

**NAME**

X509\_STORE, X509\_STORE\_add\_cert, X509\_STORE\_add\_crl, X509\_STORE\_set\_depth, X509\_STORE\_set\_flags, X509\_STORE\_set\_purpose, X509\_STORE\_set\_trust, X509\_STORE\_add\_lookup, X509\_STORE\_load\_file\_ex, X509\_STORE\_load\_file, X509\_STORE\_load\_path, X509\_STORE\_load\_store\_ex, X509\_STORE\_load\_store, X509\_STORE\_set\_default\_paths\_ex, X509\_STORE\_set\_default\_paths, X509\_STORE\_load\_locations\_ex, X509\_STORE\_load\_locations - X509\_STORE manipulation

**SYNOPSIS**

```
#include <openssl/x509_vfy.h>
```

```
typedef x509_store_st X509_STORE;
```

```
int X509_STORE_add_cert(X509_STORE *ctx, X509 *x);
int X509_STORE_add_crl(X509_STORE *ctx, X509_CRL *x);
int X509_STORE_set_depth(X509_STORE *store, int depth);
int X509_STORE_set_flags(X509_STORE *ctx, unsigned long flags);
int X509_STORE_set_purpose(X509_STORE *ctx, int purpose);
int X509_STORE_set_trust(X509_STORE *ctx, int trust);
```

```
X509_LOOKUP *X509_STORE_add_lookup(X509_STORE *store,
                                   X509_LOOKUP_METHOD *meth);
```

```
int X509_STORE_set_default_paths_ex(X509_STORE *ctx, OSSL_LIB_CTX *libctx,
                                   const char *propq);
```

```
int X509_STORE_set_default_paths(X509_STORE *ctx);
```

```
int X509_STORE_load_file_ex(X509_STORE *ctx, const char *file,
                           OSSL_LIB_CTX *libctx, const char *propq);
```

```
int X509_STORE_load_file(X509_STORE *ctx, const char *file);
```

```
int X509_STORE_load_path(X509_STORE *ctx, const char *dir);
```

```
int X509_STORE_load_store_ex(X509_STORE *ctx, const char *uri,
                             OSSL_LIB_CTX *libctx, const char *propq);
```

```
int X509_STORE_load_store(X509_STORE *ctx, const char *uri);
```

```
int X509_STORE_load_locations_ex(X509_STORE *ctx, const char *file,
                                 const char *dir, OSSL_LIB_CTX *libctx,
                                 const char *propq);
```

```
int X509_STORE_load_locations(X509_STORE *ctx,
                              const char *file, const char *dir);
```

**DESCRIPTION**

The **X509\_STORE** structure is intended to be a consolidated mechanism for holding information about X.509 certificates and CRLs, and constructing and validating chains of certificates terminating in trusted roots. It admits multiple lookup mechanisms and efficient scaling performance with large numbers of certificates, and a great deal of flexibility in how validation and policy checks are performed.

Details of the chain building and checking process are described in "Certification Path Building" in **openssl-verification-options(1)** and "Certification Path Validation" in **openssl-verification-options(1)**.

**X509\_STORE\_new(3)** creates an empty **X509\_STORE** structure, which contains no information about trusted certificates or where such certificates are located on disk, and is generally not usable. Normally, trusted certificates will be added to the **X509\_STORE** to prepare it for use, via mechanisms such as **X509\_STORE\_add\_lookup()** and **X509\_LOOKUP\_file()**, or **PEM\_read\_bio\_X509\_AUX()** and **X509\_STORE\_add\_cert()**. CRLs can also be added, and many behaviors configured as desired.

Once the **X509\_STORE** is suitably configured, **X509\_STORE\_CTX\_new()** is used to instantiate a single-use **X509\_STORE\_CTX** for each chain-building and verification operation. That process includes providing the end-entity certificate to be verified and an additional set of untrusted certificates that may be used in chain-building. As such, it is expected that the certificates included in the **X509\_STORE** are certificates that represent trusted entities such as root certificate authorities (CAs). OpenSSL represents these trusted certificates internally as **X509** objects with an associated **X509\_CERT\_AUX**, as are produced by **PEM\_read\_bio\_X509\_AUX()** and similar routines that refer to **X509\_AUX**. The public interfaces that operate on such trusted certificates still operate on pointers to **X509** objects, though.

