NAME

X509_STORE_CTX_get_error, X509_STORE_CTX_set_error, X509_STORE_CTX_get_error_depth, X509_STORE_CTX_set_error_depth, X509_STORE_CTX_get_current_cert, X509_STORE_CTX_set_current_cert, X509_STORE_CTX_get0_cert, X509_STORE_CTX_get1_chain, X509_verify_cert_error_string - get or set certificate verification status information

SYNOPSIS

```
#include <openssl/x509.h>

int X509_STORE_CTX_get_error(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set_error(X509_STORE_CTX *ctx, int s);
int X509_STORE_CTX_get_error_depth(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set_error_depth(X509_STORE_CTX *ctx, int depth);
X509 *X509_STORE_CTX_get_current_cert(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set_current_cert(X509_STORE_CTX *ctx, X509 *x);
X509 *X509_STORE_CTX_get0_cert(X509_STORE_CTX *ctx);

STACK_OF(X509) *X509_STORE_CTX_get1_chain(X509_STORE_CTX *ctx);

const char *X509_verify_cert_error_string(long n);
```

DESCRIPTION

These functions are typically called after *X509_verify_cert()* has indicated an error or in a verification callback to determine the nature of an error.

X509_STORE_CTX_get_error() returns the error code of **ctx**, see the **ERROR CODES** section for a full description of all error codes.

X509_STORE_CTX_set_error() sets the error code of **ctx** to **s**. For example it might be used in a verification callback to set an error based on additional checks.

X509_STORE_CTX_get_error_depth() returns the **depth** of the error. This is a non-negative integer representing where in the certificate chain the error occurred. If it is zero it occurred in the end entity certificate, one if it is the certificate which signed the end entity certificate and so on.

X509_STORE_CTX_set_error_depth() sets the error **depth**. This can be used in combination with *X509_STORE_CTX_set_error()* to set the depth at which an error condition was detected.

X509_STORE_CTX_get_current_cert() returns the certificate in **ctx** which caused the error or **NULL** if no certificate is relevant.

X509_STORE_CTX_set_current_cert() sets the certificate **x** in **ctx** which caused the error. This value is not intended to remain valid for very long, and remains owned by the caller. It may be examined by a verification callback invoked to handle each error encountered during chain verification and is no longer required after such a callback. If a callback wishes the save the certificate for use after it returns, it needs to increment its reference count via *X509_up_ref(3)*. Once such a*saved* certificate is no longer needed it can be freed with *X509_free(3)*.

X509_STORE_CTX_get0_cert() retrieves an internal pointer to the certificate being verified by the **ctx**.

X509_STORE_CTX_get1_chain() returns a complete validate chain if a previous call to *X509_verify_cert()* is successful. If the call to *X509_verify_cert()* is **not** successful the returned chain may be incomplete or invalid. The returned chain persists after the **ctx** structure is freed, when it is no longer needed it should be free up using:

```
sk_X509_pop_free(chain, X509_free);
```

X509_verify_cert_error_string() returns a human readable error string for verification error **n**.

RETURN VALUES

X509_STORE_CTX_get_error() returns **X509_V_OK** or an error code.

X509_STORE_CTX_get_error_depth() returns a non-negative error depth.

X509_STORE_CTX_get_current_cert() returns the certificate which caused the error or **NULL** if no certificate is relevant to the error.

X509_verify_cert_error_string() returns a human readable error string for verification error **n**.

ERROR CODES

A list of error codes and messages is shown below. Some of the error codes are defined but currently never returned: these are described as ''unused''.

X509_V_OK: ok
the operation was successful.

X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT: unable to get issuer certificate
the issuer certificate could not be found: this occurs if the issuer certificate of an untrusted certificate cannot be found.

X509_V_ERR_UNABLE_TO_GET_CRL: unable to get certificate CRL
the CRL of a certificate could not be found.

X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE: unable to decrypt certificate's signature
the certificate signature could not be decrypted. This means that the actual signature value could not be determined rather than it not matching the expected value, this is only meaningful for RSA keys.

X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE: unable to decrypt CRL's signature
the CRL signature could not be decrypted: this means that the actual signature value could not be determined rather than it not matching the expected value. Unused.

X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY: unable to decode issuer public key
the public key in the certificate SubjectPublicKeyInfo could not be read.

X509_V_ERR_CERT_SIGNATURE_FAILURE: certificate signature failure
the signature of the certificate is invalid.

X509_V_ERR_CRL_SIGNATURE_FAILURE: CRL signature failure
the signature of the certificate is invalid.

X509_V_ERR_CERT_NOT_YET_VALID: certificate is not yet valid
the certificate is not yet valid: the notBefore date is after the current time.

X509_V_ERR_CERT_HAS_EXPIRED: certificate has expired
the certificate has expired: that is the notAfter date is before the current time.

X509_V_ERR_CRL_NOT_YET_VALID: CRL is not yet valid
the CRL is not yet valid.

X509_V_ERR_CRL_HAS_EXPIRED: CRL has expired
the CRL has expired.

X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD: format error in certificate's notBefore field
the certificate notBefore field contains an invalid time.

X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD: format error in certificate's notAfter field
the certificate notAfter field contains an invalid time.

X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD: format error in CRL's lastUpdate field
the CRL lastUpdate field contains an invalid time.

