NAME

SSL_CTX_use_certificate, SSL_CTX_use_certificate_ASN1, SSL_CTX_use_certificate_file, SSL_use_certificate, SSL_use_certificate_ASN1, SSL_use_certificate_file, SSL_CTX_use_certificate_chain_file, SSL_use_certificate_chain_file, SSL_CTX_use_PrivateKey, SSL_CTX_use_PrivateKey_ASN1, SSL_CTX_use_PrivateKey_file, SSL_CTX_use_RSAPrivateKey, SSL_CTX_use_RSAPrivateKey_ASN1, SSL_CTX_use_RSAPrivateKey_file, SSL_use_PrivateKey_file, SSL_use_PrivateKey_ASN1, SSL_use_PrivateKey, SSL_use_RSAPrivateKey, SSL_use_RSAPrivateKey_ASN1, SSL_use_RSAPrivateKey_file, SSL_CTX_check_private_key, SSL_check_private_key, SSL_CTX_use_cert_and_key, SSL_use_cert_and_key - load certificate and key data

SYNOPSIS

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

int SSL_CTX_use_certificate(SSL_CTX *ctx, X509 *x);
int SSL_CTX_use_certificate_ASN1(SSL_CTX *ctx, int len, const unsigned char *d);
int SSL_CTX_use_certificate_file(SSL_CTX *ctx, const char *file, int type);
int SSL_use_certificate(SSL *ssl, X509 *x);
int SSL_use_certificate_ASN1(SSL *ssl, const unsigned char *d, int len);
int SSL_use_certificate_file(SSL *ssl, const char *file, int type);

int SSL_CTX_use_certificate_chain_file(SSL_CTX *ctx, const char *file);
int SSL_use_certificate_chain_file(SSL *ssl, const char *file);

int SSL_CTX_use_PrivateKey(SSL_CTX *ctx, EVP_PKEY *pkey);
int SSL_CTX_use_PrivateKey_ASN1(int pk, SSL_CTX *ctx, const unsigned char *d,
                                long len);
int SSL_CTX_use_PrivateKey_file(SSL_CTX *ctx, const char *file, int type);
int SSL_CTX_use_RSAPrivateKey(SSL_CTX *ctx, RSA *rsa);
int SSL_CTX_use_RSAPrivateKey_ASN1(SSL_CTX *ctx, const unsigned char *d, long len);
int SSL_CTX_use_RSAPrivateKey_file(SSL_CTX *ctx, const char *file, int type);
int SSL_use_PrivateKey(SSL *ssl, EVP_PKEY *pkey);
int SSL_use_PrivateKey_ASN1(int pk, SSL *ssl, const unsigned char *d, long len);
int SSL_use_PrivateKey_file(SSL *ssl, const char *file, int type);
int SSL_use_RSAPrivateKey(SSL *ssl, RSA *rsa);
int SSL_use_RSAPrivateKey_ASN1(SSL *ssl, const unsigned char *d, long len);
int SSL_use_RSAPrivateKey_file(SSL *ssl, const char *file, int type);

