NAME

SSL_CTX_set_async_callback, SSL_CTX_set_async_callback_arg, SSL_set_async_callback, SSL_set_async_callback_arg, SSL_get_async_status, SSL_async_callback_fn - manage asynchronous operations

SYNOPSIS

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

 typedef int (*SSL_async_callback_fn)(SSL *s, void *arg);
 int SSL_CTX_set_async_callback(SSL_CTX *ctx, SSL_async_callback_fn callback);
 int SSL_CTX_set_async_callback_arg(SSL_CTX *ctx, void *arg);
 int SSL_set_async_callback(SSL *s, SSL_async_callback_fn callback);
 int SSL_set_async_callback_arg(SSL *s, void *arg);
 int SSL_get_async_status(SSL *s, int *status);
```

DESCRIPTION

SSL_CTX_set_async_callback() sets an asynchronous callback function. All **SSL** objects generated based on this **SSL_CTX** will get this callback. If an engine supports the callback mechanism, it will be automatically called if **SSL_MODE_ASYNC** has been set and an asynchronous capable engine completes a cryptography operation to notify the application to resume the paused work flow.

SSL_CTX_set_async_callback_arg() sets the callback argument.

SSL_set_async_callback() allows an application to set a callback in an asynchronous **SSL** object, so that when an engine completes a cryptography operation, the callback will be called to notify the application to resume the paused work flow.

SSL_set_async_callback_arg() sets an argument for the **SSL** object when the above callback is called.

SSL_get_async_status() returns the engine status. This function facilitates the communication from the engine to the application. During an SSL session, cryptographic operations are dispatched to an engine. The engine status is very useful for an application to know if the operation has been successfully dispatched. If the engine does not support this additional callback method, **ASYNC_STATUS_UNSUPPORTED** will be returned. See **ASYNC_WAIT_CTX_set_status()** for a description of all of the status values.

An example of the above functions would be the following:

1. Application sets the async callback and callback data on an SSL connection by calling **SSL_set_async_callback()**.

2. Application sets **SSL_MODE_ASYNC** and makes an asynchronous SSL call

3. OpenSSL submits the asynchronous request to the engine. If a retry occurs at this point then the status within the **ASYNC_WAIT_CTX** would be set and the async callback function would be called (goto Step 7).

4. The OpenSSL engine pauses the current job and returns, so that the application can continue processing other connections.

5. At a future point in time (probably via a polling mechanism or via an interrupt) the engine will become aware that the asynchronous request has finished processing.

6. The engine will call the application's callback passing the callback data as a parameter.

7. The callback function should then run. Note: it is a requirement that the callback function is small and nonblocking as it will be run in the context of a polling mechanism or an interrupt.

8. It is the application's responsibility via the callback function to schedule recalling the OpenSSL asynchronous function and to continue processing.

9. The callback function has the option to check the status returned via **SSL_get_async_status()** to determine whether a retry happened instead of the request being submitted, allowing different processing if required.

RETURN VALUES

SSL_CTX_set_async_callback(), **SSL_set_async_callback()**, **SSL_CTX_set_async_callback_arg()**, **SSL_CTX_set_async_callback_arg()** and **SSL_get_async_status()** return 1 on success or 0 on error.

SEE ALSO

ssl(7)

HISTORY

SSL_CTX_set_async_callback(), **SSL_CTX_set_async_callback_arg()**, **SSL_set_async_callback()**, **SSL_set_async_callback_arg()** and **SSL_get_async_status()** were first added to OpenSSL 3.0.

COPYRIGHT