NAME

X509_STORE_CTX_new, X509_STORE_CTX_cleanup, X509_STORE_CTX_free, X509_STORE_CTX_init, X509_STORE_CTX_set0_trusted_stack, X509_STORE_CTX_set_cert, X509_STORE_CTX_set0_crls, X509_STORE_CTX_get0_chain, X509_STORE_CTX_set0_verified_chain, X509_STORE_CTX_get0_param, X509_STORE_CTX_set0_param, X509_STORE_CTX_get0_untrusted, X509_STORE_CTX_set0_untrusted, X509_STORE_CTX_get_num_untrusted, X509_STORE_CTX_set_default, X509_STORE_CTX_set_verify, X509_STORE_CTX_verify_fn - X509_STORE_CTX initialisation

SYNOPSIS

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

X509_STORE_CTX *X509_STORE_CTX_new(void);
void X509_STORE_CTX_cleanup(X509_STORE_CTX *ctx);
void X509_STORE_CTX_free(X509_STORE_CTX *ctx);

int X509_STORE_CTX_init(X509_STORE_CTX *ctx, X509_STORE *store,
X509 *x509, STACK_OF(X509) *chain);

void X509_STORE_CTX_set0_trusted_stack(X509_STORE_CTX *ctx, STACK_OF(X509) *sk);

void X509_STORE_CTX_set_cert(X509_STORE_CTX *ctx, X509 *x);
STACK_OF(X509) *X509_STORE_CTX_get0_chain(X609_STORE_CTX *ctx);
void X509_STORE_CTX_set0_verified_chain(X509_STORE_CTX *ctx, STACK_OF(X509) *cha
void X509_STORE_CTX_set0_crls(X509_STORE_CTX *ctx, STACK_OF(X509_CRL) *sk);

X509_VERIFY_PARAM *X509_STORE_CTX_get0_param(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set0_param(X509_STORE_CTX *ctx, X509_VERIFY_PARAM *param);
int X509_STORE_CTX_set_default(X509_STORE_CTX *ctx, const char *name);

STACK_OF(X509)* X509_STORE_CTX_get0_untrusted(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set0_untrusted(X509_STORE_CTX *ctx, STACK_OF(X509) *sk);

int X509_STORE_CTX_get_num_untrusted(X509_STORE_CTX *ctx);

typedef int (*X509_STORE_CTX_verify_fn)(X509_STORE_CTX *);
void X509_STORE_CTX_set_verify(X509_STORE_CTX *ctx, X509_STORE_CTX_verify_fn ver
```

DESCRIPTION

These functions initialise an **X509_STORE_CTX** structure for subsequent use by *X509_verify_cert()*.

X509_STORE_CTX_new() returns a newly initialised **X509_STORE_CTX** structure.

X509_STORE_CTX_cleanup() internally cleans up an **X509_STORE_CTX** structure. The context can then be reused with an new call to *X509_STORE_CTX_init()*.

X509_STORE_CTX_free() completely frees up **ctx**. After this call **ctx** is no longer valid. If**ctx** is NULL nothing is done.

X509_STORE_CTX_init() sets up **ctx** for a subsequent verification operation. It must be called before each call to *X509_verify_cert()*, i.e. a **ctx** is only good for one call to *X509_verify_cert()*; if you want to verify a second certificate with the same **ctx** then you must call *X509_STORE_CTX_cleanup()* and then *X509_STORE_CTX_init()* again before the second call to *X509_verify_cert()*. The trusted certificate store is set to **store**, the end entity certificate to be verified is set to **x509** and a set of additional certificates (which will be untrusted but may be used to build the chain) in **chain**. Any or all of the **store**, **x509** and **chain** parameters can be **NULL**.

X509_STORE_CTX_set0_trusted_stack() sets the set of trusted certificates of **ctx** to **sk**. This is an

alternative way of specifying trusted certificates instead of using an **X509_STORE**.

X509_STORE_CTX_set_cert() sets the certificate to be verified in **ctx** to **x**.

X509_STORE_CTX_set0_verified_chain() sets the validated chain used by **ctx** to be **chain**. Ownership of the chain is transferred to **ctx** and should not be free'd by the caller. *X509_STORE_CTX_get0_chain()* returns a the internal pointer used by the **ctx** that contains the validated chain.

X509_STORE_CTX_set0_crls() sets a set of CRLs to use to aid certificate verification to **sk**. These CRLs will only be used if CRL verification is enabled in the associated **X509_VERIFY_PARAM** structure. This might be used where additional ''useful'' CRLs are supplied as part of a protocol, for example in a PKCS#7 structure.

X509_STORE_CTX_get0_param() retrieves an internal pointer to the verification parameters associated with **ctx**.

X509_STORE_CTX_get0_untrusted() retrieves an internal pointer to the stack of untrusted certificates associated with **ctx**.

X509_STORE_CTX_set0_untrusted() sets the internal point to the stack of untrusted certificates associated with **ctx** to **sk**.

X509_STORE_CTX_set0_param() sets the internal verification parameter pointer to **param**. After this call **param** should not be used.

X509_STORE_CTX_set_default() looks up and sets the default verification method to **name**. This uses the function *X509_VERIFY_PARAM_lookup()* to find an appropriate set of parameters from **name**.

X509_STORE_CTX_get_num_untrusted() returns the number of untrusted certificates that were used in building the chain following a call to *X509_verify_cert()*.

X509_STORE_CTX_set_verify() provides the capability for overriding the default verify function. This function is responsible for verifying chain signatures and expiration times.

A verify function is defined as an X509_STORE_CTX_verify type which has the following signature:

```
int (*verify)(X509_STORE_CTX *);
```

This function should receive the current X509_STORE_CTX as a parameter and return 1 on success or 0 on failure.

NOTES

The certificates and CRLs in a store are used internally and should **not** be freed up until after the associated **X509_STORE_CTX** is freed.

BUGS

The certificates and CRLs in a context are used internally and should **not** be freed up until after the associated **X509_STORE_CTX** is freed. Copies should be made or reference counts increased instead.

RETURN VALUES

X509_STORE_CTX_new() returns an newly allocates context or **NULL** is an error occurred.

X509_STORE_CTX_init() returns 1 for success or 0 if an error occurred.

X509_STORE_CTX_get0_param() returns a pointer to an **X509_VERIFY_PARAM** structure or **NULL** if an error occurred.

X509_STORE_CTX_cleanup(), *X509_STORE_CTX_free()*, *X509_STORE_CTX_set0_trusted_stack()*, *X509_STORE_CTX_set_cert()*, *X509_STORE_CTX_set0_crls()* and *X509_STORE_CTX_set0_param()* do not return values.

X509_STORE_CTX_set_default() returns 1 for success or 0 if an error occurred.

X509_STORE_CTX_get_num_untrusted() returns the number of untrusted certificates used.

SEE ALSO

X509_verify_cert(3) *X509_VERIFY_PARAM_set_flags(3)*

HISTORY

X509_STORE_CTX_set0_crls() was first added to OpenSSL 1.0.0
X509_STORE_CTX_get_num_untrusted() was first added to OpenSSL 1.1.0

COPYRIGHT