## NAME

SSL_CTX_set_dh_auto, SSL_set_dh_auto, SSL_CTX_set0_tmp_dh_pkey, SSL_set0_tmp_dh_pkey, SSL_CTX_set_tmp_dh_callback, SSL_CTX_set_tmp_dh, SSL_set_tmp_dh_callback, SSL_set_tmp_dh - handle DH keys for ephemeral key exchange

## SYNOPSIS

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

long SSL_CTX_set_dh_auto(SSL_CTX *ctx, int onoff);
long SSL_set_dh_auto(SSL *s, int onoff);
int SSL_CTX_set0_tmp_dh_pkey(SSL_CTX *ctx, EVP_PKEY *dhpkey);
int SSL_set0_tmp_dh_pkey(SSL *s, EVP_PKEY *dhpkey);
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros**(7):

```
void SSL_CTX_set_tmp_dh_callback(SSL_CTX *ctx,
                                 DH *(*tmp_dh_callback)(SSL *ssl, int is_export,
                                                        int keylength));
long SSL_CTX_set_tmp_dh(SSL_CTX *ctx, DH *dh);

void SSL_set_tmp_dh_callback(SSL *ctx,
                             DH *(*tmp_dh_callback)(SSL *ssl, int is_export,
                                                    int keylength));
long SSL_set_tmp_dh(SSL *ssl, DH *dh);
```

## DESCRIPTION

The functions described on this page are relevant for servers only.

Some ciphersuites may use ephemeral Diffie-Hellman (DH) key exchange. In these cases, the session data is negotiated using the ephemeral/temporary DH key and the key supplied and certified by the certificate chain is only used for signing. Anonymous ciphers (without a permanent server key) also use ephemeral DH keys.

Using ephemeral DH key exchange yields forward secrecy as the connection can only be decrypted when the DH key is known. By generating a temporary DH key inside the server application that is lost when the application is left, it becomes impossible for an attacker to decrypt past sessions, even if they get hold of the normal (certified) key, as this key was only used for signing.

In order to perform a DH key exchange the server must use a DH group (DH parameters) and generate a DH key. The server will always generate a new DH key during the negotiation.

As generating DH parameters is extremely time consuming, an application should not generate the parameters on the fly. DH parameters can be reused, as the actual key is newly generated during the negotiation.

Typically applications should use well know DH parameters that have built-in support in OpenSSL. The macros **SSL_CTX_set_dh_auto()** and **SSL_set_dh_auto()** configure OpenSSL to use the default built-in DH parameters for the **SSL_CTX** and **SSL** objects respectively. Passing a value of 1 in the *onoff* parameter switches the feature on, and passing a value of 0 switches it off. The default setting is off.

If "auto" DH parameters are switched on then the parameters will be selected to be consistent with the size of the key associated with the server's certificate. If there is no certificate (e.g. for PSK ciphersuites), then it it will be consistent with the size of the negotiated symmetric cipher key.

Applications may supply their own DH parameters instead of using the built-in values. This approach is discouraged and applications should in preference use the built-in parameter support described above. Applications wishing to supply their own DH parameters should call **SSL_CTX_set0_tmp_dh_pkey()** or **SSL_set0_tmp_dh_pkey()** to supply the parameters for the **SSL_CTX** or **SSL** respectively. The parameters should be supplied in the *dhpkey* argument as an **EVP_PKEY** containing DH parameters. Ownership of the *dhpkey* value is passed to the **SSL_CTX** or **SSL** object as a result of this call, and so the caller should not free it if the function call is successful.

The deprecated macros **SSL_CTX_set_tmp_dh()** and **SSL_set_tmp_dh()** do the same thing as **SSL_CTX_set0_tmp_dh_pkey()** and **SSL_set0_tmp_dh_pkey()** except that the DH parameters are supplied in a **DH** object instead in the *dh* argument, and ownership of the **DH** object is retained by the application. Applications should use "auto" parameters instead, or call **SSL_CTX_set0_tmp_dh_pkey()** or **SSL_set0_tmp_dh_pkey()** as appropriate.

An application may instead specify the DH parameters via a callback function using the functions **SSL_CTX_set_tmp_dh_callback()** or **SSL_set_tmp_dh_callback()** to set the callback for the **SSL_CTX** or **SSL** object respectively. These functions are deprecated. Applications should instead use "auto" parameters, or specify the parameters via **SSL_CTX_set0_tmp_dh_pkey()** or **SSL_set0_tmp_dh_pkey()** as appropriate.

The callback will be invoked during a connection when DH parameters are required. The **SSL** object for the current connection is supplied as an argument. Previous versions of OpenSSL used the **is_export** and **keylength** arguments to control parameter generation for export and non-export cipher suites. Modern OpenSSL does not support export ciphersuites and so these arguments are unused and can be ignored by the callback. The callback should return the parameters to be used in a DH object. Ownership of the DH object is retained by the application and should later be freed.

## RETURN VALUES

All of these functions/macros return 1 for success or 0 on error.

## SEE ALSO

**ssl**(7), **SSL_CTX_set_cipher_list**(3), **SSL_CTX_set_options**(3), **openssl-ciphers**(1), **openssl-dhparam**(1)

## COPYRIGHT