

**NAME**

OSSL\_STORE\_LOADER, OSSL\_STORE\_LOADER\_fetch, OSSL\_STORE\_LOADER\_up\_ref, OSSL\_STORE\_LOADER\_free, OSSL\_STORE\_LOADER\_get0\_provider, OSSL\_STORE\_LOADER\_get0\_properties, OSSL\_STORE\_LOADER\_is\_a, OSSL\_STORE\_LOADER\_get0\_description, OSSL\_STORE\_LOADER\_do\_all\_provided, OSSL\_STORE\_LOADER\_names\_do\_all, OSSL\_STORE\_LOADER\_CTX, OSSL\_STORE\_LOADER\_new, OSSL\_STORE\_LOADER\_get0\_engine, OSSL\_STORE\_LOADER\_get0\_scheme, OSSL\_STORE\_LOADER\_set\_open, OSSL\_STORE\_LOADER\_set\_open\_ex, OSSL\_STORE\_LOADER\_set\_attach, OSSL\_STORE\_LOADER\_set\_ctrl, OSSL\_STORE\_LOADER\_set\_expect, OSSL\_STORE\_LOADER\_set\_find, OSSL\_STORE\_LOADER\_set\_load, OSSL\_STORE\_LOADER\_set\_eof, OSSL\_STORE\_LOADER\_set\_error, OSSL\_STORE\_LOADER\_set\_close, OSSL\_STORE\_register\_loader, OSSL\_STORE\_unregister\_loader, OSSL\_STORE\_open\_fn, OSSL\_STORE\_open\_ex\_fn, OSSL\_STORE\_attach\_fn, OSSL\_STORE\_ctrl\_fn, OSSL\_STORE\_expect\_fn, OSSL\_STORE\_find\_fn, OSSL\_STORE\_load\_fn, OSSL\_STORE\_eof\_fn, OSSL\_STORE\_error\_fn, OSSL\_STORE\_close\_fn - Types and functions to manipulate, register and unregister STORE loaders for different URI schemes

**SYNOPSIS**

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

```
typedef struct ossl_store_loader_st OSSL_STORE_LOADER;
```

```
OSSL_STORE_LOADER *OSSL_STORE_LOADER_fetch(OSSL_LIB_CTX *libctx,
      const char *scheme,
      const char *properties);
int OSSL_STORE_LOADER_up_ref(OSSL_STORE_LOADER *loader);
void OSSL_STORE_LOADER_free(OSSL_STORE_LOADER *loader);
const OSSL_PROVIDER *OSSL_STORE_LOADER_get0_provider(const OSSL_STORE_LOADER *
      loader);
const char *OSSL_STORE_LOADER_get0_properties(const OSSL_STORE_LOADER *loader);
const char *OSSL_STORE_LOADER_get0_description(const OSSL_STORE_LOADER *loader);
int OSSL_STORE_LOADER_is_a(const OSSL_STORE_LOADER *loader,
      const char *scheme);
void OSSL_STORE_LOADER_do_all_provided(OSSL_LIB_CTX *libctx,
      void (*user_fn)(OSSL_STORE_LOADER *loader,
      void *arg),
      void *user_arg);
int OSSL_STORE_LOADER_names_do_all(const OSSL_STORE_LOADER *loader,
      void (*fn)(const char *name, void *data),
```

```
void *data);
```

