NAME

TLSv1_2_method, TLSv1_2_server_method, TLSv1_2_client_method, SSL_CTX_new, SSL_CTX_up_ref, SSLv3_method, SSLv3_server_method, SSLv3_client_method, TLSv1_method, TLSv1_server_method, TLSv1_client_method, TLSv1_1_method, TLSv1_1_server_method, TLSv1_1_client_method, TLS_method, TLS_server_method, TLS_client_method, SSLv23_method, SSLv23_server_method, SSLv23_client_method, DTLS_method, DTLS_server_method, DTLS_client_method, DTLSv1_method, DTLSv1_server_method, DTLSv1_client_method, DTLSv1_2_method, DTLSv1_2_server_method, DTLSv1_2_client_method - create a new SSL_CTX object as framework for TLS/SSL or DTLS enabled functions

SYNOPSIS

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

SSL_CTX *SSL_CTX_new(const SSL_METHOD *method);
int SSL_CTX_up_ref(SSL_CTX *ctx);

const SSL_METHOD *TLS_method(void);
const SSL_METHOD *TLS_server_method(void);
const SSL_METHOD *TLS_client_method(void);

const SSL_METHOD *SSLv23_method(void);
const SSL_METHOD *SSLv23_server_method(void);
const SSL_METHOD *SSLv23_client_method(void);

#ifndef OPENSSL_NO_SSL3_METHOD
const SSL_METHOD *SSLv3_method(void);
const SSL_METHOD *SSLv3_server_method(void);
const SSL_METHOD *SSLv3_client_method(void);
#endif

#ifndef OPENSSL_NO_TLS1_METHOD
const SSL_METHOD *TLSv1_method(void);
const SSL_METHOD *TLSv1_server_method(void);
const SSL_METHOD *TLSv1_client_method(void);
#endif

#ifndef OPENSSL_NO_TLS1_1_METHOD
const SSL_METHOD *TLSv1_1_method(void);
const SSL_METHOD *TLSv1_1_server_method(void);
const SSL_METHOD *TLSv1_1_client_method(void);
#endif

#ifndef OPENSSL_NO_TLS1_2_METHOD
const SSL_METHOD *TLSv1_2_method(void);
const SSL_METHOD *TLSv1_2_server_method(void);
const SSL_METHOD *TLSv1_2_client_method(void);
#endif

const SSL_METHOD *DTLS_method(void);
const SSL_METHOD *DTLS_server_method(void);
const SSL_METHOD *DTLS_client_method(void);

#ifndef OPENSSL_NO_DTLS1_METHOD
const SSL_METHOD *DTLSv1_method(void);
const SSL_METHOD *DTLSv1_server_method(void);
```

