NAME

SSL_alert_type_string, SSL_alert_type_string_long, SSL_alert_desc_string, SSL_alert_desc_string_long - get textual description of alert information

SYNOPSIS

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

const char *SSL_alert_type_string(int value);
const char *SSL_alert_type_string_long(int value);

const char *SSL_alert_desc_string(int value);
const char *SSL_alert_desc_string_long(int value);
```

DESCRIPTION

SSL_alert_type_string() returns a one letter string indicating the type of the alert specified by **value**.

SSL_alert_type_string_long() returns a string indicating the type of the alert specified by **value**.

SSL_alert_desc_string() returns a two letter string as a short form describing the reason of the alert specified by **value**.

SSL_alert_desc_string_long() returns a string describing the reason of the alert specified by **value**.

NOTES

When one side of an SSL/TLS communication wants to inform the peer about a special situation, it sends an alert. The alert is sent as a special message and does not influence the normal data stream (unless its contents results in the communication being canceled).

A warning alert is sent, when a non-fatal error condition occurs. The ‘‘close notify’’ alert is sent as a warning alert. Other examples for non-fatal errors are certificate errors (‘‘certificate expired’’, ‘‘unsupported certificate’’), for which a warning alert may be sent. (The sending party may however decide to send a fatal error.) The receiving side may cancel the connection on reception of a warning alert on it discretion.

Several alert messages must be sent as fatal alert messages as specified by the TLS RFC. A fatal alert always leads to a connection abort.

RETURN VALUES

The following strings can occur for *SSL_alert_type_string()* or *SSL_alert_type_string_long()*:

‘‘W’’/‘‘warning’’
‘‘F’’/‘‘fatal’’
‘‘U’’/‘‘unknown’’
: This indicates that no support is available for this alert type. Probably **value** does not contain a correct alert message.

The following strings can occur for *SSL_alert_desc_string()* or *SSL_alert_desc_string_long()*:

‘‘CN’’/‘‘close notify’’
: The connection shall be closed. This is a warning alert.

‘‘UM’’/‘‘unexpected message’’
: An inappropriate message was received. This alert is always fatal and should never be observed in communication between proper implementations.

‘‘BM’’/‘‘bad record mac’’
: This alert is returned if a record is received with an incorrect MAC. This message is always fatal.

‘‘DF’’/‘‘decompression failure’’
: The decompression function received improper input (e.g. data that would expand to excessive length). This message is always fatal.

‘‘HF’’/‘‘handshake failure’’
: Reception of a handshake_failure alert message indicates that the sender was unable to negotiate an acceptable set of security parameters given the options available. This is a fatal error.

‘‘NC’’/‘‘no certificate’’
A client, that was asked to send a certificate, does not send a certificate (SSLv3 only).

‘‘BC’’/‘‘bad certificate’’
A certificate was corrupt, contained signatures that did not verify correctly, etc

‘‘UC’’/‘‘unsupported certificate’’
A certificate was of an unsupported type.

‘‘CR’’/‘‘certificate revoked’’
A certificate was revoked by its signer.

‘‘CE’’/‘‘certificate expired’’
A certificate has expired or is not currently valid.

‘‘CU’’/‘‘certificate unknown’’
Some other (unspecified) issue arose in processing the certificate, rendering it unacceptable.

‘‘IP’’/‘‘illegal parameter’’
A field in the handshake was out of range or inconsistent with other fields. This is always fatal.

‘‘DC’’/‘‘decryption failed’’
A TLSCiphertext decrypted in an invalid way: either it wasn’t an even multiple of the block length or its padding values, when checked, weren’t correct. This message is always fatal.

‘‘RO’’/‘‘record overflow’’
A TLSCiphertext record was received which had a length more than 2^14+2048 bytes, or a record decrypted to a TLSCompressed record with more than 2^14+1024 bytes. This message is always fatal.

‘‘CA’’/‘‘unknown CA’’
A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or couldn’t be matched with a known, trusted CA. This message is always fatal.

‘‘AD’’/‘‘access denied’’
A valid certificate was received, but when access control was applied, the sender decided not to proceed with negotiation. This message is always fatal.

‘‘DE’’/‘‘decode error’’
A message could not be decoded because some field was out of the specified range or the length of the message was incorrect. This message is always fatal.

‘‘CY’’/‘‘decrypt error’’
A handshake cryptographic operation failed, including being unable to correctly verify a signature, decrypt a key exchange, or validate a finished message.

‘‘ER’’/‘‘export restriction’’
A negotiation not in compliance with export restrictions was detected; for example, attempting to transfer a 1024 bit ephemeral RSA key for the RSA_EXPORT handshake method. This message is always fatal.

‘‘PV’’/‘‘protocol version’’
The protocol version the client has attempted to negotiate is recognized, but not supported. (For example, old protocol versions might be avoided for security reasons). This message is always fatal.

‘‘IS’’/‘‘insufficient security’’
Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client. This message is always fatal.

‘‘IE’’/‘‘internal error’’
An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue (such as a memory allocation failure). This message is always fatal.

‘‘US’’/‘‘user canceled’’
: This handshake is being canceled for some reason unrelated to a protocol failure. If the user cancels an operation after the handshake is complete, just closing the connection by sending a close_notify is more appropriate. This alert should be followed by a close_notify. This message is generally a warning.

‘‘NR’’/‘‘no renegotiation’’
: Sent by the client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these would normally lead to renegotiation; when that is not appropriate, the recipient should respond with this alert; at that point, the original requester can decide whether to proceed with the connection. One case where this would be appropriate would be where a server has spawned a process to satisfy a request; the process might receive security parameters (key length, authentication, etc.) at startup and it might be difficult to communicate changes to these parameters after that point. This message is always a warning.

‘‘UP’’/‘‘unknown PSK identity’’
: Sent by the server to indicate that it does not recognize a PSK identity or an SRP identity.

‘‘UK’’/‘‘unknown’’
: This indicates that no description is available for this alert type. Probably **value** does not contain a correct alert message.

SEE ALSO

ssl(3) , *SSL_CTX_set_info_callback(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>.