NAME

err - error codes

SYNOPSIS

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

unsigned long ERR_get_error(void);
unsigned long ERR_peek_error(void);
unsigned long ERR_get_error_line(const char **file, int *line);
unsigned long ERR_peek_error_line(const char **file, int *line);
unsigned long ERR_get_error_line_data(const char **file, int *line,
const char **data, int *flags);
unsigned long ERR_peek_error_line_data(const char **file, int *line,
const char **data, int *flags);

int ERR_GET_LIB(unsigned long e);
int ERR_GET_FUNC(unsigned long e);
int ERR_GET_REASON(unsigned long e);

void ERR_clear_error(void);

char *ERR_error_string(unsigned long e, char *buf);
const char *ERR_lib_error_string(unsigned long e);
const char *ERR_func_error_string(unsigned long e);
const char *ERR_reason_error_string(unsigned long e);

void ERR_print_errors(BIO *bp);
void ERR_print_errors_fp(FILE *fp);

void ERR_load_crypto_strings(void);
void ERR_free_strings(void);

void ERR_remove_state(unsigned long pid);

void ERR_put_error(int lib, int func, int reason, const char *file,
int line);
void ERR_add_error_data(int num, ...);

void ERR_load_strings(int lib,ERR_STRING_DATA str[]);
unsigned long ERR_PACK(int lib, int func, int reason);
int ERR_get_next_error_library(void);
```

DESCRIPTION

When a call to the OpenSSL library fails, this is usually signalled by the return value, and an error code is stored in an error queue associated with the current thread. The **err** library provides functions to obtain these error codes and textual error messages.

The *ERR_get_error(3)* manpage describes how to access error codes.

Error codes contain information about where the error occurred, and what went wrong. *ERR_GET_LIB(3)* describes how to extract this information. A method to obtain human-readable error messages is described in *ERR_error_string(3)*.

ERR_clear_error(3) can be used to clear the error queue.

Note that *ERR_remove_state(3)* should be used to avoid memory leaks when threads are terminated.

ADDING NEW ERROR CODES TO OPENSSL

See *ERR_put_error(3)* if you want to record error codes in the OpenSSL error system from within your application.

The remainder of this section is of interest only if you want to add new error codes to OpenSSL or add error codes from external libraries.

Reporting errors

Each sub-library has a specific macro *XXXerr()* that is used to report errors. Its first argument is a function code **XXX_F_...**, the second argument is a reason code **XXX_R_...**. Function codes are derived from the function names; reason codes consist of textual error descriptions. For example, the function *ssl23_read()* reports a ''handshake failure'' as follows:

```
SSLerr(SSL_F_SSL23_READ, SSL_R_SSL_HANDSHAKE_FAILURE);
```

Function and reason codes should consist of upper case characters, numbers and underscores only. The error file generation script translates function codes into function names by looking in the header files for an appropriate function name, if none is found it just uses the capitalized form such as ''SSL23_READ'' in the above example.

The trailing section of a reason code (after the ''_R_'') is translated into lower case and underscores changed to spaces.

When you are using new function or reason codes, run **make errors**. The necessary**#define**s will then automatically be added to the sub-library's header file.

Although a library will normally report errors using its own specific XXXerr macro, another library's macro can be used. This is normally only done when a library wants to include ASN1 code which must use the *ASN1err()* macro.

Adding new libraries

When adding a new sub-library to OpenSSL, assign it a library number **ERR_LIB_XXX**, define a macro *XXXerr()* (both in **err.h**), add its name to **ERR_str_libraries**[] (in **crypto/err/err.c**), and add `ERR_load_XXX_strings()` to the *ERR_load_crypto_strings()* function (in **crypto/err/err_all.c**). Finally, add an entry

```
L XXX xxx.h xxx_err.c
```

to **crypto/err/openssl.ec**, and add **xxx_err.c** to the Makefile. Running**mak e errors**will then generate a file **xxx_err.c**, and add all error codes used in the library to **xxx.h**.

Additionally the library include file must have a certain form. Typically it will initially look like this:

```
#ifndef HEADER_XXX_H
#define HEADER_XXX_H

#ifdef __cplusplus
extern "C" {
#endif

/* Include files */

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

/* Macros, structures and function prototypes */


/* BEGIN ERROR CODES */
```

The **BEGIN ERROR CODES** sequence is used by the error code generation script as the point to place new error codes, any text after this point will be overwritten when **make errors**is run. The closing #endif etc will be automatically added by the script.

The generated C error code file **xxx_err.c** will load the header files **stdio.h**, **openssl/err.h** and **openssl/xxx.h** so the header file must load any additional header files containing any definitions it uses.

USING ERROR CODES IN EXTERNAL LIBRARIES

It is also possible to use OpenSSL's error code scheme in external libraries. The library needs to load its own codes and call the OpenSSL error code insertion script **mkerr.pl** explicitly to add codes to the header file and generate the C error code file. This will normally be done if the external library needs to generate new ASN1 structures but it can also be used to add more general purpose error code handling.

TBA more details

INTERNALS

The error queues are stored in a hash table with one **ERR_STATE** entry for each pid. *ERR_get_state()* returns the current thread's **ERR_STATE**. An **ERR_STATE** can hold up to **ERR_NUM_ERRORS** error codes. When more error codes are added, the old ones are overwritten, on the assumption that the most recent errors are most important.

Error strings are also stored in hash table. The hash tables can be obtained by calling ERR_get_err_state_table(void) and ERR_get_string_table(void) respectively.

SEE ALSO

CRYPTO_set_locking_callback(3), *ERR_get_error(3)*, *ERR_GET_LIB(3)*, *ERR_clear_error(3)*, *ERR_error_string(3)*, *ERR_print_errors(3)*, *ERR_load_crypto_strings(3)*, *ERR_remove_state(3)*, *ERR_put_error(3)*, *ERR_load_strings(3)*, *SSL_get_error(3)*