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)**:

```
OSSL_STORE_LOADER *OSSL_STORE_LOADER_new(ENGINE *e, const char *scheme);
const ENGINE *OSSL_STORE_LOADER_get0_engine(const OSSL_STORE_LOADER
      *store_loader);
const char *OSSL_STORE_LOADER_get0_scheme(const OSSL_STORE_LOADER
      *store_loader);
```

```
/* struct ossl_store_loader_ctx_st is defined differently by each loader */
typedef struct ossl_store_loader_ctx_st OSSL_STORE_LOADER_CTX;
```

```
typedef OSSL_STORE_LOADER_CTX *(*OSSL_STORE_open_fn)(
    const char *uri, const UI_METHOD *ui_method, void *ui_data);
int OSSL_STORE_LOADER_set_open(OSSL_STORE_LOADER *store_loader,
    OSSL_STORE_open_fn store_open_function);
typedef OSSL_STORE_LOADER_CTX *(*OSSL_STORE_open_ex_fn)(
    const char *uri, const UI_METHOD *ui_method, void *ui_data);
int OSSL_STORE_LOADER_set_open_ex
(OSSL_STORE_LOADER *store_loader,
    OSSL_STORE_open_ex_fn store_open_ex_function);
typedef OSSL_STORE_LOADER_CTX *(*OSSL_STORE_attach_fn)
(const OSSL_STORE_LOADER *loader, BIO *bio,
    OSSL_LIB_CTX *libctx, const char *propq,
    const UI_METHOD *ui_method, void *ui_data);
int OSSL_STORE_LOADER_set_attach(OSSL_STORE_LOADER *loader,
    OSSL_STORE_attach_fn attach_function);
typedef int (*OSSL_STORE_ctrl_fn)(OSSL_STORE_LOADER_CTX *ctx, int cmd,
    va_list args);
int OSSL_STORE_LOADER_set_ctrl(OSSL_STORE_LOADER *store_loader,
    OSSL_STORE_ctrl_fn store_ctrl_function);
typedef int (*OSSL_STORE_expect_fn)(OSSL_STORE_LOADER_CTX *ctx, int expected);
int OSSL_STORE_LOADER_set_expect(OSSL_STORE_LOADER *loader,
    OSSL_STORE_expect_fn expect_function);
typedef int (*OSSL_STORE_find_fn)(OSSL_STORE_LOADER_CTX *ctx,
    OSSL_STORE_SEARCH *criteria);
int OSSL_STORE_LOADER_set_find(OSSL_STORE_LOADER *loader,
    OSSL_STORE_find_fn find_function);
```

```

typedef OSSL_STORE_INFO *(*OSSL_STORE_load_fn)(OSSL_STORE_LOADER_CTX *ctx,
        UI_METHOD *ui_method,
        void *ui_data);
int OSSL_STORE_LOADER_set_load(OSSL_STORE_LOADER *store_loader,
        OSSL_STORE_load_fn store_load_function);
typedef int (*OSSL_STORE_eof_fn)(OSSL_STORE_LOADER_CTX *ctx);
int OSSL_STORE_LOADER_set_eof(OSSL_STORE_LOADER *store_loader,
        OSSL_STORE_eof_fn store_eof_function);
typedef int (*OSSL_STORE_error_fn)(OSSL_STORE_LOADER_CTX *ctx);
int OSSL_STORE_LOADER_set_error(OSSL_STORE_LOADER *store_loader,
        OSSL_STORE_error_fn store_error_function);
typedef int (*OSSL_STORE_close_fn)(OSSL_STORE_LOADER_CTX *ctx);
int OSSL_STORE_LOADER_set_close(OSSL_STORE_LOADER *store_loader,
        OSSL_STORE_close_fn store_close_function);
void OSSL_STORE_LOADER_free(OSSL_STORE_LOADER *store_loader);

int OSSL_STORE_register_loader(OSSL_STORE_LOADER *loader);
OSSL_STORE_LOADER *OSSL_STORE_unregister_loader(const char *scheme);

```

## DESCRIPTION

**OSSL\_STORE\_LOADER** is a method for OSSL\_STORE loaders, which implement **OSSL\_STORE\_open()**, **OSSL\_STORE\_open\_ex()**, **OSSL\_STORE\_load()**, **OSSL\_STORE\_eof()**, **OSSL\_STORE\_error()** and **OSSL\_STORE\_close()** for specific storage schemes.

**OSSL\_STORE\_LOADER\_fetch()** looks for an implementation for a storage *scheme* within the providers that has been loaded into the **OSSL\_LIB\_CTX** given by *libctx*, and with the properties given by *properties*.

**OSSL\_STORE\_LOADER\_up\_ref()** increments the reference count for the given *loader*.

**OSSL\_STORE\_LOADER\_free()** decrements the reference count for the given *loader*, and when the count reaches zero, frees it.

**OSSL\_STORE\_LOADER\_get0\_provider()** returns the provider of the given *loader*.

**OSSL\_STORE\_LOADER\_get0\_properties()** returns the property definition associated with the given *loader*.

**OSSL\_STORE\_LOADER\_is\_a()** checks if *loader* is an implementation of an algorithm that's identifiable with *scheme*.

**OSSL\_STORE\_LOADER\_get0\_description()** returns a description of the *loader*, meant for display and human consumption. The description is at the discretion of the *loader* implementation.

**OSSL\_STORE\_LOADER\_do\_all\_provided()** traverses all store implementations by all activated providers in the library context *libctx*, and for each of the implementations, calls *user\_fn* with the implementation method and *user\_arg* as arguments.

**OSSL\_STORE\_LOADER\_names\_do\_all()** traverses all names for the given *loader*, and calls *fn* with each name and *data*.

### Legacy Types and Functions (deprecated)

These functions help applications and engines to create loaders for schemes they support. These are all deprecated and discouraged in favour of provider implementations, see **provider-storemgmt(7)**.

**OSSL\_STORE\_LOADER\_CTX** is a type template, to be defined by each loader using "struct `ossl_store_loader_ctx_st { ... }`".

