NAME

OPENSSL_INIT_new, OPENSSL_INIT_set_config_filename, OPENSSL_INIT_set_config_appname, OPENSSL_INIT_set_config_file_flags, OPENSSL_INIT_free, OPENSSL_init_crypto, OPENSSL_cleanup, OPENSSL_atexit, OPENSSL_thread_stop_ex, OPENSSL_thread_stop - OpenSSL initialisation and deinitialisation functions

SYNOPSIS

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

void OPENSSL_cleanup(void);
int OPENSSL_init_crypto(uint64_t opts, const OPENSSL_INIT_SETTINGS *settings);
int OPENSSL_atexit(void (*handler)(void));
void OPENSSL_thread_stop_ex(OSSL_LIB_CTX *ctx);
void OPENSSL_thread_stop(void);

OPENSSL_INIT_SETTINGS *OPENSSL_INIT_new(void);
int OPENSSL_INIT_set_config_filename(OPENSSL_INIT_SETTINGS *init,
                                     const char* filename);
int OPENSSL_INIT_set_config_file_flags(OPENSSL_INIT_SETTINGS *init,
                                       unsigned long flags);
int OPENSSL_INIT_set_config_appname(OPENSSL_INIT_SETTINGS *init,
                                    const char* name);
void OPENSSL_INIT_free(OPENSSL_INIT_SETTINGS *init);
```

DESCRIPTION

During normal operation OpenSSL (libcrypto) will allocate various resources at start up that must, subsequently, be freed on close down of the library. Additionally some resources are allocated on a per thread basis (if the application is multi-threaded), and these resources must be freed prior to the thread closing.

As of version 1.1.0 OpenSSL will automatically allocate all resources that it needs so no explicit initialisation is required. Similarly it will also automatically deinitialise as required.

However, there may be situations when explicit initialisation is desirable or needed, for example when some nondefault initialisation is required. The function **OPENSSL_init_crypto()** can be used for this purpose for libcrypto (see also **OPENSSL_init_ssl**(3) for the libssl equivalent).

Numerous internal OpenSSL functions call **OPENSSL_init_crypto()**. Therefore, in order to perform nondefault initialisation, **OPENSSL_init_crypto()** MUST be called by application code prior to any other OpenSSL function calls.

The **opts** parameter specifies which aspects of libcrypto should be initialised. Valid options are:

OPENSSL_INIT_NO_LOAD_CRYPTO_STRINGS
: Suppress automatic loading of the libcrypto error strings. This option is not a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_LOAD_CRYPTO_STRINGS** will be ignored.

OPENSSL_INIT_LOAD_CRYPTO_STRINGS
: Automatic loading of the libcrypto error strings. With this option the library will automatically load the libcrypto error strings. This option is a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_NO_LOAD_CRYPTO_STRINGS** will be ignored.

OPENSSL_INIT_ADD_ALL_CIPHERS
: With this option the library will automatically load and make available all libcrypto ciphers. This option is a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_NO_ADD_ALL_CIPHERS** will be ignored.

OPENSSL_INIT_ADD_ALL_DIGESTS
: With this option the library will automatically load and make available all libcrypto digests. This option is a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_NO_ADD_ALL_DIGESTS** will be ignored.

OPENSSL_INIT_NO_ADD_ALL_CIPHERS
: With this option the library will suppress automatic loading of libcrypto ciphers. This option is not a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_ADD_ALL_CIPHERS** will be ignored.

OPENSSL_INIT_NO_ADD_ALL_DIGESTS
: With this option the library will suppress automatic loading of libcrypto digests. This option is not a default option. Once selected subsequent calls to **OPENSSL_init_crypto()** with the option **OPENSSL_INIT_ADD_ALL_DIGESTS** will be ignored.

OPENSSL_INIT_LOAD_CONFIG
: With this option an OpenSSL configuration file will be automatically loaded and used by calling **OPENSSL_config()**. This is a default option. Note that in OpenSSL 1.1.1 this was the default for libssl but not for libcrypto (see **OPENSSL_init_ssl**(3) for further details about libssl initialisation). In OpenSSL 1.1.0 this was a nondefault option for both libssl and libcrypto. See the description of **OPENSSL_INIT_new()**, below.

OPENSSL_INIT_NO_LOAD_CONFIG
: With this option the loading of OpenSSL configuration files will be suppressed. It is the equivalent of calling **OPENSSL_no_config()**. This is not a default option.

OPENSSL_INIT_ASYNC
: With this option the library with automatically initialise the libcrypto async sub-library (see **ASYNC_start_job**(3)). This is a default option.

OPENSSL_INIT_ENGINE_RDRAND
: With this option the library will automatically load and initialise the RDRAND engine (if available). This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_DYNAMIC
: With this option the library will automatically load and initialise the dynamic engine. This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_OPENSSL
: With this option the library will automatically load and initialise the openssl engine. This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_CRYPTODEV
: With this option the library will automatically load and initialise the cryptodev engine (if available). This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_CAPI
: With this option the library will automatically load and initialise the CAPI engine (if available). This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_PADLOCK
: With this option the library will automatically load and initialise the padlock engine (if available). This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_AFALG
: With this option the library will automatically load and initialise the AFALG engine. This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ENGINE_ALL_BUILTIN
: With this option the library will automatically load and initialise all the built in engines listed above with the exception of the openssl and afalg engines. This not a default option and is deprecated in OpenSSL 3.0.

OPENSSL_INIT_ATFORK
: With this option the library will register its fork handlers. See **OPENSSL_fork_prepare**(3) for details.

OPENSSL_INIT_NO_ATEXIT
: By default OpenSSL will attempt to clean itself up when the process exits via an "atexit" handler. Using this option suppresses that behaviour. This means that the application will have to clean up OpenSSL explicitly using **OPENSSL_cleanup()**.

Multiple options may be combined together in a single call to **OPENSSL_init_crypto()**. For example:

