NAME

SSL_CTX_set_client_hello_cb, SSL_client_hello_cb_fn, SSL_client_hello_isv2, SSL_client_hello_get0_legacy_version, SSL_client_hello_get0_random, SSL_client_hello_get0_session_id, SSL_client_hello_get0_ciphers, SSL_client_hello_get0_compression_methods, SSL_client_hello_get1_extensions_present, SSL_client_hello_get0_ext - callback functions for early server-side ClientHello processing

SYNOPSIS

```
typedef int (*SSL_client_hello_cb_fn)(SSL *s, int *al, void *arg);
void SSL_CTX_set_client_hello_cb(SSL_CTX *c, SSL_client_hello_cb_fn *f,
                                 void *arg);
int SSL_client_hello_isv2(SSL *s);
unsigned int SSL_client_hello_get0_legacy_version(SSL *s);
size_t SSL_client_hello_get0_random(SSL *s, const unsigned char **out);
size_t SSL_client_hello_get0_session_id(SSL *s, const unsigned char **out);
size_t SSL_client_hello_get0_ciphers(SSL *s, const unsigned char **out);
size_t SSL_client_hello_get0_compression_methods(SSL *s,
                                                 const unsigned char **out);
int SSL_client_hello_get1_extensions_present(SSL *s, int **out,
                                             size_t *outlen);
int SSL_client_hello_get0_ext(SSL *s, unsigned int type, const unsigned char **out,
                              size_t *outlen);
```

DESCRIPTION

SSL_CTX_set_client_hello_cb() sets the callback function, which is automatically called during the early stages of ClientHello processing on the server. The argument supplied when setting the callback is passed back to the callback at run time. A callback that returns failure (0) will cause the connection to terminate, and callbacks returning failure should indicate what alert value is to be sent in the **al** parameter. A callback may also return a negative value to suspend the handshake, and the handshake function will return immediately. **SSL_get_error**(3) will return SSL_ERROR_WANT_CLIENT_HELLO_CB to indicate that the handshake was suspended. It is the job of the ClientHello callback to store information about the state of the last call if needed to continue. On the next call into the handshake function, the ClientHello callback will be called again, and, if it returns success, normal handshake processing will continue from that point.

SSL_client_hello_isv2() indicates whether the ClientHello was carried in a SSLv2 record and is in the SSLv2 format. The SSLv2 format has substantial differences from the normal SSLv3 format, including using three bytes per cipher suite, and not allowing extensions. Additionally, the SSLv2 format 'challenge' field is exposed via **SSL_client_hello_get0_random()**, padded to SSL3_RANDOM_SIZE bytes with zeros if needed. For SSLv2 format ClientHellos,

SSL_client_hello_get0_compression_methods() returns a dummy list that only includes the null compression method, since the SSLv2 format does not include a mechanism by which to negotiate compression.

SSL_client_hello_get0_random(), **SSL_client_hello_get0_session_id()**, **SSL_client_hello_get0_ciphers()**, and **SSL_client_hello_get0_compression_methods()** provide access to the corresponding ClientHello fields, returning the field length and optionally setting an out pointer to the octets of that field.

Similarly, **SSL_client_hello_get0_ext()** provides access to individual extensions from the ClientHello on a per-extension basis. For the provided wire protocol extension type value, the extension value and length are returned in the output parameters (if present).

SSL_client_hello_get1_extensions_present() can be used prior to **SSL_client_hello_get0_ext()**, to determine which extensions are present in the ClientHello before querying for them. The **out** and **outlen** parameters are both required, and on success the caller must release the storage allocated for ***out** using **OPENSSL_free()**. The contents of ***out** is an array of integers holding the numerical value of the TLS extension types in the order they appear in the ClientHello. ***outlen** contains the number of elements in the array. In situations when the ClientHello has no extensions, the function will return success with ***out** set to NULL and ***outlen** set to 0.

NOTES

The ClientHello callback provides a vast window of possibilities for application code to affect the TLS handshake. A primary use of the callback is to allow the server to examine the server name indication extension provided by the client in order to select an appropriate certificate to present, and make other configuration adjustments relevant to that server name and its configuration. Such configuration changes can include swapping out the associated SSL_CTX pointer, modifying the server's list of permitted TLS versions, changing the server's cipher list in response to the client's cipher list, etc.

It is also recommended that applications utilize a ClientHello callback and not use a servername callback, in order to avoid unexpected behavior that occurs due to the relative order of processing between things like session resumption and the historical servername callback.

The SSL_client_hello_* family of functions may only be called from code executing within a ClientHello callback.

RETURN VALUES

The application's supplied ClientHello callback returns SSL_CLIENT_HELLO_SUCCESS on success, SSL_CLIENT_HELLO_ERROR on failure, and SSL_CLIENT_HELLO_RETRY to suspend processing.

SSL_client_hello_isv2() returns 1 for SSLv2-format ClientHellos and 0 otherwise.

SSL_client_hello_get0_random(), **SSL_client_hello_get0_session_id()**, **SSL_client_hello_get0_ciphers()**, and **SSL_client_hello_get0_compression_methods()** return the length of the corresponding ClientHello fields. If zero is returned, the output pointer should not be assumed to be valid.

SSL_client_hello_get0_ext() returns 1 if the extension of type 'type' is present, and 0 otherwise.

SSL_client_hello_get1_extensions_present() returns 1 on success and 0 on failure.

SEE ALSO

ssl(7), **SSL_CTX_set_tlsext_servername_callback**(3), **SSL_bytes_to_cipher_list**(3)

HISTORY

The SSL ClientHello callback, **SSL_client_hello_isv2()**, **SSL_client_hello_get0_random()**, **SSL_client_hello_get0_session_id()**, **SSL_client_hello_get0_ciphers()**, **SSL_client_hello_get0_compression_methods()**, **SSL_client_hello_get0_ext()**, and **SSL_client_hello_get1_extensions_present()** were added in OpenSSL 1.1.1.

COPYRIGHT