**OSSL\_STORE\_open\_fn**, **OSSL\_STORE\_open\_ex\_fn**, **OSSL\_STORE\_ctrl\_fn**, **OSSL\_STORE\_expect\_fn**, **OSSL\_STORE\_find\_fn**, **OSSL\_STORE\_load\_fn**, **OSSL\_STORE\_eof\_fn**, and **OSSL\_STORE\_close\_fn** are the function pointer types used within a STORE loader. The functions pointed at define the functionality of the given loader.

### **OSSL\_STORE\_open\_fn** and **OSSL\_STORE\_open\_ex\_fn**

**OSSL\_STORE\_open\_ex\_fn** takes a URI and is expected to interpret it in the best manner possible according to the scheme the loader implements. It also takes a **UI\_METHOD** and associated data, to be used any time something needs to be prompted for, as well as a library context *libctx* with an associated property query *propq*, to be used when fetching necessary algorithms to perform the loads. Furthermore, this function is expected to initialize what needs to be initialized, to create a private data store (**OSSL\_STORE\_LOADER\_CTX**, see above), and to return it. If something goes wrong, this function is expected to return NULL.

**OSSL\_STORE\_open\_fn** does the same thing as **OSSL\_STORE\_open\_ex\_fn** but uses NULL for the library context *libctx* and property query *propq*.

### **OSSL\_STORE\_attach\_fn**

This function takes a **BIO**, otherwise works like **OSSL\_STORE\_open\_ex\_fn**.

### **OSSL\_STORE\_ctrl\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer, a command number *cmd* and a **va\_list** *args* and is used to manipulate loader specific parameters.

Loader specific command numbers must begin at **OSSL\_STORE\_C\_CUSTOM\_START**. Any number below that is reserved for future globally known command numbers.

This function is expected to return 1 on success, 0 on error.

### **OSSL\_STORE\_expect\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and a **OSSL\_STORE\_INFO** identity *expected*, and is used to tell the loader what object type is expected. *expected* may be zero to signify that no specific object type is expected.

This function is expected to return 1 on success, 0 on error.

### **OSSL\_STORE\_find\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and a **OSSL\_STORE\_SEARCH** search criterion, and is used to tell the loader what to search for.

When called with the loader context being NULL, this function is expected to return 1 if the loader supports the criterion, otherwise 0.

When called with the loader context being something other than NULL, this function is expected to return 1 on success, 0 on error.

### **OSSL\_STORE\_load\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and a **UI\_METHOD** with associated data. It's expected to load the next available data, mold it into a data structure that can be wrapped in a **OSSL\_STORE\_INFO** using one of the **OSSL\_STORE\_INFO(3)** functions. If no more data is available or an error occurs, this function is expected to return NULL. The **OSSL\_STORE\_eof\_fn** and **OSSL\_STORE\_error\_fn** functions must indicate if it was in fact the end of data or if an error occurred.

Note that this function retrieves *one* data item only.

### **OSSL\_STORE\_eof\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and is expected to return 1 to indicate that the end of available data has been reached. It is otherwise expected to return 0.

### **OSSL\_STORE\_error\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and is expected to return 1 to indicate that an error occurred in a previous call to the **OSSL\_STORE\_load\_fn** function. It is otherwise expected to return 0.

**OSSL\_STORE\_close\_fn**

This function takes a **OSSL\_STORE\_LOADER\_CTX** pointer and is expected to close or shut down what needs to be closed, and finally free the contents of the **OSSL\_STORE\_LOADER\_CTX** pointer. It returns 1 on success and 0 on error.

**OSSL\_STORE\_LOADER\_new()** creates a new **OSSL\_STORE\_LOADER**. It takes an **ENGINE** *e* and a string *scheme*. *scheme* must *always* be set. Both *e* and *scheme* are used as is and must therefore be alive as long as the created loader is.

**OSSL\_STORE\_LOADER\_get0\_engine()** returns the engine of the *store\_loader*.

**OSSL\_STORE\_LOADER\_get0\_scheme()** returns the scheme of the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_opener()** sets the opener function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_opener\_ex()** sets the opener with library context function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_attach()** sets the attacher function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_ctrl()** sets the control function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_expect()** sets the expect function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_load()** sets the loader function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_eof()** sets the end of file checker function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_set\_close()** sets the closing function for the *store\_loader*.

**OSSL\_STORE\_LOADER\_free()** frees the given *store\_loader*.

**OSSL\_STORE\_register\_loader()** register the given *store\_loader* and thereby makes it available for use with **OSSL\_STORE\_open()**, **OSSL\_STORE\_open\_ex()**, **OSSL\_STORE\_load()**, **OSSL\_STORE\_eof()** and **OSSL\_STORE\_close()**.

