NAME

ne_ssl_clicert_read, ne_ssl_clicert_name, ne_ssl_clicert_encrypted, ne_ssl_clicert_decrypt, ne_ssl_clicert_owner, ne_ssl_clicert_free - SSL client certificate handling

SYNOPSIS

#include <ne_ssl.h>

ne_ssl_client_cert *ne_ssl_clicert_read(const char *_filename_);

const char *ne_ssl_clicert_name(const ne_ssl_client_cert *_ccert_);

int ne_ssl_clicert_encrypted(const ne_ssl_client_cert *_ccert_);

int ne_ssl_clicert_decrypt(ne_ssl_client_cert *_ccert_, const char *_password_);

const ne_ssl_certificate *ne_ssl_clicert_owner(const ne_ssl_client_cert *_ccert_);

void ne_ssl_clicert_free(ne_ssl_client_cert *_ccert_);

DESCRIPTION

The **ne_ssl_clicert_read** function reads a client certificate from a PKCS#12-formatted file, and returns an **ne_ssl_client_cert** object. If the client certificate is encrypted, it must be decrypted before it is used. An **ne_ssl_client_cert** object holds a client certificate and the associated private key, not just a certificate; the term "client certificate" will used to refer to this pair.

A client certificate can be in one of two states: *encrypted* or *decrypted*. The **ne_ssl_clicert_encrypted** function will return non-zero if the client certificate is in the *encrypted* state. A client certificate object returned by **ne_ssl_clicert_read** may be initially in either state, depending on whether the file was encrypted or not.

ne_ssl_clicert_decrypt can be used to decrypt a client certificate using the appropriate password. This function must only be called if the object is in the *encrypted* state; if decryption fails, the certificate state does not change, so decryption can be attempted more than once using different passwords.

A client certificate can be given a "friendly name" when it is created; **ne_ssl_clicert_name** will return this name (or NULL if no friendly name was specified). **ne_ssl_clicert_name** can be used when the client certificate is in either the encrypted or decrypted state, and will return the same string for the lifetime of the object.

The function **ne_ssl_clicert_owner** returns the certificate part of the client certificate; it must only be

called if the client certificate is in the *decrypted* state.

When the client certificate is no longer needed, the **ne_ssl_clicert_free** function should be used to destroy the object.

RETURN VALUE

ne_ssl_clicert_read returns a client certificate object, or NULL if the file could not be read. **ne_ssl_clicert_encrypted** returns zero if the object is in the decrypted state, or non-zero if it is in the encrypted state. **ne_ssl_clicert_name** returns a NUL-terminated friendly name string, or NULL. **ne_ssl_clicert_owner** returns a certificate object.

EXAMPLES

The following code reads a client certificate and decrypts it if necessary, then loads it into an HTTP session.

```
ne_ssl_client_cert *ccert;

ccert = ne_ssl_clicert_read("/path/to/client.p12");

if (ccert == NULL) {
  /* handle error... */
} else if (ne_ssl_clicert_encrypted(ccert)) {
  char *password = prompt_for_password();

  if (ne_ssl_clicert_decrypt(ccert, password)) {
    /* could not decrypt! handle error... */
  }
}

ne_ssl_set_clicert(sess, ccert);
```

SEE ALSO

ne_ssl_cert_read

AUTHOR

Joe Orton
Author.

COPYRIGHT