X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD: format error in CRL's nextUpdate field
the CRL nextUpdate field contains an invalid time.

X509_V_ERR_OUT_OF_MEM: out of memory
an error occurred trying to allocate memory. This should never happen.

X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT: self signed certificate
the passed certificate is self signed and the same certificate cannot be found in the list of trusted certificates.

X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN: self signed certificate in certificate chain
the certificate chain could be built up using the untrusted certificates but the root could not be found locally.

X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY: unable to get local issuer certificate
the issuer certificate of a locally looked up certificate could not be found. This normally means the list of trusted certificates is not complete.

X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE: unable to verify the first certificate
no signatures could be verified because the chain contains only one certificate and it is not self signed.

X509_V_ERR_CERT_CHAIN_TOO_LONG: certificate chain too long
the certificate chain length is greater than the supplied maximum depth. Unused.

X509_V_ERR_CERT_REVOKED: certificate revoked
the certificate has been revoked.

X509_V_ERR_INVALID_CA: invalid CA certificate
a CA certificate is invalid. Either it is not a CA or its extensions are not consistent with the supplied purpose.

X509_V_ERR_PATH_LENGTH_EXCEEDED: path length constraint exceeded
the basicConstraints path-length parameter has been exceeded.

X509_V_ERR_INVALID_PURPOSE: unsupported certificate purpose
the supplied certificate cannot be used for the specified purpose.

X509_V_ERR_CERT_UNTRUSTED: certificate not trusted
the root CA is not marked as trusted for the specified purpose.

X509_V_ERR_CERT_REJECTED: certificate rejected
the root CA is marked to reject the specified purpose.

X509_V_ERR_SUBJECT_ISSUER_MISMATCH: subject issuer mismatch
the current candidate issuer certificate was rejected because its subject name did not match the issuer name of the current certificate. This is only set if issuer check debugging is enabled it is used for status notification and is **not** in itself an error.

X509_V_ERR_AKID_SKID_MISMATCH: authority and subject key identifier mismatch
the current candidate issuer certificate was rejected because its subject key identifier was present and did not match the authority key identifier current certificate. This is only set if issuer check debugging is enabled it is used for status notification and is **not** in itself an error.

X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH: authority and issuer serial number mismatch
the current candidate issuer certificate was rejected because its issuer name and serial number was present and did not match the authority key identifier of the current certificate. This is only set if issuer check debugging is enabled it is used for status notification and is **not** in itself an error.

X509_V_ERR_KEYUSAGE_NO_CERTSIGN:key usage does not include certificate signing
the current candidate issuer certificate was rejected because its keyUsage extension does not permit certificate signing. This is only set if issuer check debugging is enabled it is used for status notification and is **not** in itself an error.

X509_V_ERR_INVALID_EXTENSION: invalid or inconsistent certificate extension
A certificate extension had an invalid value (for example an incorrect encoding) or some value inconsistent with other extensions.

X509_V_ERR_INVALID_POLICY_EXTENSION: invalid or inconsistent certificate policy extension
A certificate policies extension had an invalid value (for example an incorrect encoding) or some value inconsistent with other extensions. This error only occurs if policy processing is enabled.

X509_V_ERR_NO_EXPLICIT_POLICY: no explicit policy
The verification flags were set to require and explicit policy but none was present.

X509_V_ERR_DIFFERENT_CRL_SCOPE: Different CRL scope
The only CRLs that could be found did not match the scope of the certificate.

X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE: Unsupported extension feature
Some feature of a certificate extension is not supported. Unused.

X509_V_ERR_PERMITTED_VIOLATION: permitted subtree violation
A name constraint violation occurred in the permitted subtrees.

X509_V_ERR_EXCLUDED_VIOLATION: excluded subtree violation
A name constraint violation occurred in the excluded subtrees.

X509_V_ERR_SUBTREE_MINMAX: name constraints minimum and maximum not supported
A certificate name constraints extension included a minimum or maximum field: this is not supported.

X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE: unsupported name constraint type
An unsupported name constraint type was encountered. OpenSSL currently only supports directory name, DNS name, email and URI types.

X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX: unsupported or invalid name constraint syntax
The format of the name constraint is not recognised: for example an email address format of a form not mentioned in RFC3280. This could be caused by a garbage extension or some new feature not currently supported.

X509_V_ERR_CRL_PATH_VALIDATION_ERROR: CRL path validation error
An error occurred when attempting to verify the CRL path. This error can only happen if extended CRL checking is enabled.

X509_V_ERR_APPLICATION_VERIFICATION: application verification failure
an application specific error. This will never be returned unless explicitly set by an application.

NOTES

The above functions should be used instead of directly referencing the fields in the **X509_VERIFY_CTX** structure.

In versions of OpenSSL before 1.0 the current certificate returned by *X509_STORE_CTX_get_current_cert()* was never **NULL**. Applications should check the return value before printing out any debugging information relating to the current certificate.

If an unrecognised error code is passed to *X509_verify_cert_error_string()* the numerical value of the unknown code is returned in a static buffer. This is not thread safe but will never happen unless an invalid code is passed.

SEE ALSO

X509_verify_cert(3) , *X509_up_ref(3)* , *X509_free(3)* .

COPYRIGHT