

**NAME**

SSL\_CTX\_set\_info\_callback, SSL\_CTX\_get\_info\_callback, SSL\_set\_info\_callback,  
 SSL\_get\_info\_callback - handle information callback for SSL connections

**SYNOPSIS**

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

void SSL_CTX_set_info_callback(SSL_CTX *ctx, void (*callback)());
void (*SSL_CTX_get_info_callback(const SSL_CTX *ctx))();

void SSL_set_info_callback(SSL *ssl, void (*callback)());
void (*SSL_get_info_callback(const SSL *ssl))();
```

**DESCRIPTION**

*SSL\_CTX\_set\_info\_callback()* sets the **callback** function, that can be used to obtain state information for SSL objects created from **ctx** during connection setup and use. The setting for **ctx** is overridden from the setting for a specific SSL object, if specified. When **callback** is NULL, no callback function is used.

*SSL\_set\_info\_callback()* sets the **callback** function, that can be used to obtain state information for **ssl** during connection setup and use. When **callback** is NULL, the callback setting currently valid for **ctx** is used.

*SSL\_CTX\_get\_info\_callback()* returns a pointer to the currently set information callback function for **ctx**.

*SSL\_get\_info\_callback()* returns a pointer to the currently set information callback function for **ssl**.

**NOTES**

When setting up a connection and during use, it is possible to obtain state information from the SSL/TLS engine. When set, an information callback function is called whenever the state changes, an alert appears, or an error occurs.

The callback function is called as **callback(SSL \*ssl, int where, int ret)**. The **where** argument specifies information about where (in which context) the callback function was called. If **ret** is 0, an error condition occurred. If an alert is handled, **SSL\_CB\_ALERT** is set and **ret** specifies the alert information.

**where** is a bitmask made up of the following bits:

**SSL\_CB\_LOOP**

Callback has been called to indicate state change inside a loop.

**SSL\_CB\_EXIT**

Callback has been called to indicate error exit of a handshake function. (May be soft error with retry option for non-blocking setups.)

**SSL\_CB\_READ**

Callback has been called during read operation.

**SSL\_CB\_WRITE**

Callback has been called during write operation.

**SSL\_CB\_ALERT**

Callback has been called due to an alert being sent or received.

**SSL\_CB\_READ\_ALERT (SSL\_CB\_ALERT|SSL\_CB\_READ)**

**SSL\_CB\_WRITE\_ALERT (SSL\_CB\_ALERT|SSL\_CB\_WRITE)**

**SSL\_CB\_ACCEPT\_LOOP (SSL\_ST\_ACCEPT|SSL\_CB\_LOOP)**

**SSL\_CB\_ACCEPT\_EXIT (SSL\_ST\_ACCEPT|SSL\_CB\_EXIT)**

**SSL\_CB\_CONNECT\_LOOP (SSL\_ST\_CONNECT|SSL\_CB\_LOOP)**

**SSL\_CB\_CONNECT\_EXIT (SSL\_ST\_CONNECT|SSL\_CB\_EXIT)**

**SSL\_CB\_HANDSHAKE\_START**

Callback has been called because a new handshake is started.

SSL\_CB\_HANDSHAKE\_DONE 0x20

Callback has been called because a handshake is finished.

The current state information can be obtained using the [SSL\\_state\\_string\(3\)](#) family of functions.

The **ret** information can be evaluated using the [SSL\\_alert\\_type\\_string\(3\)](#) family of functions.

## RETURN VALUES

[SSL\\_set\\_info\\_callback\(\)](#) does not provide diagnostic information.

[SSL\\_get\\_info\\_callback\(\)](#) returns the current setting.

## EXAMPLES

The following example callback function prints state strings, information about alerts being handled and error messages to the **bio\_err** BIO.

```
void apps_ssl_info_callback(SSL *s, int where, int ret)
{
    const char *str;
    int w;

    w = where & ~SSL_ST_MASK;

    if (w & SSL_ST_CONNECT) str = "SSL_connect";
    else if (w & SSL_ST_ACCEPT) str = "SSL_accept";
    else str = "undefined";

    if (where & SSL_CB_LOOP)
    {
        BIO_printf(bio_err, "%s:%s\n", str, SSL_state_string_long(s));
    }
    else if (where & SSL_CB_ALERT)
    {
        str = (where & SSL_CB_READ) ? "read" : "write";
        BIO_printf(bio_err, "SSL3 alert %s:%s:%s\n",
            str,
            SSL_alert_type_string_long(ret),
            SSL_alert_desc_string_long(ret));
    }
    else if (where & SSL_CB_EXIT)
    {
        if (ret == 0)
            BIO_printf(bio_err, "%s:failed in %s\n",
                str, SSL_state_string_long(s));
        else if (ret < 0)
        {
            BIO_printf(bio_err, "%s:error in %s\n",
                str, SSL_state_string_long(s));
        }
    }
}
```

## SEE ALSO

[ssl\(3\)](#), [SSL\\_state\\_string\(3\)](#), [SSL\\_alert\\_type\\_string\(3\)](#)

## COPYRIGHT

Copyright 2001-2016 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>>.