```
const SSL_METHOD *DTLSv1_client_method(void);
#endif

#ifndef OPENSSL_NO_DTLS1_2_METHOD
const SSL_METHOD *DTLSv1_2_method(void);
const SSL_METHOD *DTLSv1_2_server_method(void);
const SSL_METHOD *DTLSv1_2_client_method(void);
#endif
```

DESCRIPTION

SSL_CTX_new() creates a new **SSL_CTX** object as framework to establish TLS/SSL or DTLS enabled connections. An **SSL_CTX** object is reference counted. Creating an **SSL_CTX** object for the first time increments the reference count. Freeing it (using SSL_CTX_free) decrements it. When the reference count drops to zero, any memory or resources allocated to the **SSL_CTX** object are freed. *SSL_CTX_up_ref()* increments the reference count for an existing **SSL_CTX** structure.

NOTES

The SSL_CTX object uses **method** as connection method. The methods exist in a generic type (for client and server use), a server only type, and a client only type. **method** can be of the following types:

TLS_method(), TLS_server_method(), TLS_client_method()
: These are the general-purpose *version-flexible* SSL/TLS methods. The actual protocol version used will be negotiated to the highest version mutually supported by the client and the server. The supported protocols are SSLv3, TLSv1, TLSv1.1 and TLSv1.2. Applications should use these methods, and avoid the version-specific methods described below.

SSLv23_method(), SSLv23_server_method(), SSLv23_client_method()
: Use of these functions is deprecated. They have been replaced with the above *TLS_method(), TLS_server_method()* and *TLS_client_method()* respectively. New code should use those functions instead.

TLSv1_2_method(), TLSv1_2_server_method(), TLSv1_2_client_method()
: A TLS/SSL connection established with these methods will only understand the TLSv1.2 protocol.

TLSv1_1_method(), TLSv1_1_server_method(), TLSv1_1_client_method()
: A TLS/SSL connection established with these methods will only understand the TLSv1.1 protocol.

TLSv1_method(), TLSv1_server_method(), TLSv1_client_method()
: A TLS/SSL connection established with these methods will only understand the TLSv1 protocol.

SSLv3_method(), SSLv3_server_method(), SSLv3_client_method()
: A TLS/SSL connection established with these methods will only understand the SSLv3 protocol. The SSLv3 protocol is deprecated and should not be used.

DTLS_method(), DTLS_server_method(), DTLS_client_method()
: These are the version-flexible DTLS methods. Currently supported protocols are DTLS 1.0 and DTLS 1.2.

DTLSv1_2_method(), DTLSv1_2_server_method(), DTLSv1_2_client_method()
: These are the version-specific methods for DTLSv1.2.

DTLSv1_method(), DTLSv1_server_method(), DTLSv1_client_method()
: These are the version-specific methods for DTLSv1.

SSL_CTX_new() initializes the list of ciphers, the session cache setting, the callbacks, the keys and certificates and the options to their default values.

TLS_method(), TLS_server_method(), TLS_client_method(), DTLS_method(), DTLS_server_method() and *DTLS_client_method()* are the *version-flexible* methods. All other methods only support one specific protocol version. Use the*ver sion-flexible* methods instead of the version specific methods.

If you want to limit the supported protocols for the version flexible methods you can use *SSL_CTX_set_min_proto_version(3)*, *SSL_set_min_proto_version(3)*,

SSL_CTX_set_max_proto_version(3) and *SSL_set_max_proto_version(3)* functions. Using these functions it is possible to choose e.g. *TLS_server_method()* and be able to negotiate with all possible clients, but to only allow newer protocols like TLS 1.0, TLS 1.1 or TLS 1.2.

The list of protocols available can also be limited using the **SSL_OP_NO_SSLv3**, **SSL_OP_NO_TLSv1**, **SSL_OP_NO_TLSv1_1** and **SSL_OP_NO_TLSv1_2** options of the *SSL_CTX_set_options(3)* or *SSL_set_options(3)* functions, but this approach is not recommended. Clients should avoid creating ''holes'' in the set of protocols they support. When disabling a protocol, make sure that you also disable either all previous or all subsequent protocol versions. In clients, when a protocol version is disabled without disabling *all* previous protocol versions, the effect is to also disable all subsequent protocol versions.

The SSLv3 protocol is deprecated and should generally not be used. Applications should typically use *SSL_CTX_set_min_proto_version(3)* to set the minimum protocol to at least **TLS1_VERSION**.

RETURN VALUES

The following return values can occur:

NULL
: The creation of a new SSL_CTX object failed. Check the error stack to find out the reason.

Pointer to an SSL_CTX object
: The return value points to an allocated SSL_CTX object.

SSL_CTX_up_ref() returns 1 for success and 0 for failure.

HISTORY

Support for SSLv2 and the corresponding *SSLv2_method()*, *SSLv2_server_method()* and *SSLv2_client_method()* functions where removed in OpenSSL 1.1.0.

SSLv23_method(), *SSLv23_server_method()* and *SSLv23_client_method()* were deprecated and the preferred *TLS_method()*, *TLS_server_method()* and *TLS_client_method()* functions were introduced in OpenSSL 1.1.0.

All version-specific methods were deprecated in OpenSSL 1.1.0.

SEE ALSO

SSL_CTX_set_options(3), *SSL_CTX_free(3)*, *SSL_accept(3)*, *SSL_CTX_set_min_proto_version(3)*, *ssl(3)*, *SSL_set_connect_state(3)*

COPYRIGHT