```
OPENSSL_init_crypto(OPENSSL_INIT_NO_ADD_ALL_CIPHERS
                    | OPENSSL_INIT_NO_ADD_ALL_DIGESTS, NULL);
```

The **OPENSSL_cleanup()** function deinitialises OpenSSL (both libcrypto and libssl). All resources allocated by OpenSSL are freed. Typically there should be no need to call this function directly as it is initiated automatically on application exit. This is done via the standard C library **atexit()** function. In the event that the application will close in a manner that will not call the registered **atexit()** handlers then the application should call **OPENSSL_cleanup()** directly. Developers of libraries using OpenSSL are discouraged from calling this function and should instead, typically, rely on auto-deinitialisation. This is to avoid error conditions where both an application and a library it depends on both use OpenSSL, and the library deinitialises it before the application has finished using it.

Once **OPENSSL_cleanup()** has been called the library cannot be reinitialised. Attempts to call **OPENSSL_init_crypto()** will fail and an ERR_R_INIT_FAIL error will be added to the error stack. Note that because initialisation has failed OpenSSL error strings will not be available, only an error code. This code can be put through the openssl errstr command line application to produce a human readable error (see **openssl-errstr**(1)).

The **OPENSSL_atexit()** function enables the registration of a function to be called during **OPENSSL_cleanup()**. Stop handlers are called after deinitialisation of resources local to a thread, but before other process wide resources are freed. In the event that multiple stop handlers are registered, no guarantees are made about the order of execution.

The **OPENSSL_thread_stop_ex()** function deallocates resources associated with the current thread for the given OSSL_LIB_CTX **ctx**. The **ctx** parameter can be NULL in which case the default OSSL_LIB_CTX is used.

Typically, this function will be called automatically by the library when the thread exits as long as the OSSL_LIB_CTX has not been freed before the thread exits. If **OSSL_LIB_CTX_free()** is called

OPENSSL_thread_stop_ex will be called automatically for the current thread (but not any other threads that may have used this OSSL_LIB_CTX).

OPENSSL_thread_stop_ex should be called on all threads that will exit after the OSSL_LIB_CTX is freed. Typically this is not necessary for the default OSSL_LIB_CTX (because all resources are cleaned up on library exit) except if thread local resources should be freed before library exit, or under the circumstances described in the NOTES section below.

OPENSSL_thread_stop() is the same as **OPENSSL_thread_stop_ex()** except that the default OSSL_LIB_CTX is always used.

The **OPENSSL_INIT_LOAD_CONFIG** flag will load a configuration file, as with **CONF_modules_load_file**(3) with NULL filename and application name and the **CONF_MFLAGS_IGNORE_MISSING_FILE**, **CONF_MFLAGS_IGNORE_RETURN_CODES** and **CONF_MFLAGS_DEFAULT_SECTION** flags. The filename, application name, and flags can be customized by providing a non-null **OPENSSL_INIT_SETTINGS** object. The object can be allocated via **OPENSSL_INIT_new()**. The **OPENSSL_INIT_set_config_filename()** function can be used to specify a nondefault filename, which is copied and need not refer to persistent storage. Similarly, **OPENSSL_INIT_set_config_appname()** can be used to specify a nondefault application name. Finally, OPENSSL_INIT_set_file_flags can be used to specify nondefault flags. If the **CONF_MFLAGS_IGNORE_RETURN_CODES** flag is not included, any errors in the configuration file will cause an error return from **OPENSSL_init_crypto** or indirectly **OPENSSL_init_ssl**(3). The object can be released with **OPENSSL_INIT_free()** when done.

NOTES

Resources local to a thread are deallocated automatically when the thread exits (e.g. in a pthreads environment, when **pthread_exit()** is called). On Windows platforms this is done in response to a DLL_THREAD_DETACH message being sent to the libcrypto32.dll entry point. Some windows functions may cause threads to exit without sending this message (for example **ExitProcess()**). If the application uses such functions, then the application must free up OpenSSL resources directly via a call to **OPENSSL_thread_stop()** on each thread. Similarly this message will also not be sent if OpenSSL is linked statically, and therefore applications using static linking should also call **OPENSSL_thread_stop()** on each thread. Additionally if OpenSSL is loaded dynamically via **LoadLibrary()** and the threads are not destroyed until after **FreeLibrary()** is called then each thread should call **OPENSSL_thread_stop()** prior to the **FreeLibrary()** call.

On Linux/Unix where OpenSSL has been loaded via **dlopen()** and the application is multi-threaded and if **dlclose()** is subsequently called prior to the threads being destroyed then OpenSSL will not be able to deallocate resources associated with those threads. The application should either call **OPENSSL_thread_stop()** on each thread prior to the **dlclose()** call, or alternatively the original **dlopen()**

call should use the RTLD_NODELETE flag (where available on the platform).

RETURN VALUES

The functions OPENSSL_init_crypto, **OPENSSL_atexit()** and **OPENSSL_INIT_set_config_appname()** return 1 on success or 0 on error.

SEE ALSO

OPENSSL_init_ssl(3)

HISTORY

The **OPENSSL_init_crypto()**, **OPENSSL_cleanup()**, **OPENSSL_atexit()**, **OPENSSL_thread_stop()**, **OPENSSL_INIT_new()**, **OPENSSL_INIT_set_config_appname()** and **OPENSSL_INIT_free()** functions were added in OpenSSL 1.1.0.

COPYRIGHT