NAME

CURLOPT_PINNEDPUBLICKEY − pinned public key

SYNOPSIS

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

CURLcode curl_easy_setopt(CURL *handle, CURLOPT_PINNEDPUBLICKEY,
                          char *pinnedpubkey);
```

DESCRIPTION

Pass a pointer to a null−terminated string as parameter. The string can be the filename of your pinned public key. The file format expected is "PEM" or "DER". The string can also be any number of base64 encoded sha256 hashes preceded by "sha256//" and separated by ";"

When negotiating a TLS or SSL connection, the server sends a certificate indicating its identity. A public key is extracted from this certificate and if it does not exactly match the public key provided to this option, curl aborts the connection before sending or receiving any data.

This option is independent of option *CURLOPT_SSL_VERIFYPEER(3)*. If you turn off that option then the peer is still verified by public key.

On mismatch, *CURLE_SSL_PINNEDPUBKEYNOTMATCH* is returned.

The application does not have to keep the string around after setting this option.

DEFAULT

NULL

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

```
int main(void)
{
 CURL *curl = curl_easy_init();
 if(curl) {
  CURLcode result;
  curl_easy_setopt(curl, CURLOPT_URL, "https://example.com");
  curl_easy_setopt(curl, CURLOPT_PINNEDPUBLICKEY, "/etc/publickey.der");
  /* OR
  curl_easy_setopt(curl, CURLOPT_PINNEDPUBLICKEY,
                   "sha256//YhKJKSzoTt2b5FP18fvpHo7fJYqQCjAa3HWY3"
                   "tvRMwE=;sha256//t62CeU2tQiqkexU74Gxa2eg7fRbEg"
                   "oChTociMee9wno=");
  */

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

PUBLIC KEY EXTRACTION

If you do not have the server's public key file you can extract it from the server's certificate.

```
# retrieve the server’s certificate if you do not already have it
```

```
#
# be sure to examine the certificate to see if it is what you expected
#
# Windows-specific:
# - Use NUL instead of /dev/null.
# - OpenSSL may wait for input instead of disconnecting. Hit enter.
# - If you do not have sed, then copy the certificate into a file:
#   Lines from -----BEGIN CERTIFICATE----- to -----END CERTIFICATE-----.
#
openssl s_client -servername www.example.com -connect www.example.com:443 \
  < /dev/null | sed -n "/-----BEGIN/,/-----END/p" > www.example.com.pem

# extract public key in pem format from certificate
openssl x509 -in www.example.com.pem -pubkey -noout > www.example.com.pubkey.pem

# convert public key from pem to der
openssl asn1parse -noout -inform pem -in www.example.com.pubkey.pem \
  -out www.example.com.pubkey.der

# sha256 hash and base64 encode der to string for use
openssl dgst -sha256 -binary www.example.com.pubkey.der | openssl base64
```

The public key in PEM format contains a header, base64 data and a footer:

```
-----BEGIN PUBLIC KEY-----
[BASE 64 DATA]
-----END PUBLIC KEY-----
```

HISTORY

PEM/DER support

7.39.0: OpenSSL, GnuTLS

7.43.0: wolfSSL

7.47.0: mbedTLS

7.58.1: Schannel

sha256 support

7.44.0: OpenSSL, GnuTLS and wolfSSL

7.47.0: mbedTLS

7.58.1: Schannel

Other SSL backends not supported.

AVAILABILITY

Added in curl 7.39.0

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_CAINFO(3), **CURLOPT_CAPATH**(3), **CURLOPT_SSL_VERIFYHOST**(3), **CURLOPT_SSL_VERIFYPEER**(3)