



*Full credit is given to the above companies including the OS that this PDF file was generated!*

## **Red Hat Enterprise Linux Release 9.2 Manual Pages on 'SSL\_select\_current\_cert.3ossl' command**

**\$ man SSL\_select\_current\_cert.3ossl**

SSL\_CTX\_ADD1\_CHAIN\_CERT(3ossl) OpenSSL SSL\_CTX\_ADD1\_CHAIN\_CERT(3ossl)

### NAME

SSL\_CTX\_set0\_chain, SSL\_CTX\_set1\_chain, SSL\_CTX\_add0\_chain\_cert,  
SSL\_CTX\_add1\_chain\_cert, SSL\_CTX\_get0\_chain\_certs,  
SSL\_CTX\_clear\_chain\_certs, SSL\_set0\_chain, SSL\_set1\_chain,  
SSL\_add0\_chain\_cert, SSL\_add1\_chain\_cert, SSL\_get0\_chain\_certs,  
SSL\_clear\_chain\_certs, SSL\_CTX\_build\_cert\_chain, SSL\_build\_cert\_chain,  
SSL\_CTX\_select\_current\_cert, SSL\_select\_current\_cert,  
SSL\_CTX\_set\_current\_cert, SSL\_set\_current\_cert - extra chain  
certificate processing

### SYNOPSIS

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

```
int SSL_CTX_set0_chain(SSL_CTX *ctx, STACK_OF(X509) *sk);
```

```
int SSL_CTX_set1_chain(SSL_CTX *ctx, STACK_OF(X509) *sk);
```

```
int SSL_CTX_add0_chain_cert(SSL_CTX *ctx, X509 *x509);
```

```
int SSL_CTX_add1_chain_cert(SSL_CTX *ctx, X509 *x509);
```

```
int SSL_CTX_get0_chain_certs(SSL_CTX *ctx, STACK_OF(X509) **sk);
```

```
int SSL_CTX_clear_chain_certs(SSL_CTX *ctx);
```

```
int SSL_set0_chain(SSL *ssl, STACK_OF(X509) *sk);
```

```

int SSL_set1_chain(SSL *ssl, STACK_OF(X509) *sk);
int SSL_add0_chain_cert(SSL *ssl, X509 *x509);
int SSL_add1_chain_cert(SSL *ssl, X509 *x509);
int SSL_get0_chain_certs(SSL *ssl, STACK_OF(X509) **sk);
int SSL_clear_chain_certs(SSL *ssl);

int SSL_CTX_build_cert_chain(SSL_CTX *ctx, flags);
int SSL_build_cert_chain(SSL *ssl, flags);

int SSL_CTX_select_current_cert(SSL_CTX *ctx, X509 *x509);
int SSL_select_current_cert(SSL *ssl, X509 *x509);
int SSL_CTX_set_current_cert(SSL_CTX *ctx, long op);
int SSL_set_current_cert(SSL *ssl, long op);

```

## DESCRIPTION

SSL\_CTX\_set0\_chain() and SSL\_CTX\_set1\_chain() set the certificate chain associated with the current certificate of ctx to sk.

SSL\_CTX\_add0\_chain\_cert() and SSL\_CTX\_add1\_chain\_cert() append the single certificate x509 to the chain associated with the current certificate of ctx.

SSL\_CTX\_get0\_chain\_certs() retrieves the chain associated with the current certificate of ctx.

SSL\_CTX\_clear\_chain\_certs() clears any existing chain associated with the current certificate of ctx. (This is implemented by calling SSL\_CTX\_set0\_chain() with sk set to NULL).

SSL\_CTX\_build\_cert\_chain() builds the certificate chain for ctx. Normally this uses the chain store or the verify store if the chain store is not set. If the function is successful the built chain will replace any existing chain. The flags parameter can be set to

SSL\_BUILD\_CHAIN\_FLAG\_UNTRUSTED to use existing chain certificates as untrusted CAs, SSL\_BUILD\_CHAIN\_FLAG\_NO\_ROOT to omit the root CA from the built chain, SSL\_BUILD\_CHAIN\_FLAG\_CHECK to use all existing chain certificates only to build the chain (effectively sanity checking and rearranging them if necessary), the flag SSL\_BUILD\_CHAIN\_FLAG\_IGNORE\_ERROR ignores any errors during verification: if flag SSL\_BUILD\_CHAIN\_FLAG\_CLEAR\_ERROR is also set verification errors are cleared from the error queue. Details of the chain building process are described in "Certification Path Building" in openssl-verification-options(1).