**X509\_STORE\_add\_cert()** and **X509\_STORE\_add\_crl()** add the respective object to the **X509\_STORE**'s local storage. Untrusted objects should not be added in this way. The added object's reference count is incremented by one, hence the caller retains ownership of the object and needs to free it when it is no longer needed.

**X509\_STORE\_set\_depth()**, **X509\_STORE\_set\_flags()**, **X509\_STORE\_set\_purpose()**, **X509\_STORE\_set\_trust()**, and **X509\_STORE\_set1\_param()** set the default values for the corresponding values used in certificate chain validation. Their behavior is documented in the corresponding **X509\_VERIFY\_PARAM** manual pages, e.g., **X509\_VERIFY\_PARAM\_set\_depth(3)**.

**X509\_STORE\_add\_lookup()** finds or creates a **X509\_LOOKUP(3)** with the **X509\_LOOKUP\_METHOD(3)** *meth* and adds it to the **X509\_STORE** *store*. This also associates the **X509\_STORE** with the lookup, so **X509\_LOOKUP** functions can look up objects in that store.

**X509\_STORE\_load\_file\_ex()** loads trusted certificate(s) into an **X509\_STORE** from a given file. The

library context *libctx* and property query *propq* are used when fetching algorithms from providers.

**X509\_STORE\_load\_file()** is similar to **X509\_STORE\_load\_file\_ex()** but uses NULL for the library context *libctx* and property query *propq*.

**X509\_STORE\_load\_path()** loads trusted certificate(s) into an **X509\_STORE** from a given directory path. The certificates in the directory must be in hashed form, as documented in **X509\_LOOKUP\_hash\_dir(3)**.

**X509\_STORE\_load\_store\_ex()** loads trusted certificate(s) into an **X509\_STORE** from a store at a given URI. The library context *libctx* and property query *propq* are used when fetching algorithms from providers.

**X509\_STORE\_load\_store()** is similar to **X509\_STORE\_load\_store\_ex()** but uses NULL for the library context *libctx* and property query *propq*.

**X509\_STORE\_load\_locations\_ex()** combines **X509\_STORE\_load\_file\_ex()** and **X509\_STORE\_load\_path()** for a given file and/or directory path. It is permitted to specify just a file, just a directory, or both paths.

**X509\_STORE\_load\_locations()** is similar to **X509\_STORE\_load\_locations\_ex()** but uses NULL for the library context *libctx* and property query *propq*.

**X509\_STORE\_set\_default\_paths\_ex()** is somewhat misnamed, in that it does not set what default paths should be used for loading certificates. Instead, it loads certificates into the **X509\_STORE** from the hardcoded default paths. The library context *libctx* and property query *propq* are used when fetching algorithms from providers.

**X509\_STORE\_set\_default\_paths()** is similar to **X509\_STORE\_set\_default\_paths\_ex()** but uses NULL for the library context *libctx* and property query *propq*.

## RETURN VALUES

**X509\_STORE\_add\_cert()**, **X509\_STORE\_add\_crl()**, **X509\_STORE\_set\_depth()**, **X509\_STORE\_set\_flags()**, **X509\_STORE\_set\_purpose()**, **X509\_STORE\_set\_trust()**, **X509\_STORE\_load\_file\_ex()**, **X509\_STORE\_load\_file()**, **X509\_STORE\_load\_path()**, **X509\_STORE\_load\_store\_ex()**, **X509\_STORE\_load\_store()**, **X509\_STORE\_load\_locations\_ex()**, **X509\_STORE\_load\_locations()**, **X509\_STORE\_set\_default\_paths\_ex()** and **X509\_STORE\_set\_default\_paths()** return 1 on success or 0 on failure.

**X509\_STORE\_add\_lookup()** returns the found or created **X509\_LOOKUP(3)**, or NULL on error.

**SEE ALSO**

**X509\_LOOKUP\_hash\_dir(3)**. **X509\_VERIFY\_PARAM\_set\_depth(3)**. **X509\_STORE\_new(3)**,  
**X509\_STORE\_get0\_param(3)**

**HISTORY**

The functions **X509\_STORE\_set\_default\_paths\_ex()**, **X509\_STORE\_load\_file\_ex()**,  
**X509\_STORE\_load\_store\_ex()** and **X509\_STORE\_load\_locations\_ex()** were added in OpenSSL 3.0.

**COPYRIGHT**

Copyright 2017-2022 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>.