NAME

neon - HTTP and WebDAV client library

DESCRIPTION

neon is an HTTP and WebDAV client library. The major abstractions exposed are the HTTP *session*, created by ne_session_create; and the HTTP *request*, created by ne_request_create. HTTP authentication is handled transparently for server and proxy servers, see ne_set_server_auth; complete SSL/TLS support is also included, see ne_ssl_set_verify.

CONVENTIONS

Some conventions are used throughout the neon API, to provide a consistent and simple interface; these are documented below.

Thread-safeness and global initialization

neon itself is implemented to be thread-safe (avoiding any use of global state), but relies on the operating system providing a thread-safe resolver interface. Modern operating systems offer the thread-safe **getaddrinfo** interface, which neon supports; some others implement **gethostbyname** using thread-local storage.

To allow thread-safe use of SSL in the OpenSSL and GnuTLS libraries neon must be configured using the --enable-threadsafe-ssl; if this is done, locking callbacks will be registered by ne_sock_init; note that care must be exercised if neon is used in conjunction with another library which uses OpenSSL or GnuTLS.

Some platforms and libraries used by neon require global initialization before use; notably:

- SIGPIPE signal disposition must be set to *ignored* or otherwise handled to avoid process termination when writing to a socket which has been shutdown by the peer.
- and GnuTLS require global initialization to load shared lookup tables.
- Win32 socket library requires initialization before use.

The ne_sock_init function should be called before any other use of neon to perform any necessary initialization needed for the particular platform. Applications wishing to perform all the necessary process-global initialization steps themselves may omit to call ne_sock_init (and ne_sock_exit); neon neither checks whether these functions are called nor calls them itself.

For some applications and configurations it may be necessary to call ne_i18n_init to initialize the support for internationalization in neon.

Asynchronous signal safety

No function in neon is defined to be "async-signal safe" - that is, no function is safe to call from a signal handler. Any call into the neon library from a signal handler will have undefined behaviour - in other words, it may crash the process.

Functions using global state

Any function in neon may modify the errno global variable as a side-effect. Except where explicitly documented, the value of errno is unspecified after any neon function call.

Other than in the use of errno, the only functions which use or modify process-global state in neon are as follows:

- ne_i18n_init, and ne_sock_exit, as described above
- and **ne_debug**, if enabled at compile time; for debugging output
- for installing a process-global callback to be invoked on **malloc** failure

Namespaces

To avoid possible collisions between names used for symbols and preprocessor macros by an application and the libraries it uses, it is good practice for each library to reserve a particular *namespace prefix*. An application which ensures it uses no names with these prefixes is then guaranteed to avoid such collisions.

The neon library reserves the use of the namespace prefixes ne_ and NE_. The libraries used by neon may also reserve certain namespaces; collisions between these libraries and a neon-based application will not be detected at compile time, since the underlying library interfaces are not exposed through the neon header files. Such collisions can only be detected at link time, when the linker attempts to resolve symbols. The following list documents some of the namespaces claimed by libraries used by neon; this list may be incomplete.

SSL, ssl, TLS, tls, ERR_, BIO_, d2i_, i2d_, ASN1_
: Some of the many prefixes used by the OpenSSL library; little attempt has been made to keep exported symbols within any particular prefixes for this library.

gnutls_, gcry_, gpg_
: Namespaces used by the GnuTLS library (and dependencies thereof)

XML_, Xml[A-Z]
: Namespaces used by the expat library.

xml[A-Z], html[A-Z], docb[A-Z]
: Namespaces used by the libxml2 library; a relatively small number of symbols are used without these prefixes.

inflate, deflate, crc32, compress, uncompress, adler32, zlib
: Namespaces used by the zlib library; a relatively small number of symbols are used without these prefixes.

krb5, gss, GSS, asn1, decode_krb5, encode_krb5, profile, mit
: Some of the prefixes used by the MIT GSSAPI library and dependencies thereof; a number of symbols lie outside these prefixes.

pakchois_
: Namespace used by the pakchois library.

px_
: Namespace used by the libproxy library.

Argument validation

neon does not attempt to validate that the parameters passed to functions conform to the API (for instance, checking that pointer arguments are not NULL). Any use of the neon API which is not documented to produce a certain behaviour results is said to produce *undefined behaviour*; it is likely that neon will segfault under these conditions.

URI paths, WebDAV metadata

The path strings passed to any function must be *URI-encoded* by the application; neon never performs any URI encoding or decoding internally. WebDAV property names and values must be valid UTF-8 encoded Unicode strings.

User interaction

As a pure library interface, neon will never produce output on **stdout** or **stderr**; all user interaction is the responsibility of the application.

Memory handling

neon does not attempt to cope gracefully with an out-of-memory situation; instead, by default, the **abort** function is called to immediately terminate the process. An application may register a custom function which will be called before **abort** in such a situation; see ne_oom_callback.

Callbacks and userdata

Whenever a callback is registered, a userdata pointer is also used to allow the application to associate a context with the callback. The userdata is of type **void ***, allowing any pointer to be used.

Large File Support

Since version 0.27.0, neon transparently uses the "LFS transitional" interfaces in functions which use file descriptors. This allows use of files larger than 2GiB on platforms with a native 32-bit off_t type, where LFS support is available.

Some neon interfaces use the ne_off_t type, which is defined to be either off_t or off64_t according to whether LFS support is detected at build time. neon does not use or require the -D_FILE_OFFSET_BITS=64 macro definition.

SEE ALSO

ne_session_create(3), ne_oom_callback, **https://notroj.github.io/neon/**

AUTHOR

Joe Orton <neon@lists.manyfish.co.uk>
Author.

COPYRIGHT