NAME

SSL_CTX_set_default_passwd_cb, SSL_CTX_set_default_passwd_cb_userdata, SSL_CTX_get_default_passwd_cb, SSL_CTX_get_default_passwd_cb_userdata, SSL_set_default_passwd_cb, SSL_set_default_passwd_cb_userdata, SSL_get_default_passwd_cb, SSL_get_default_passwd_cb_userdata - set or get passwd callback for encrypted PEM file handling

SYNOPSIS

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

void SSL_CTX_set_default_passwd_cb(SSL_CTX *ctx, pem_password_cb *cb);
void SSL_CTX_set_default_passwd_cb_userdata(SSL_CTX *ctx, void *u);
pem_password_cb *SSL_CTX_get_default_passwd_cb(SSL_CTX *ctx);
void *SSL_CTX_get_default_passwd_cb_userdata(SSL_CTX *ctx);

void SSL_set_default_passwd_cb(SSL *s, pem_password_cb *cb);
void SSL_set_default_passwd_cb_userdata(SSL *s, void *u);
pem_password_cb *SSL_get_default_passwd_cb(SSL *s);
void *SSL_get_default_passwd_cb_userdata(SSL *s);
```

DESCRIPTION

SSL_CTX_set_default_passwd_cb() sets the default password callback called when loading/storing a PEM certificate with encryption.

SSL_CTX_set_default_passwd_cb_userdata() sets a pointer to userdata, **u**, which will be provided to the password callback on invocation.

SSL_CTX_get_default_passwd_cb() returns a function pointer to the password callback currently set in **ctx**. If no callback was explicitly set, the NULL pointer is returned.

SSL_CTX_get_default_passwd_cb_userdata() returns a pointer to the userdata currently set in **ctx**. If no userdata was explicitly set, the NULL pointer is returned.

SSL_set_default_passwd_cb(), **SSL_set_default_passwd_cb_userdata()**, **SSL_get_default_passwd_cb()** and **SSL_get_default_passwd_cb_userdata()** perform the same function as their SSL_CTX counterparts, but using an SSL object.

The password callback, which must be provided by the application, hands back the password to be used during decryption. On invocation a pointer to userdata is provided. The function must store the password into the provided buffer **buf** which is of size **size**. The actual length of the password must be returned to the calling function. **rwflag** indicates whether the callback is used for reading/decryption

(rwflag=0) or writing/encryption (rwflag=1). For more details, see **pem_password_cb**(3).

NOTES

When loading or storing private keys, a password might be supplied to protect the private key. The way this password can be supplied may depend on the application. If only one private key is handled, it can be practical to have the callback handle the password dialog interactively. If several keys have to be handled, it can be practical to ask for the password once, then keep it in memory and use it several times. In the last case, the password could be stored into the userdata storage and the callback only returns the password already stored.

When asking for the password interactively, the callback can use **rwflag** to check, whether an item shall be encrypted (rwflag=1). In this case the password dialog may ask for the same password twice for comparison in order to catch typos, that would make decryption impossible.

Other items in PEM formatting (certificates) can also be encrypted, it is however not usual, as certificate information is considered public.

RETURN VALUES

These functions do not provide diagnostic information.

EXAMPLES

The following example returns the password provided as userdata to the calling function. The password is considered to be a '\0' terminated string. If the password does not fit into the buffer, the password is truncated.

```
int my_cb(char *buf, int size, int rwflag, void *u)
{
    strncpy(buf, (char *)u, size);
    buf[size - 1] = '\0';
    return strlen(buf);
}
```

SEE ALSO

ssl(7), **SSL_CTX_use_certificate**(3)

HISTORY

SSL_CTX_get_default_passwd_cb(), **SSL_CTX_get_default_passwd_cb_userdata()**, **SSL_set_default_passwd_cb()** and **SSL_set_default_passwd_cb_userdata()** were added in OpenSSL 1.1.0.

COPYRIGHT