CURLE_OK)
    printf("*** transfer succeeded ***0);
  else
    printf("*** transfer failed ***0);

  curl_easy_cleanup(ch);
  curl_global_cleanup();
  return rv;
}
```

AVAILABILITY

Added in 7.11.0 for OpenSSL. Added in 7.42.0 for wolfSSL/CyaSSL. Other SSL backends not supported.

RETURN VALUE

Returns CURLE_OK if the option is supported, and CURLE_UNKNOWN_OPTION if not.

SEE ALSO

CURLOPT_SSL_CTX_DATA(3), **CURLOPT_SSL_VERIFYPEER**(3),