Rocky Enterprise Linux 9.2 Manual Pages on command 'CRYPTO_EX_free.3ossl'

$ man CRYPTO_EX_free.3ossl

CRYPTO_GET_EX_NEW_INDEX(3ossl) OpenSSL CRYPTO_GET_EX_NEW_INDEX(3ossl)

NAME

CRYPTO_EX_new, CRYPTO_EX_free, CRYPTO_EX_dup, CRYPTO_free_ex_index, CRYPTO_get_ex_new_index, CRYPTO_alloc_ex_data, CRYPTO_set_ex_data, CRYPTO_get_ex_data, CRYPTO_free_ex_data, CRYPTO_new_ex_data - functions supporting application-specific data

SYNOPSIS

```
#include <openssl/crypto.h>
int CRYPTO_get_ex_new_index(int class_index,
                            long argl, void *argp,
                            CRYPTO_EX_new *new_func,
                            CRYPTO_EX_dup *dup_func,
                            CRYPTO_EX_free *free_func);
typedef void CRYPTO_EX_new(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
                           int idx, long argl, void *argp);
typedef void CRYPTO_EX_free(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
                            int idx, long argl, void *argp);
typedef int CRYPTO_EX_dup(CRYPTO_EX_DATA *to, const CRYPTO_EX_DATA *from,
```

```
                        void **from_d, int idx, long argl, void *argp);
 int CRYPTO_new_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad);
 int CRYPTO_alloc_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad,
                        int idx);
 int CRYPTO_set_ex_data(CRYPTO_EX_DATA *r, int idx, void *arg);
 void *CRYPTO_get_ex_data(const CRYPTO_EX_DATA *r, int idx);
 void CRYPTO_free_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *r);
 int CRYPTO_free_ex_index(int class_index, int idx);
```

DESCRIPTION

Several OpenSSL structures can have application-specific data attached to them, known as "exdata." The specific structures are:

- BIO
- DH
- DSA
- EC_KEY
- ENGINE
- EVP_PKEY
- RSA
- SSL
- SSL_CTX
- SSL_SESSION
- UI
- UI_METHOD
- X509
- X509_STORE
- X509_STORE_CTX

In addition, the APP name is reserved for use by application code.

Each is identified by an CRYPTO_EX_INDEX_xxx define in the header file <openssl/crypto.h>. In addition, CRYPTO_EX_INDEX_APP is reserved for applications to use this facility for their own structures.

The API described here is used by OpenSSL to manipulate exdata for specific structures. Since the application data can be anything at all it is passed and retrieved as a void * type.

The CRYPTO_EX_DATA type is opaque. To initialize the exdata part of a structure, call CRYPTO_new_ex_data(). This is only necessary for CRYPTO_EX_INDEX_APP objects.

Exdata types are identified by an index, an integer guaranteed to be unique within structures for the lifetime of the program. Applications using exdata typically call CRYPTO_get_ex_new_index at startup, and store the result in a global variable, or write a wrapper function to provide lazy evaluation. The class_index should be one of the CRYPTO_EX_INDEX_xxx values. The argl and argp parameters are saved to be passed to the callbacks but are otherwise not used. In order to transparently manipulate exdata, three callbacks must be provided. The semantics of those callbacks are described below.

When copying or releasing objects with exdata, the callback functions are called in increasing order of their index value.

If a dynamic library can be unloaded, it should call CRYPTO_free_ex_index() when this is done. This will replace the callbacks with no-ops so that applications don't crash. Any existing exdata will be leaked.

To set or get the exdata on an object, the appropriate type-specific routine must be used. This is because the containing structure is opaque and the CRYPTO_EX_DATA field is not accessible. In both API's, the idx parameter should be an already-created index value.

When setting exdata, the pointer specified with a particular index is saved, and returned on a subsequent "get" call. If the application is going to release the data, it must make sure to set a NULL value at the index, to avoid likely double-free crashes.

The function CRYPTO_free_ex_data is used to free all exdata attached to a structure. The appropriate type-specific routine must be used. The class_index identifies the structure type, the obj is a pointer to the actual structure, and r is a pointer to the structure's exdata field.

Callback Functions

This section describes how the callback functions are used.

Applications that are defining their own exdata using

CYPRTO_EX_INDEX_APP must call them as described here.

When a structure is initially allocated (such as RSA_new()) then the new_func() is called for every defined index. There is no requirement that the entire parent, or containing, structure has been set up. The new_func() is typically used only to allocate memory to store the exdata, and perhaps an "initialized" flag within that memory. The exdata value may be allocated later on with CRYPTO_alloc_ex_data(), or may be set by calling CRYPTO_set_ex_data().

When a structure is free'd (such as SSL_CTX_free()) then the free_func() is called for every defined index. Again, the state of the parent structure is not guaranteed. The free_func() may be called with a NULL pointer.

Both new_func() and free_func() take the same parameters. The parent is the pointer to the structure that contains the exdata. The ptr is the current exdata item; for new_func() this will typically be NULL. The r parameter is a pointer to the exdata field of the object. The idx is the index and is the value returned when the callbacks were initially registered via CRYPTO_get_ex_new_index() and can be used if the same callback handles different types of exdata.

dup_func() is called when a structure is being copied. This is only done for SSL, SSL_SESSION, EC_KEY objects and BIO chains via BIO_dup_chain(). The to and from parameters are pointers to the destination and source CRYPTO_EX_DATA structures, respectively. The *from_d parameter is a pointer to the source exdata. When the dup_func() returns, the value in *from_d is copied to the destination ex_data. If the pointer contained in *pptr is not modified by the dup_func(), then both to and from will point to the same data. The idx, argl and argp parameters are as described for the other two callbacks. If the dup_func() returns 0 the whole CRYPTO_dup_ex_data() will fail.

RETURN VALUES

CRYPTO_get_ex_new_index() returns a new index or -1 on failure.

CRYPTO_free_ex_index(), CRYPTO_alloc_ex_data() and CRYPTO_set_ex_data()

return 1 on success or 0 on failure.

CRYPTO_get_ex_data() returns the application data or NULL on failure; note that NULL may be a valid value.

dup_func() should return 0 for failure and 1 for success.

HISTORY

CRYPTO_alloc_ex_data() was added in OpenSSL 3.0.

The signature of the dup_func() callback was changed in OpenSSL 3.0 to use the type void ** for from_d. Previously this parameter was of type void *.

Support for ENGINE "exdata" was deprecated in OpenSSL 3.0.

COPYRIGHT