Each of these functions operates on the current end entity (i.e. server or client) certificate. This is the last certificate loaded or selected on the corresponding ctx structure.

SSL\_CTX\_select\_current\_cert() selects x509 as the current end entity certificate, but only if x509 has already been loaded into ctx using a function such as SSL\_CTX\_use\_certificate().

SSL\_set0\_chain(), SSL\_set1\_chain(), SSL\_add0\_chain\_cert(), SSL\_add1\_chain\_cert(), SSL\_get0\_chain\_certs(), SSL\_clear\_chain\_certs(), SSL\_build\_cert\_chain(), SSL\_select\_current\_cert() and SSL\_set\_current\_cert() are similar except they apply to SSL structure ssl.

SSL\_CTX\_set\_current\_cert() changes the current certificate to a value based on the op argument. Currently op can be SSL\_CERT\_SET\_FIRST to use the first valid certificate or SSL\_CERT\_SET\_NEXT to set the next valid certificate after the current certificate. These two operations can be used to iterate over all certificates in an SSL\_CTX structure.

SSL\_set\_current\_cert() also supports the option SSL\_CERT\_SET\_SERVER.

If ssl is a server and has sent a certificate to a connected client

this option sets that certificate to the current certificate and returns 1. If the negotiated cipher suite is anonymous (and thus no certificate will be sent) 2 is returned and the current certificate is unchanged. If ssl is not a server or a certificate has not been sent 0 is returned and the current certificate is unchanged.

All these functions are implemented as macros. Those containing a 1 increment the reference count of the supplied certificate or chain so it must be freed at some point after the operation. Those containing a 0 do not increment reference counts and the supplied certificate or chain MUST NOT be freed after the operation.

## NOTES

The chains associate with an SSL\_CTX structure are copied to any SSL structures when SSL\_new() is called. SSL structures will not be affected by any chains subsequently changed in the parent SSL\_CTX.

One chain can be set for each key type supported by a server. So, for example, an RSA and a DSA certificate can (and often will) have different chains.

The functions SSL\_CTX\_build\_cert\_chain() and SSL\_build\_cert\_chain() can be used to check application configuration and to ensure any necessary subordinate CAs are sent in the correct order. Misconfigured applications sending incorrect certificate chains often cause problems with peers.

For example an application can add any set of certificates using SSL\_CTX\_use\_certificate\_chain\_file() then call SSL\_CTX\_build\_cert\_chain() with the option SSL\_BUILD\_CHAIN\_FLAG\_CHECK to check and reorder them.

Applications can issue non fatal warnings when checking chains by

setting the flag `SSL_BUILD_CHAIN_FLAG_IGNORE_ERRORS` and checking the return value.

Calling `SSL_CTX_build_cert_chain()` or `SSL_build_cert_chain()` is more efficient than the automatic chain building as it is only performed once. Automatic chain building is performed on each new session.

If any certificates are added using these functions no certificates added using `SSL_CTX_add_extra_chain_cert()` will be used.

## RETURN VALUES

`SSL_set_current_cert()` with `SSL_CERT_SET_SERVER` return 1 for success, 2 if no server certificate is used because the cipher suites is anonymous and 0 for failure.

`SSL_CTX_build_cert_chain()` and `SSL_build_cert_chain()` return 1 for success and 0 for failure. If the flag `SSL_BUILD_CHAIN_FLAG_IGNORE_ERROR` and a verification error occurs then 2 is returned.

All other functions return 1 for success and 0 for failure.

## SEE ALSO

`ssl(7)`, `SSL_CTX_add_extra_chain_cert(3)`

## HISTORY

These functions were added in OpenSSL 1.0.2.

## COPYRIGHT

Copyright 2013-2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (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>>.

3.0.7                    2023-07-13   SSL\_CTX\_ADD1\_CHAIN\_CERT(3ossl)