Rocky Enterprise Linux 9.2 Manual Pages on command 'SSL_get0_next_proto_negotiated.3ossl'

$ man SSL_get0_next_proto_negotiated.3ossl

SSL_CTX_SET_ALPN_SELECT_CB(3ossl) OpenSSL SSL_CTX_SET_ALPN_SELECT_CB(3ossl)

NAME

SSL_CTX_set_alpn_protos, SSL_set_alpn_protos,

SSL_CTX_set_alpn_select_cb, SSL_CTX_set_next_proto_select_cb,

SSL_CTX_set_next_protos_advertised_cb, SSL_select_next_proto,

SSL_get0_alpn_selected, SSL_get0_next_proto_negotiated - handle

application layer protocol negotiation (ALPN)

SYNOPSIS

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

int SSL_CTX_set_alpn_protos(SSL_CTX *ctx, const unsigned char *protos,
                            unsigned int protos_len);
int SSL_set_alpn_protos(SSL *ssl, const unsigned char *protos,
                        unsigned int protos_len);
void SSL_CTX_set_alpn_select_cb(SSL_CTX *ctx,
                                int (*cb) (SSL *ssl,
```

```
                                const unsigned char **out,
                                unsigned char *outlen,
                                const unsigned char *in,
                                unsigned int inlen,
                                void *arg), void *arg);
    void SSL_get0_alpn_selected(const SSL *ssl, const unsigned char **data,
                                unsigned int *len);

    void SSL_CTX_set_next_protos_advertised_cb(SSL_CTX *ctx,
                                int (*cb)(SSL *ssl,
                                        const unsigned char **out,
                                        unsigned int *outlen,
                                        void *arg),
                                void *arg);
    void SSL_CTX_set_next_proto_select_cb(SSL_CTX *ctx,
                                int (*cb)(SSL *s,
                                        unsigned char **out,
                                        unsigned char *outlen,
                                        const unsigned char *in,
                                        unsigned int inlen,
                                        void *arg),
                                void *arg);
    int SSL_select_next_proto(unsigned char **out, unsigned char *outlen,
                                const unsigned char *server,
                                unsigned int server_len,
                                const unsigned char *client,
                                unsigned int client_len);
    void SSL_get0_next_proto_negotiated(const SSL *s, const unsigned char **data,
                                unsigned *len);
```

DESCRIPTION

SSL_CTX_set_alpn_protos() and SSL_set_alpn_protos() are used by the client to set the list of protocols available to be negotiated. The

protos must be in protocol-list format, described below. The length of protos is specified in protos_len.

SSL_CTX_set_alpn_select_cb() sets the application callback cb used by a server to select which protocol to use for the incoming connection. When cb is NULL, ALPN is not used. The arg value is a pointer which is passed to the application callback.

cb is the application defined callback. The in, inlen parameters are a vector in protocol-list format. The value of the out, outlen vector should be set to the value of a single protocol selected from the in, inlen vector. The out buffer may point directly into in, or to a buffer that outlives the handshake. The arg parameter is the pointer set via SSL_CTX_set_alpn_select_cb().

SSL_select_next_proto() is a helper function used to select protocols. It implements the standard protocol selection. It is expected that this function is called from the application callback cb. The protocol data in server, server_len and client, client_len must be in the protocol-list format described below. The first item in the server, server_len list that matches an item in the client, client_len list is selected, and returned in out, outlen. The out value will point into either server or client, so it should be copied immediately. If no match is found, the first item in client, client_len is returned in out, outlen. This function can also be used in the NPN callback.

SSL_CTX_set_next_proto_select_cb() sets a callback cb that is called when a client needs to select a protocol from the server's provided list, and a user-defined pointer argument arg which will be passed to this callback. For the callback itself, out must be set to point to the selected protocol (which may be within in). The length of the protocol name must be written into outlen. The server's advertised protocols are provided in in and inlen. The callback can assume that in

is syntactically valid. The client must select a protocol. It is fatal to the connection if this callback returns a value other than SSL_TLSEXT_ERR_OK. The arg parameter is the pointer set via SSL_CTX_set_next_proto_select_cb().

SSL_CTX_set_next_protos_advertised_cb() sets a callback cb that is called when a TLS server needs a list of supported protocols for Next Protocol Negotiation. The returned list must be in protocol-list format, described below. The list is returned by setting out to point to it and outlen to its length. This memory will not be modified, but the SSL does keep a reference to it. The callback should return SSL_TLSEXT_ERR_OK if it wishes to advertise. Otherwise, no such extension will be included in the ServerHello.

SSL_get0_alpn_selected() returns a pointer to the selected protocol in data with length len. It is not NUL-terminated. data is set to NULL and len is set to 0 if no protocol has been selected. data must not be freed.

SSL_get0_next_proto_negotiated() sets data and len to point to the client's requested protocol for this connection. If the client did not request any protocol or NPN is not enabled, then data is set to NULL and len to 0. Note that the client can request any protocol it chooses. The value returned from this function need not be a member of the list of supported protocols provided by the callback.

NOTES

The protocol-lists must be in wire-format, which is defined as a vector of nonempty, 8-bit length-prefixed, byte strings. The length-prefix byte is not included in the length. Each string is limited to 255 bytes. A byte-string length of 0 is invalid. A truncated byte-string is invalid. The length of the vector is not in the vector itself, but in a separate variable.

Example:

```
unsigned char vector[] = {
    6, 's', 'p', 'd', 'y', '/', '1',
    8, 'h', 't', 't', 'p', '/', '1', '.', '1'
};
unsigned int length = sizeof(vector);
```

The ALPN callback is executed after the servername callback; as that servername callback may update the SSL_CTX, and subsequently, the ALPN callback.

If there is no ALPN proposed in the ClientHello, the ALPN callback is not invoked.

RETURN VALUES

SSL_CTX_set_alpn_protos() and SSL_set_alpn_protos() return 0 on success, and non-0 on failure. WARNING: these functions reverse the return value convention.

SSL_select_next_proto() returns one of the following:

OPENSSL_NPN_NEGOTIATED

A match was found and is returned in out, outlen.

OPENSSL_NPN_NO_OVERLAP

No match was found. The first item in client, client_len is returned in out, outlen.

The ALPN select callback cb, must return one of the following:

SSL_TLSEXT_ERR_OK

ALPN protocol selected.

SSL_TLSEXT_ERR_ALERT_FATAL

There was no overlap between the client's supplied list and the server configuration.

SSL_TLSEXT_ERR_NOACK

ALPN protocol not selected, e.g., because no ALPN protocols are configured for this connection.

The callback set using SSL_CTX_set_next_proto_select_cb() should return SSL_TLSEXT_ERR_OK if successful. Any other value is fatal to the connection.

The callback set using SSL_CTX_set_next_protos_advertised_cb() should return SSL_TLSEXT_ERR_OK if it wishes to advertise. Otherwise, no such extension will be included in the ServerHello.

SEE ALSO

ssl(7), SSL_CTX_set_tlsext_servername_callback(3), SSL_CTX_set_tlsext_servername_arg(3)

COPYRIGHT