NAME

ne_request_create, ne_request_dispatch, ne_request_destroy - low-level HTTP request handling

SYNOPSIS

#include <ne_request.h>

ne_request *ne_request_create(ne_session ***_session_, const char ***_method_**, const char ***_path_**);**

int ne_request_dispatch(ne_request ***_req_);**

void ne_request_destroy(ne_request ***_req_);**

DESCRIPTION

The **ne_request** object represents an HTTP request and the associated response. The **ne_request_create** function creates a new request object for the given _session_. The target resource for the request is identified by the _path_, and the method to be performed on that resource via the _method_ parameter.

The _path_ string used must conform to the abs_path definition given in RFC2396, with an optional "?query" part, and must be URI-escaped by the caller (for instance, using **ne_path_escape**). If the string comes from an untrusted source, failure to perform URI-escaping results in a security vulnerability.

To dispatch a request, and process the response, the **ne_request_dispatch** function can be used. An alternative is to use the (more complex, but more flexible) combination of the **ne_begin_request**, **ne_end_request**, and **ne_read_response_block** functions; see **ne_begin_request**.

To add extra headers in the request, the functions ne_add_request_header and ne_print_request_header can be used. To include a message body with the request, one of the functions **ne_set_request_body_buffer**, ne_set_request_body_fd, or **ne_set_request_body_provider** can be used.

The return value of **ne_request_dispatch** indicates merely whether the request was sent and the response read successfully. To discover the result of the operation, ne_get_status, along with any processing of the response headers and message body.

A request can only be dispatched once: calling **ne_request_dispatch** more than once on a single **ne_request** object produces undefined behaviour. Once all processing associated with the request object is complete, use the **ne_request_destroy** function to destroy the resources associated with it. Any subsequent use of the request object produces undefined behaviour.

If a request is being using a non-idempotent method such as POST, the NE_REQFLAG_IDEMPOTENT flag should be disabled; see ne_set_request_flag.

RETURN VALUE

The **ne_request_create** function returns a pointer to a request object (and never NULL).

The **ne_request_dispatch** function returns zero if the request was dispatched successfully, and a non-zero error code otherwise.

ERRORS

NE_ERROR
: Request failed (see session error string)

NE_LOOKUP
: The DNS lookup for the server (or proxy server) failed.

NE_AUTH
: Authentication failed on the server.

NE_PROXYAUTH
: Authentication failed on the proxy server.

NE_CONNECT
: A connection to the server could not be established.

NE_TIMEOUT
: A timeout occurred while waiting for the server to respond.

EXAMPLE

An example of applying a MKCOL operation to the resource at the location http://www.example.com/foo/bar/:

```
ne_session *sess = ne_session_create("http", "www.example.com", 80);
ne_request *req = ne_request_create(sess, "MKCOL", "/foo/bar/");
if (ne_request_dispatch(req)) {
   printf("Request failed: %s\n", ne_get_error(sess));
}
ne_request_destroy(req);
```

SEE ALSO

ne_get_error, ne_set_error, ne_get_status, ne_add_request_header, ne_set_request_body_buffer, ne_set_request_flag.

AUTHOR

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

COPYRIGHT