int SSL_CTX_check_private_key(const SSL_CTX *ctx);
int SSL_check_private_key(const SSL *ssl);
```

int SSL_CTX_use_cert_and_key(SSL_CTX *ctx, X509 *x, EVP_PKEY *pkey, STACK_OF(X509) *chain, int overri
int SSL_use_cert_and_key(SSL *ssl, X509 *x, EVP_PKEY *pkey, STACK_OF(X509) *chain, int override);

DESCRIPTION

These functions load the certificates and private keys into the SSL_CTX or SSL object, respectively.

The SSL_CTX_* class of functions loads the certificates and keys into the SSL_CTX object **ctx**. The information is passed to SSL objects **ssl** created from **ctx** with **SSL_new**(3) by copying, so that changes applied to **ctx** do not propagate to already existing SSL objects.

The SSL_* class of functions only loads certificates and keys into a specific SSL object. The specific information is kept, when **SSL_clear**(3) is called for this SSL object.

SSL_CTX_use_certificate() loads the certificate **x** into **ctx**, **SSL_use_certificate()** loads **x** into **ssl**. The rest of the certificates needed to form the complete certificate chain can be specified using the **SSL_CTX_add_extra_chain_cert**(3) function.

SSL_CTX_use_certificate_ASN1() loads the ASN1 encoded certificate from the memory location **d** (with length **len**) into **ctx**, **SSL_use_certificate_ASN1()** loads the ASN1 encoded certificate into **ssl**.

SSL_CTX_use_certificate_file() loads the first certificate stored in **file** into **ctx**. The formatting **type** of the certificate must be specified from the known types SSL_FILETYPE_PEM, SSL_FILETYPE_ASN1. **SSL_use_certificate_file()** loads the certificate from **file** into **ssl**. See the NOTES section on why **SSL_CTX_use_certificate_chain_file()** should be preferred.

SSL_CTX_use_certificate_chain_file() loads a certificate chain from **file** into **ctx**. The certificates must be in PEM format and must be sorted starting with the subject's certificate (actual client or server certificate), followed by intermediate CA certificates if applicable, and ending at the highest level (root) CA. **SSL_use_certificate_chain_file()** is similar except it loads the certificate chain into **ssl**.

SSL_CTX_use_PrivateKey() adds **pkey** as private key to **ctx**. **SSL_CTX_use_RSAPrivateKey()** adds the private key **rsa** of type RSA to **ctx**. **SSL_use_PrivateKey()** adds **pkey** as private key to **ssl**; **SSL_use_RSAPrivateKey()** adds **rsa** as private key of type RSA to **ssl**. If a certificate has already been set and the private key does not belong to the certificate an error is returned. To change a [certificate/private-key] pair, the new certificate needs to be set first with **SSL_use_certificate()** or **SSL_CTX_use_certificate()** before setting the private key with **SSL_CTX_use_PrivateKey()** or **SSL_use_PrivateKey()**.

SSL_CTX_use_cert_and_key() and **SSL_use_cert_and_key()** assign the X.509 certificate **x**, private key **key**, and certificate **chain** onto the corresponding **ssl** or **ctx**. The **pkey** argument must be the private

key of the X.509 certificate **x**. If the **override** argument is 0, then **x**, **pkey** and **chain** are set only if all were not previously set. If **override** is non-0, then the certificate, private key and chain certs are always set. If **pkey** is NULL, then the public key of **x** is used as the private key. This is intended to be used with hardware (via the ENGINE interface) that stores the private key securely, such that it cannot be accessed by OpenSSL. The reference count of the public key is incremented (twice if there is no private key); it is not copied nor duplicated. This allows all private key validations checks to succeed without an actual private key being assigned via **SSL_CTX_use_PrivateKey()**, etc.

SSL_CTX_use_PrivateKey_ASN1() adds the private key of type **pk** stored at memory location **d** (length **len**) to **ctx**. **SSL_CTX_use_RSAPrivateKey_ASN1()** adds the private key of type RSA stored at memory location **d** (length **len**) to **ctx**. **SSL_use_PrivateKey_ASN1()** and **SSL_use_RSAPrivateKey_ASN1()** add the private key to **ssl**.

SSL_CTX_use_PrivateKey_file() adds the first private key found in **file** to **ctx**. The formatting **type** of the private key must be specified from the known types SSL_FILETYPE_PEM, SSL_FILETYPE_ASN1. **SSL_CTX_use_RSAPrivateKey_file()** adds the first private RSA key found in **file** to **ctx**. **SSL_use_PrivateKey_file()** adds the first private key found in **file** to **ssl**; **SSL_use_RSAPrivateKey_file()** adds the first private RSA key found to **ssl**.

SSL_CTX_check_private_key() checks the consistency of a private key with the corresponding certificate loaded into **ctx**. If more than one key/certificate pair (RSA/DSA) is installed, the last item installed will be checked. If e.g. the last item was an RSA certificate or key, the RSA key/certificate pair will be checked. **SSL_check_private_key()** performs the same check for **ssl**. If no key/certificate was explicitly added for this **ssl**, the last item added into **ctx** will be checked.

NOTES

The internal certificate store of OpenSSL can hold several private key/certificate pairs at a time. The certificate used depends on the cipher selected, see also **SSL_CTX_set_cipher_list**(3).

When reading certificates and private keys from file, files of type SSL_FILETYPE_ASN1 (also known as **DER**, binary encoding) can only contain one certificate or private key, consequently **SSL_CTX_use_certificate_chain_file()** is only applicable to PEM formatting. Files of type SSL_FILETYPE_PEM can contain more than one item.

SSL_CTX_use_certificate_chain_file() adds the first certificate found in the file to the certificate store. The other certificates are added to the store of chain certificates using **SSL_CTX_add1_chain_cert**(3). Note: versions of OpenSSL before 1.0.2 only had a single certificate chain store for all certificate types, OpenSSL 1.0.2 and later have a separate chain store for each type.
SSL_CTX_use_certificate_chain_file() should be used instead of the **SSL_CTX_use_certificate_file()** function in order to allow the use of complete certificate chains even when no trusted CA storage is

used or when the CA issuing the certificate shall not be added to the trusted CA storage.

If additional certificates are needed to complete the chain during the TLS negotiation, CA certificates are additionally looked up in the locations of trusted CA certificates, see **SSL_CTX_load_verify_locations**(3).

The private keys loaded from file can be encrypted. In order to successfully load encrypted keys, a function returning the passphrase must have been supplied, see **SSL_CTX_set_default_passwd_cb**(3). (Certificate files might be encrypted as well from the technical point of view, it however does not make sense as the data in the certificate is considered public anyway.)

All of the functions to set a new certificate will replace any existing certificate of the same type that has already been set. Similarly all of the functions to set a new private key will replace any private key that has already been set. Applications should call **SSL_CTX_check_private_key**(3) or **SSL_check_private_key**(3) as appropriate after loading a new certificate and private key to confirm that the certificate and key match.

RETURN VALUES

On success, the functions return 1. Otherwise check out the error stack to find out the reason.

SEE ALSO

ssl(7), **SSL_new**(3), **SSL_clear**(3), **SSL_CTX_load_verify_locations**(3), **SSL_CTX_set_default_passwd_cb**(3), **SSL_CTX_set_cipher_list**(3), **SSL_CTX_set_client_CA_list**(3), **SSL_CTX_set_client_cert_cb**(3), **SSL_CTX_add_extra_chain_cert**(3)

COPYRIGHT