## NAME

SSL_CTX_set_mode, SSL_CTX_clear_mode, SSL_set_mode, SSL_clear_mode, SSL_CTX_get_mode, SSL_get_mode - manipulate SSL engine mode

## SYNOPSIS

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

long SSL_CTX_set_mode(SSL_CTX *ctx, long mode);
long SSL_CTX_clear_mode(SSL_CTX *ctx, long mode);
long SSL_set_mode(SSL *ssl, long mode);
long SSL_clear_mode(SSL *ssl, long mode);

long SSL_CTX_get_mode(SSL_CTX *ctx);
long SSL_get_mode(SSL *ssl);
```

## DESCRIPTION

**SSL_CTX_set_mode()** adds the mode set via bit-mask in **mode** to **ctx**. Options already set before are not cleared. **SSL_CTX_clear_mode()** removes the mode set via bit-mask in **mode** from **ctx**.

**SSL_set_mode()** adds the mode set via bit-mask in **mode** to **ssl**. Options already set before are not cleared. **SSL_clear_mode()** removes the mode set via bit-mask in **mode** from **ssl**.

**SSL_CTX_get_mode()** returns the mode set for **ctx**.

**SSL_get_mode()** returns the mode set for **ssl**.

## NOTES

The following mode changes are available:

SSL_MODE_ENABLE_PARTIAL_WRITE

Allow SSL_write_ex(..., n, &r) to return with 0 < r < n (i.e. report success when just a single record has been written). This works in a similar way for **SSL_write()**. When not set (the default), **SSL_write_ex()** or **SSL_write()** will only report success once the complete chunk was written. Once **SSL_write_ex()** or **SSL_write()** returns successful, **r** bytes have been written and the next call to **SSL_write_ex()** or **SSL_write()** must only send the n-r bytes left, imitating the behaviour of **write()**.

SSL_MODE_ACCEPT_MOVING_WRITE_BUFFER

Make it possible to retry **SSL_write_ex()** or **SSL_write()** with changed buffer location (the buffer contents must stay the same). This is not the default to avoid the misconception that nonblocking

**SSL_write()** behaves like nonblocking **write()**.

SSL_MODE_AUTO_RETRY

During normal operations, non-application data records might need to be sent or received that the application is not aware of. If a non-application data record was processed, **SSL_read_ex**(3) and **SSL_read**(3) can return with a failure and indicate the need to retry with **SSL_ERROR_WANT_READ**. If such a non-application data record was processed, the flag **SSL_MODE_AUTO_RETRY** causes it to try to process the next record instead of returning.

In a nonblocking environment applications must be prepared to handle incomplete read/write operations. Setting **SSL_MODE_AUTO_RETRY** for a nonblocking **BIO** will process non-application data records until either no more data is available or an application data record has been processed.

In a blocking environment, applications are not always prepared to deal with the functions returning intermediate reports such as retry requests, and setting the **SSL_MODE_AUTO_RETRY** flag will cause the functions to only return after successfully processing an application data record or a failure.

Turning off **SSL_MODE_AUTO_RETRY** can be useful with blocking **BIO**s in case they are used in combination with something like **select()** or **poll()**. Otherwise the call to **SSL_read()** or **SSL_read_ex()** might hang when a non-application record was sent and no application data was sent.

SSL_MODE_RELEASE_BUFFERS

When we no longer need a read buffer or a write buffer for a given SSL, then release the memory we were using to hold it. Using this flag can save around 34k per idle SSL connection. This flag has no effect on SSL v2 connections, or on DTLS connections.

SSL_MODE_SEND_FALLBACK_SCSV

Send TLS_FALLBACK_SCSV in the ClientHello. To be set only by applications that reconnect with a downgraded protocol version; see draft-ietf-tls-downgrade-scsv-00 for details.

DO NOT ENABLE THIS if your application attempts a normal handshake. Only use this in explicit fallback retries, following the guidance in draft-ietf-tls-downgrade-scsv-00.

SSL_MODE_ASYNC

Enable asynchronous processing. TLS I/O operations may indicate a retry with SSL_ERROR_WANT_ASYNC with this mode set if an asynchronous capable engine is used to perform cryptographic operations. See **SSL_get_error**(3).

SSL_MODE_DTLS_SCTP_LABEL_LENGTH_BUG

Older versions of OpenSSL had a bug in the computation of the label length used for computing the endpoint-pair shared secret. The bug was that the terminating zero was included in the length of the label. Setting this option enables this behaviour to allow interoperability with such broken implementations. Please note that setting this option breaks interoperability with correct implementations. This option only applies to DTLS over SCTP.

All modes are off by default except for SSL_MODE_AUTO_RETRY which is on by default since 1.1.1.

## RETURN VALUES

**SSL_CTX_set_mode()** and **SSL_set_mode()** return the new mode bit-mask after adding **mode**.

**SSL_CTX_get_mode()** and **SSL_get_mode()** return the current bit-mask.

## SEE ALSO

**ssl**(7), **SSL_read_ex**(3), **SSL_read**(3), **SSL_write_ex**(3) or **SSL_write**(3), **SSL_get_error**(3)

## HISTORY

SSL_MODE_ASYNC was added in OpenSSL 1.1.0.

## COPYRIGHT