**OSSL\_STORE\_unregister\_loader()** unregister the store loader for the given *scheme*.

**RETURN VALUES**

**OSSL\_STORE\_LOADER\_fetch()** returns a pointer to an **OSSL\_STORE\_LOADER** object, or **NULL** on error.

**OSSL\_STORE\_LOADER\_up\_ref()** returns 1 on success, or 0 on error.

**OSSL\_STORE\_LOADER\_names\_do\_all()** returns 1 if the callback was called for all names. A return value of 0 means that the callback was not called for any names.

**OSSL\_STORE\_LOADER\_free()** doesn't return any value.

**OSSL\_STORE\_LOADER\_get0\_provider()** returns a pointer to a provider object, or NULL on error.

**OSSL\_STORE\_LOADER\_get0\_properties()** returns a pointer to a property definition string, or NULL on error.

**OSSL\_STORE\_LOADER\_is\_a()** returns 1 if *loader* was identifiable, otherwise 0.

**OSSL\_STORE\_LOADER\_get0\_description()** returns a pointer to a description, or NULL if there isn't one.

The functions with the types **OSSL\_STORE\_open\_fn**, **OSSL\_STORE\_open\_ex\_fn**, **OSSL\_STORE\_ctrl\_fn**, **OSSL\_STORE\_expect\_fn**, **OSSL\_STORE\_load\_fn**, **OSSL\_STORE\_eof\_fn** and **OSSL\_STORE\_close\_fn** have the same return values as **OSSL\_STORE\_open()**, **OSSL\_STORE\_open\_ex()**, **OSSL\_STORE\_ctrl()**, **OSSL\_STORE\_expect()**, **OSSL\_STORE\_load()**, **OSSL\_STORE\_eof()** and **OSSL\_STORE\_close()**, respectively.

**OSSL\_STORE\_LOADER\_new()** returns a pointer to a **OSSL\_STORE\_LOADER** on success, or NULL on failure.

**OSSL\_STORE\_LOADER\_set\_open()**, **OSSL\_STORE\_LOADER\_set\_open\_ex()**, **OSSL\_STORE\_LOADER\_set\_ctrl()**, **OSSL\_STORE\_LOADER\_set\_load()**, **OSSL\_STORE\_LOADER\_set\_eof()** and **OSSL\_STORE\_LOADER\_set\_close()** return 1 on success, or 0 on failure.

**OSSL\_STORE\_register\_loader()** returns 1 on success, or 0 on failure.

**OSSL\_STORE\_unregister\_loader()** returns the unregistered loader on success, or NULL on failure.

## SEE ALSO

**ossl\_store(7)**, **OSSL\_STORE\_open(3)**, **OSSL\_LIB\_CTX(3)**, **provider-storemgmt(7)**

## HISTORY

**OSSL\_STORE\_LOADER\_fetch()**, **OSSL\_STORE\_LOADER\_up\_ref()**,

**OSSL\_STORE\_LOADER\_free()**, **OSSL\_STORE\_LOADER\_get0\_provider()**, **OSSL\_STORE\_LOADER\_get0\_properties()**, **OSSL\_STORE\_LOADER\_is\_a()**, **OSSL\_STORE\_LOADER\_do\_all\_provided()** and **OSSL\_STORE\_LOADER\_names\_do\_all()** were added in OpenSSL 3.0.

**OSSL\_STORE\_open\_ex\_fn()** was added in OpenSSL 3.0.

**OSSL\_STORE\_LOADER**, **OSSL\_STORE\_LOADER\_CTX**, **OSSL\_STORE\_LOADER\_new()**, **OSSL\_STORE\_LOADER\_set0\_scheme()**, **OSSL\_STORE\_LOADER\_get0\_scheme()**, **OSSL\_STORE\_LOADER\_get0\_engine()**, **OSSL\_STORE\_LOADER\_set\_expect()**, **OSSL\_STORE\_LOADER\_set\_find()**, **OSSL\_STORE\_LOADER\_set\_attach()**, **OSSL\_STORE\_LOADER\_set\_open\_ex()**, **OSSL\_STORE\_LOADER\_set\_open()**, **OSSL\_STORE\_LOADER\_set\_ctrl()**, **OSSL\_STORE\_LOADER\_set\_load()**, **OSSL\_STORE\_LOADER\_set\_eof()**, **OSSL\_STORE\_LOADER\_set\_close()**, **OSSL\_STORE\_LOADER\_free()**, **OSSL\_STORE\_register\_loader()**, **OSSL\_STORE\_LOADER\_set\_error()**, **OSSL\_STORE\_unregister\_loader()**, **OSSL\_STORE\_open\_fn()**, **OSSL\_STORE\_ctrl\_fn()**, **OSSL\_STORE\_load\_fn()**, **OSSL\_STORE\_eof\_fn()** and **OSSL\_STORE\_close\_fn()** were added in OpenSSL 1.1.1, and became deprecated in OpenSSL 3.0.

## **COPYRIGHT**

Copyright 2016-2023 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.