NAME

SSL_extension_supported, SSL_CTX_add_client_custom_ext, SSL_CTX_add_server_custom_ext, custom_ext_add_cb, custom_ext_free_cb, custom_ext_parse_cb - custom TLS extension handling

SYNOPSIS

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

int SSL_CTX_add_client_custom_ext(SSL_CTX *ctx, unsigned int ext_type,
custom_ext_add_cb add_cb,
custom_ext_free_cb free_cb, void *add_arg,
custom_ext_parse_cb parse_cb,
void *parse_arg);

int SSL_CTX_add_server_custom_ext(SSL_CTX *ctx, unsigned int ext_type,
custom_ext_add_cb add_cb,
custom_ext_free_cb free_cb, void *add_arg,
custom_ext_parse_cb parse_cb,
void *parse_arg);

int SSL_extension_supported(unsigned int ext_type);

typedef int (*custom_ext_add_cb)(SSL *s, unsigned int ext_type,
const unsigned char **out,
size_t *outlen, int *al,
void *add_arg);

typedef void (*custom_ext_free_cb)(SSL *s, unsigned int ext_type,
const unsigned char *out,
void *add_arg);

typedef int (*custom_ext_parse_cb)(SSL *s, unsigned int ext_type,
const unsigned char *in,
size_t inlen, int *al,
void *parse_arg);
```

DESCRIPTION

SSL_CTX_add_client_custom_ext() adds a custom extension for a TLS client with extension type **ext_type** and callbacks **add_cb**, **free_cb** and **parse_cb**.

SSL_CTX_add_server_custom_ext() adds a custom extension for a TLS server with extension type **ext_type** and callbacks **add_cb**, **free_cb** and **parse_cb**.

In both cases the extension type must not be handled by OpenSSL internally or an error occurs.

SSL_extension_supported() returns 1 if the extension **ext_type** is handled internally by OpenSSL and 0 otherwise.

EXTENSION CALLBACKS

The callback **add_cb** is called to send custom extension data to be included in ClientHello for TLS clients or ServerHello for servers. The **ext_type** parameter is set to the extension type which will be added and **add_arg** to the value set when the extension handler was added.

If the application wishes to include the extension **ext_type** it should set ***out** to the extension data, set ***outlen** to the length of the extension data and return 1.

If the **add_cb** does not wish to include the extension it must return 0.

If **add_cb** returns -1 a fatal handshake error occurs using the TLS alert value specified in ***al**.

For clients (but not servers) if **add_cb** is set to NULL a zero length extension is added for **ext_type**.

For clients every registered **add_cb** is always called to see if the application wishes to add an extension to ClientHello.

For servers every registered **add_cb** is called once if and only if the corresponding extension was received in ClientHello to see if the application wishes to add the extension to ServerHello. That is, if no corresponding extension was received in ClientHello then **add_cb** will not be called.

If an extension is added (that is **add_cb** returns 1) **free_cb** is called (if it is set) with the value of **out** set by the add callback. It can be used to free up any dynamic extension data set by **add_cb**. Since **out** is constant (to permit use of constant data in **add_cb**) applications may need to cast away const to free the data.

The callback **parse_cb** receives data for TLS extensions. For TLS clients the extension data will come from ServerHello and for TLS servers it will come from ClientHello.

The extension data consists of **inlen** bytes in the buffer **in** for the extension **extension_type**.

If the **parse_cb** considers the extension data acceptable it must return 1. If it returns 0 or a negative value a fatal handshake error occurs using the TLS alert value specified in ***al**.

The buffer **in** is a temporary internal buffer which will not be valid after the callback returns.

NOTES

The **add_arg** and **parse_arg** parameters can be set to arbitrary values which will be passed to the corresponding callbacks. They can, for example, be used to store the extension data received in a convenient structure or pass the extension data to be added or freed when adding extensions.

The **ext_type** parameter corresponds to the **extension_type** field of RFC5246 et al. It is **not** a NID.

If the same custom extension type is received multiple times a fatal **decode_error** alert is sent and the handshake aborts. If a custom extension is received in ServerHello which was not sent in ClientHello a fatal **unsupported_extension** alert is sent and the handshake is aborted. The ServerHello **add_cb** callback is only called if the corresponding extension was received in ClientHello. This is compliant with the TLS specifications. This behaviour ensures that each callback is called at most once and that an application can never send unsolicited extensions.

RETURN VALUES

SSL_CTX_add_client_custom_ext() and *SSL_CTX_add_server_custom_ext()* return 1 for success and 0 for failure. A failure can occur if an attempt is made to add the same **ext_type** more than once, if an attempt is made to use an extension type handled internally by OpenSSL or if an internal error occurs (for example a memory allocation failure).

SSL_extension_supported() returns 1 if the extension **ext_type** is handled internally by OpenSSL and 0 otherwise.

COPYRIGHT