NAME

SSL_CTX_load_verify_locations, SSL_CTX_set_default_verify_paths, SSL_CTX_set_default_verify_dir, SSL_CTX_set_default_verify_file - set default locations for trusted CA certificates

SYNOPSIS

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

int SSL_CTX_load_verify_locations(SSL_CTX *ctx, const char *CAfile,
const char *CApath);

int SSL_CTX_set_default_verify_paths(SSL_CTX *ctx);

int SSL_CTX_set_default_verify_dir(SSL_CTX *ctx);

int SSL_CTX_set_default_verify_file(SSL_CTX *ctx);
```

DESCRIPTION

SSL_CTX_load_verify_locations() specifies the locations for **ctx**, at which CA certificates for verification purposes are located. The certificates available via **CAfile** and **CApath** are trusted.

SSL_CTX_set_default_verify_paths() specifies that the default locations from which CA certificates are loaded should be used. There is one default directory and one default file. The default CA certificates directory is called ‘‘certs’’ in the default OpenSSL directory. Alternatively the SSL_CERT_DIR environment variable can be defined to override this location. The default CA certificates file is called ‘‘cert.pem’’ in the default OpenSSL directory. Alternatively the SSL_CERT_FILE environment variable can be defined to override this location.

SSL_CTX_set_default_verify_dir() is similar to *SSL_CTX_set_default_verify_paths()* except that just the default directory is used.

SSL_CTX_set_default_verify_file() is similar to *SSL_CTX_set_default_verify_paths()* except that just the default file is used.

NOTES

If **CAfile** is not NULL, it points to a file of CA certificates in PEM format. The file can contain several CA certificates identified by

```
-----BEGIN CERTIFICATE-----
... (CA certificate in base64 encoding) ...
-----END CERTIFICATE-----
```

sequences. Before, between, and after the certificates text is allowed which can be used e.g. for descriptions of the certificates.

The **CAfile** is processed on execution of the *SSL_CTX_load_verify_locations()* function.

If **CApath** is not NULL, it points to a directory containing CA certificates in PEM format. The files each contain one CA certificate. The files are looked up by the CA subject name hash value, which must hence be available. If more than one CA certificate with the same name hash value exist, the extension must be different (e.g. 9d66eef0.0, 9d66eef0.1 etc). The search is performed in the ordering of the extension number, regardless of other properties of the certificates. Use the **c_rehash** utility to create the necessary links.

The certificates in **CApath** are only looked up when required, e.g. when building the certificate chain or when actually performing the verification of a peer certificate.

When looking up CA certificates, the OpenSSL library will first search the certificates in **CAfile**, then those in **CApath**. Certificate matching is done based on the subject name, the key identifier (if present), and the serial number as taken from the certificate to be verified. If these data do not match, the next certificate will be tried. If a first certificate matching the parameters is found, the verification process will be performed; no other certificates for the same parameters will be searched in case of failure.

In server mode, when requesting a client certificate, the server must send the list of CAs of which it will accept client certificates. This list is not influenced by the contents of**CAfile** or **CA path** and must explicitly be set using the *SSL_CTX_set_client_CA_list(3)* family of functions.

When building its own certificate chain, an OpenSSL client/server will try to fill in missing certificates from **CAfile**/**CApath**, if the certificate chain was not explicitly specified (see *SSL_CTX_add_extra_chain_cert(3)*, *SSL_CTX_use_certificate(3)*.

WARNINGS

If several CA certificates matching the name, key identifier, and serial number condition are available, only the first one will be examined. This may lead to unexpected results if the same CA certificate is available with different expiration dates. If a "certificate expired" verification error occurs, no other certificate will be searched. Make sure to not have expired certificates mixed with valid ones.

EXAMPLES

Generate a CA certificate file with descriptive text from the CA certificates ca1.pem ca2.pem ca3.pem:

```
#!/bin/sh
rm CAfile.pem
for i in ca1.pem ca2.pem ca3.pem ; do
openssl x509 -in $i -text >> CAfile.pem
done
```

Prepare the directory /some/where/certs containing several CA certificates for use as **CApath**:

```
cd /some/where/certs
c_rehash .
```

RETURN VALUES

For SSL_CTX_load_verify_locations the following return values can occur:

0 The operation failed because **CAfile** and **CApath** are NULL or the processing at one of the locations specified failed. Check the error stack to find out the reason.

1 The operation succeeded.

SSL_CTX_set_default_verify_paths(), *SSL_CTX_set_default_verify_dir()* and *SSL_CTX_set_default_verify_file()* all return 1 on success or 0 on failure. A missing default location is still treated as a success.

SEE ALSO

ssl(3), *SSL_CTX_set_client_CA_list(3)*, *SSL_get_client_CA_list(3)*, *SSL_CTX_use_certificate(3)*, *SSL_CTX_add_extra_chain_cert(3)*, *SSL_CTX_set_cert_store(3)*, *SSL_CTX_set_client_CA_list(3)*

COPYRIGHT