

NAME

SSL_CTX_set_cert_cb, SSL_set_cert_cb - handle certificate callback function

SYNOPSIS

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

```
void SSL_CTX_set_cert_cb(SSL_CTX *c, int (*cert_cb)(SSL *ssl, void *arg), void *arg);  
void SSL_set_cert_cb(SSL *s, int (*cert_cb)(SSL *ssl, void *arg), void *arg);
```

```
int (*cert_cb)(SSL *ssl, void *arg);
```

DESCRIPTION

SSL_CTX_set_cert_cb() and *SSL_set_cert_cb()* sets the *cert_cb()* callback, **arg** value is pointer which is passed to the application callback.

When *cert_cb()* is NULL, no callback function is used.

cert_cb() is the application defined callback. It is called before a certificate will be used by a client or server. The callback can then inspect the passed **ssl** structure and set or clear any appropriate certificates. If the callback is successful it **MUST** return 1 even if no certificates have been set. A zero is returned on error which will abort the handshake with a fatal internal error alert. A negative return value will suspend the handshake and the handshake function will return immediately. [SSL_get_error\(3\)](#) will return SSL_ERROR_WANT_X509_LOOKUP to indicate, that the handshake was suspended. The next call to the handshake function will again lead to the call of *cert_cb()*. It is the job of the *cert_cb()* to store information about the state of the last call, if required to continue.

NOTES

An application will typically call *SSL_use_certificate()* and *SSL_use_PrivateKey()* to set the end entity certificate and private key. It can add intermediate and optionally the root CA certificates using *SSL_add1_chain_cert()*.

It might also call *SSL_certs_clear()* to delete any certificates associated with the **SSL** object.

The certificate callback functionality supersedes the (largely broken) functionality provided by the old client certificate callback interface. It is **always** called even if a certificate is already set so the callback can modify or delete the existing certificate.

A more advanced callback might examine the handshake parameters and set whatever chain is appropriate. For example a legacy client supporting only TLS v1.0 might receive a certificate chain signed using SHA1 whereas a TLS v1.2 client which advertises support for SHA256 could receive a chain using SHA256.

Normal server sanity checks are performed on any certificates set by the callback. So if an EC chain is set for a curve the client does not support it will **not** be used.

SEE ALSO

[ssl\(3\)](#), [SSL_use_certificate\(3\)](#), [SSL_add1_chain_cert\(3\)](#), [SSL_get_client_CA_list\(3\)](#), [SSL_clear\(3\)](#), [SSL_free\(3\)](#)

COPYRIGHT

Copyright 2014-2016 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the OpenSSL license (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.