NAME

OSSL_STORE_CTX, OSSL_STORE_post_process_info_fn, OSSL_STORE_open,
OSSL_STORE_open_ex, OSSL_STORE_ctrl, OSSL_STORE_load, OSSL_STORE_eof,
OSSL_STORE_error, OSSL_STORE_close - Types and functions to read objects from a URI

SYNOPSIS

#include <openssl/store.h>

typedef struct ossl_store_ctx_st OSSL_STORE_CTX;

typedef OSSL_STORE_INFO *(*OSSL_STORE_post_process_info_fn)(OSSL_STORE_INFO *,
                                                void *);

OSSL_STORE_CTX *OSSL_STORE_open(const char *uri, const UI_METHOD *ui_method,
                void *ui_data,
                OSSL_STORE_post_process_info_fn post_process,
                void *post_process_data);
OSSL_STORE_CTX *
OSSL_STORE_open_ex(const char *uri, OSSL_LIB_CTX *libctx, const char *propq,
          const UI_METHOD *ui_method, void *ui_data,
          const OSSL_PARAM params[],
          OSSL_STORE_post_process_info_fn post_process,
          void *post_process_data);

OSSL_STORE_INFO *OSSL_STORE_load(OSSL_STORE_CTX *ctx);
int OSSL_STORE_eof(OSSL_STORE_CTX *ctx);
int OSSL_STORE_error(OSSL_STORE_CTX *ctx);
int OSSL_STORE_close(OSSL_STORE_CTX *ctx);

The following function has 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):

int OSSL_STORE_ctrl(OSSL_STORE_CTX *ctx, int cmd, ... /* args */);

DESCRIPTION

These functions help the application to fetch supported objects (see "SUPPORTED OBJECTS" in
OSSL_STORE_INFO(3) for information on which those are) from a given URI. The general method to
do so is to "open" the URI using **OSSL_STORE_open()**, read each available and supported object
using **OSSL_STORE_load()** as long as **OSSL_STORE_eof()** hasn't been reached, and finish it off with
OSSL_STORE_close().

The retrieved information is stored in a **OSSL_STORE_INFO**, which is further described in **OSSL_STORE_INFO**(3).

Types

OSSL_STORE_CTX is a context variable that holds all the internal information for **OSSL_STORE_open**(), **OSSL_STORE_open_ex**(), **OSSL_STORE_load**(), **OSSL_STORE_eof**() and **OSSL_STORE_close**() to work together.

Functions

OSSL_STORE_open_ex() takes a uri or path *uri*, password UI method *ui_method* with associated data *ui_data*, and post processing callback *post_process* with associated data *post_process_data*, a library context *libctx* with an associated property query *propq*, and opens a channel to the data located at the URI and returns a **OSSL_STORE_CTX** with all necessary internal information. The given *ui_method* and *ui_data* will be reused by all functions that use **OSSL_STORE_CTX** when interaction is needed, for instance to provide a password. The auxiliary **OSSL_PARAM**(3) parameters in *params* can be set to further modify the store operation. The given *post_process* and *post_process_data* will be reused by **OSSL_STORE_load**() to manipulate or drop the value to be returned. The *post_process* function drops values by returning NULL, which will cause **OSSL_STORE_load**() to start its process over with loading the next object, until *post_process* returns something other than NULL, or the end of data is reached as indicated by **OSSL_STORE_eof**().

OSSL_STORE_open() is similar to **OSSL_STORE_open_ex**() but uses NULL for the *params*, the library context *libctx* and property query *propq*.

OSSL_STORE_ctrl() takes a **OSSL_STORE_CTX**, and command number *cmd* and more arguments not specified here. The available loader specific command numbers and arguments they each take depends on the loader that's used and is documented together with that loader.

There are also global controls available:

OSSL_STORE_C_USE_SECMEM

> Controls if the loader should attempt to use secure memory for any allocated **OSSL_STORE_INFO** and its contents. This control expects one argument, a pointer to an *int* that is expected to have the value 1 (yes) or 0 (no). Any other value is an error.

OSSL_STORE_load() takes a **OSSL_STORE_CTX** and tries to load the next available object and return it wrapped with **OSSL_STORE_INFO**.

OSSL_STORE_eof() takes a **OSSL_STORE_CTX** and checks if we've reached the end of data.

OSSL_STORE_error() takes a **OSSL_STORE_CTX** and checks if an error occurred in the last **OSSL_STORE_load()** call. Note that it may still be meaningful to try and load more objects, unless **OSSL_STORE_eof()** shows that the end of data has been reached.

OSSL_STORE_close() takes a **OSSL_STORE_CTX**, closes the channel that was opened by **OSSL_STORE_open()** and frees all other information that was stored in the **OSSL_STORE_CTX**, as well as the **OSSL_STORE_CTX** itself. If *ctx* is NULL it does nothing.

NOTES

A string without a scheme prefix (that is, a non-URI string) is implicitly interpreted as using the *file:* scheme.

There are some tools that can be used together with **OSSL_STORE_open()** to determine if any failure is caused by an unparsable URI, or if it's a different error (such as memory allocation failures); if the URI was parsable but the scheme unregistered, the top error will have the reason "OSSL_STORE_R_UNREGISTERED_SCHEME".

These functions make no direct assumption regarding the pass phrase received from the password callback. The loaders may make assumptions, however. For example, the **file:** scheme loader inherits the assumptions made by OpenSSL functionality that handles the different file types; this is mostly relevant for PKCS#12 objects. See **passphrase-encoding**(7) for further information.

RETURN VALUES

OSSL_STORE_open() returns a pointer to a **OSSL_STORE_CTX** on success, or NULL on failure.

OSSL_STORE_load() returns a pointer to a **OSSL_STORE_INFO** on success, or NULL on error or when end of data is reached. Use **OSSL_STORE_error()** and **OSSL_STORE_eof()** to determine the meaning of a returned NULL.

OSSL_STORE_eof() returns 1 if the end of data has been reached or an error occurred, 0 otherwise.

OSSL_STORE_error() returns 1 if an error occurred in an **OSSL_STORE_load()** call, otherwise 0.

OSSL_STORE_ctrl() and **OSSL_STORE_close()** returns 1 on success, or 0 on failure.

SEE ALSO

ossl_store(7), **OSSL_STORE_INFO**(3), **OSSL_STORE_register_loader**(3), **passphrase-encoding**(7)

HISTORY

OSSL_STORE_open_ex() was added in OpenSSL 3.0.

OSSL_STORE_CTX, **OSSL_STORE_post_process_info_fn()**, **OSSL_STORE_open()**, **OSSL_STORE_ctrl()**, **OSSL_STORE_load()**, **OSSL_STORE_eof()** and **OSSL_STORE_close()** were added in OpenSSL 1.1.1.

Handling of NULL *ctx* argument for **OSSL_STORE_close()** was introduced in OpenSSL 1.1.1h.

OSSL_STORE_open_ex() was added in OpenSSL 3.0.

OSSL_STORE_ctrl() and **OSSL_STORE_vctrl()** were deprecated in OpenSSL 3.0.

COPYRIGHT