New APIs for DTLS Multicast

# wolfSSL\_CTX\_mcast\_set\_member\_id

## Synopsis

#include <wolfssl/ssl.h>

int

wolfSSL\_CTX\_set\_mcast\_member\_id(WOLFSSL\_CTX\* ctx, word16 id);

## Description

Sets the multicast group member id for the provided wolfSSL context ctx. Group member IDs are contrained from 0 to 255, inclusive. All sessions made from this context will have the member ID set and will not perform a handshake.

## Return Values

* SSL\_SUCCESS
* SSL\_FATAL\_ERROR – an error was encountered
* BAD\_FUNC\_ARG – ctx is NULL or id is an invalid value

# wolfSSL\_set\_secret

## Synopsis

#include <wolfssl/ssl.h>

int

wolfSSL\_set\_secret(WOLFSSL\* ssl, word16 epoch,  
const byte\* preMasterSecret, word32 preMasterSz,  
const byte\* clientRandom, const byte\* serverRandom,   
const byte\* suite);

## Description

The multicast session is using keying material and a cipher suite dictated by the floating master. This function sets the premaster secret preMasterSecret, with size preMasterSz, for the ssl object as well as the clientRandom and serverRandom values used to calculate the master secret using the hash specified by the cipher suite. If the session is DTLS, the epoch is set to the value provided in the parameter epoch, otherwise the parameter is ignored.

## Return Values

* SSL\_SUCCESS – keying material successfully set and keys generated
* SSL\_FATAL\_ERROR – an error was encountered
* BAD\_FUNC\_ARG

# wolfSSL\_mcast\_read

## Synopsis

#include <wolfssl/ssl.h>

int

wolfSSL\_mcast\_read(WOLFSSL\* ssl, word16\* id, void\* data, int sz);

## Description

Attempts to read a record from the session’s read socket. The decrypted record is placed into the provided buffer data which has a size of sz. The multicast member ID of the peer who sent the record is copied into id if the pointer is non-null. While this function is intended to be used with DTLS multicast sessions, it will work on non-multicast or non-DTLS session. (id will remain unchanged if the session is not multicast.)

## Return Values

* >0 – Positive number of bytes copied into the data buffer.
* 0 – socket is closed or the session is closed
* SSL\_FATAL\_ERROR – An error was encountered while trying to read data. Check the error register for the error code. Note, some errors aren’t truly fatal.
* BAD\_FUNC\_ARG – Either ssl or data was null or sz was 0. There may not be an ssl object to set the error code in.

## Errors

* SSL\_ERROR\_WANT\_READ
* SSL\_ERROR\_WANT\_WRITE
* SSL\_ERROR\_ZERO\_RETURN
* MEMORY\_E

# wolfSSL\_CTX\_mcast\_set\_highwater\_cb

## Synopsis

#include <wolfssl/ssl.h>

typedef (\*CallbackMcastHighwater)(word16 peerId, word32 maxSeq,  
word32 curSeq, void\* ctx);

int

wolfSSL\_CTX\_mcast\_set\_highwater\_cb(WOLFSSL\_CTX\* ctx,  
CallbackMcastHighwater cb, word32 maxSeq,  
word32 first, word32 second);

## Description

Sets the multicast DTLS sequence number highwater mark callback function in the provided wolfSSL context object, ctx. This will be the callback function in any sessions made from the context.

The maxSeq value is the sum of all the highest sequence number of the messages received from all the peers. This is limit age of the key. The value first is the first warning threshold when the highwater callback function is called. The value second is the second warning threshold when the highwater callback function is called.

The prototype for the callback function has four parameters. The first parameter is the ID of the peer that triggered the callback. The next parameter is the maxSeq value. Next is the current sequence number, curSeq. Last the void pointer, ctx, is private data defined by the application to use by the callback function also provided by the application. It is not to be confused with the wolfSSL context.

As an example, set maxSeq to 1000, first to 750, second to 900. When a peer’s received sequence number is 750, the callback function is called. At sequence number 900, the callback function is called again. The callback is called one final time at 1000.

## Return Values

* SSL\_SUCCESS
* BAD\_FUNC\_ARG – Either ssl was null or the bounds did not make sense.

# wolfSSL\_mcast\_set\_highwater\_ctx

## Synopsis

#include <wolfssl/ssl.h>

int

wolfSSL\_mcast\_set\_highwater\_ctx(WOLFSSL\* ssl, void\* ctx);

## Description

Sets the context for , ctx, passed to the multicast DTLS sequence number highwater mark callback function for the session, ssl. The context data is not used by wolfSSL and is only meaningful to the application using wolfSSL providing the callback function.

## Return Values

* SSL\_SUCCESS
* BAD\_FUNC\_ARG – ssl was null

# wolfSSL\_mcast\_peer\_add

## Synopsis

#include <wolfssl/ssl.h>

int

wolfSSL\_mcast\_peer\_add(WOLFSSL\* ssl, word16 peerId, int remove);

## Description

Adds or removes a peerId from the multicast peer ID sequence number tracking. Datagrams from peers whose IDs are not in the list are ignored. When remove is zero, the ID is added; when remove is non-zero, the ID is removed.

## Return Values

* SSL\_SUCCESS
* BAD\_FUNC\_ARG – ssl was null

# wolfSSL\_mcast\_get\_max\_peers

## Synopsis

#include <wolfssl/ssl.h>

int wolfSSL\_mcast\_get\_max\_peers(void);

## Description

Returns the maximum number of allowed peers (WOLFSSL\_MULTICAST\_PEERS).