NAME

SSL_CTX_set_client_cert_cb, SSL_CTX_get_client_cert_cb - handle client certificate callback function

SYNOPSIS

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

 void SSL_CTX_set_client_cert_cb(SSL_CTX *ctx,
                                 int (*client_cert_cb)(SSL *ssl, X509 **x509,
                                                       EVP_PKEY **pkey));
 int (*SSL_CTX_get_client_cert_cb(SSL_CTX *ctx))(SSL *ssl, X509 **x509,
                                                 EVP_PKEY **pkey);
```

DESCRIPTION

SSL_CTX_set_client_cert_cb() sets the *client_cert_cb* callback, that is called when a client certificate is requested by a server and no certificate was yet set for the SSL object.

When *client_cert_cb* is NULL, no callback function is used.

SSL_CTX_get_client_cert_cb() returns a pointer to the currently set callback function.

client_cert_cb is the application defined callback. If it wants to set a certificate, a certificate/private key combination must be set using the *x509* and *pkey* arguments and "1" must be returned. The certificate will be installed into *ssl*, see the NOTES and BUGS sections. If no certificate should be set, "0" has to be returned and no certificate will be sent. 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 *client_cert_cb*. It is the job of the *client_cert_cb* to store information about the state of the last call, if required to continue.

NOTES

During a handshake (or renegotiation) a server may request a certificate from the client. A client certificate must only be sent, when the server did send the request.

When a certificate was set using the **SSL_CTX_use_certificate**(3) family of functions, it will be sent to the server. The TLS standard requires that only a certificate is sent, if it matches the list of acceptable CAs sent by the server. This constraint is violated by the default behavior of the OpenSSL library. Using the callback function it is possible to implement a proper selection routine or to allow a user interaction to choose the certificate to be sent.

If a callback function is defined and no certificate was yet defined for the SSL object, the callback function will be called. If the callback function returns a certificate, the OpenSSL library will try to load the private key and certificate data into the SSL object using the **SSL_use_certificate()** and **SSL_use_private_key()** functions. Thus it will permanently install the certificate and key for this SSL object. It will not be reset by calling **SSL_clear**(3). If the callback returns no certificate, the OpenSSL library will not send a certificate.

RETURN VALUES

SSL_CTX_get_client_cert_cb() returns function pointer of *client_cert_cb* or NULL if the callback is not set.

BUGS

The *client_cert_cb* cannot return a complete certificate chain, it can only return one client certificate. If the chain only has a length of 2, the root CA certificate may be omitted according to the TLS standard and thus a standard conforming answer can be sent to the server. For a longer chain, the client must send the complete chain (with the option to leave out the root CA certificate). This can only be accomplished by either adding the intermediate CA certificates into the trusted certificate store for the SSL_CTX object (resulting in having to add CA certificates that otherwise maybe would not be trusted), or by adding the chain certificates using the **SSL_CTX_add_extra_chain_cert**(3) function, which is only available for the SSL_CTX object as a whole and that therefore probably can only apply for one client certificate, making the concept of the callback function (to allow the choice from several certificates) questionable.

Once the SSL object has been used in conjunction with the callback function, the certificate will be set for the SSL object and will not be cleared even when **SSL_clear**(3) is being called. It is therefore mandatory to destroy the SSL object using **SSL_free**(3) and create a new one to return to the previous state.

SEE ALSO

ssl(7), **SSL_CTX_use_certificate**(3), **SSL_CTX_add_extra_chain_cert**(3), **SSL_get_client_CA_list**(3), **SSL_clear**(3), **SSL_free**(3)

COPYRIGHT