

NAME

BIO_set_callback_ex, BIO_get_callback_ex, BIO_set_callback, BIO_get_callback, BIO_set_callback_arg, BIO_get_callback_arg, BIO_debug_callback, BIO_debug_callback_ex, BIO_callback_fn_ex, BIO_callback_fn – BIO callback functions

SYNOPSIS

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

```
typedef long (*BIO_callback_fn_ex)(BIO *b, int oper, const char *argp,
                                   size_t len, int argi,
                                   long argl, int ret, size_t *processed);
```

```
void BIO_set_callback_ex(BIO *b, BIO_callback_fn_ex callback);
BIO_callback_fn_ex BIO_get_callback_ex(const BIO *b);
```

```
void BIO_set_callback_arg(BIO *b, char *arg);
char *BIO_get_callback_arg(const BIO *b);
```

```
long BIO_debug_callback_ex(BIO *bio, int oper, const char *argp, size_t len,
                           int argi, long argl, int ret, size_t *processed);
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros** (7):

```
typedef long (*BIO_callback_fn)(BIO *b, int oper, const char *argp, int argi,
                                long argl, long ret);
```

```
void BIO_set_callback(BIO *b, BIO_callback_fn cb);
BIO_callback_fn BIO_get_callback(const BIO *b);
long BIO_debug_callback(BIO *bio, int cmd, const char *argp, int argi,
                        long argl, long ret);
```

```
typedef struct bio_mmsg_cb_args_st {
    BIO_MSG      *msg;
    size_t        stride, num_msg;
    uint64_t      flags;
    size_t        *msgs_processed;
} BIO_MMSG_CB_ARGS;
```

DESCRIPTION

BIO_set_callback_ex() and **BIO_get_callback_ex()** set and retrieve the BIO callback. The callback is called during most high-level BIO operations. It can be used for debugging purposes to trace operations on a BIO or to modify its operation.

BIO_set_callback() and **BIO_get_callback()** set and retrieve the old format BIO callback. New code should not use these functions, but they are retained for backwards compatibility. Any callback set via **BIO_set_callback_ex()** will get called in preference to any set by **BIO_set_callback()**.

BIO_set_callback_arg() and **BIO_get_callback_arg()** are macros which can be used to set and retrieve an argument for use in the callback.

BIO_debug_callback_ex() is a standard debugging callback which prints out information relating to each BIO operation. If the callback argument is set it is interpreted as a BIO to send the information to, otherwise stderr is used. The **BIO_debug_callback()** function is the deprecated version of the same callback for use with the old callback format **BIO_set_callback()** function.

BIO_callback_fn_ex is the type of the callback function and BIO_callback_fn is the type of the old format callback function. The meaning of each argument is described below:

b The BIO the callback is attached to is passed in **b**.

oper

oper is set to the operation being performed. For some operations the callback is called twice, once before and once after the actual operation, the latter case has **oper** or'ed with BIO_CB_RETURN.

len The length of the data requested to be read or written. This is only useful if **oper** is BIO_CB_READ, BIO_CB_WRITE or BIO_CB_GETS.

argp argi argl

The meaning of the arguments **argp**, **argi** and **argl** depends on the value of **oper**, that is the operation being performed.

processed

processed is a pointer to a location which will be updated with the amount of data that was actually read or written. Only used for BIO_CB_READ, BIO_CB_WRITE, BIO_CB_GETS and BIO_CB_PUTS.

ret **ret** is the return value that would be returned to the application if no callback were present. The actual value returned is the return value of the callback itself. In the case of callbacks called before the actual BIO operation 1 is placed in **ret**, if the return value is not positive it will be immediately returned to the application and the BIO operation will not be performed.

The callback should normally simply return **ret** when it has finished processing, unless it specifically wishes to modify the value returned to the application.

CALLBACK OPERATIONS

In the notes below, **callback** defers to the actual callback function that is called.

BIO_free(b)

```
callback_ex(b, BIO_CB_FREE, NULL, 0, 0, 0L, 1L, NULL)
```

or

```
callback(b, BIO_CB_FREE, NULL, 0L, 0L, 1L)
```

is called before the free operation.

BIO_read_ex(b, data, dlen, readbytes)

```
callback_ex(b, BIO_CB_READ, data, dlen, 0, 0L, 1L, NULL)
```

or

```
callback(b, BIO_CB_READ, data, dlen, 0L, 1L)
```

is called before the read and

```
callback_ex(b, BIO_CB_READ | BIO_CB_RETURN, data, dlen, 0, 0L, retvalue,
            &readbytes)
```

or

```
callback(b, BIO_CB_READ|BIO_CB_RETURN, data, dlen, 0L, retvalue)
```

after.

BIO_write(b, data, dlen, written)

```
callback_ex(b, BIO_CB_WRITE, data, dlen, 0, 0L, 1L, NULL)
```

or

```
callback(b, BIO_CB_WRITE, datat, dlen, 0L, 1L)
```

is called before the write and

```
callback_ex(b, BIO_CB_WRITE | BIO_CB_RETURN, data, dlen, 0, 0L, retvalue,
            &written)
```

or

```
callback(b, BIO_CB_WRITE|BIO_CB_RETURN, data, dlen, 0L, retvalue)
```

after.

BIO_gets(b, buf, size)

```
callback_ex(b, BIO_CB_GETS, buf, size, 0, 0L, 1, NULL, NULL)
```

or

```
callback(b, BIO_CB_GETS, buf, size, 0L, 1L)
```

is called before the operation and

```
callback_ex(b, BIO_CB_GETS | BIO_CB_RETURN, buf, size, 0, 0L, retvalue,
            &readbytes)
```

or

```
callback(b, BIO_CB_GETS|BIO_CB_RETURN, buf, size, 0L, retvalue)
```

after.

BIO_puts(b, buf)

```
callback_ex(b, BIO_CB_PUTS, buf, 0, 0, 0L, 1L, NULL);
```

or

```
callback(b, BIO_CB_PUTS, buf, 0, 0L, 1L)
```

is called before the operation and

```
callback_ex(b, BIO_CB_PUTS | BIO_CB_RETURN, buf, 0, 0, 0L, retvalue, &written)
```

or

```
callback(b, BIO_CB_PUTS|BIO_CB_RETURN, buf, 0, 0L, retvalue)
```

after.

BIO_ctrl(BIO *b, int cmd, long larg, void *parg)

```
callback_ex(b, BIO_CB_CTRL, parg, 0, cmd, larg, 1L, NULL)
```

or

```
callback(b, BIO_CB_CTRL, parg, cmd, larg, 1L)
```

is called before the call and

```
callback_ex(b, BIO_CB_CTRL | BIO_CB_RETURN, parg, 0, cmd, larg, ret, NULL)
```

or

```
callback(b, BIO_CB_CTRL|BIO_CB_RETURN, parg, cmd, larg, ret)
```

after.

Note: **cmd == BIO_CTRL_SET_CALLBACK** is special, because **parg** is not the argument of type **BIO_info_cb** itself. In this case **parg** is a pointer to the actual call parameter, see **BIO_callback_ctrl**.

BIO_sendmmsg(BIO *b, BIO_MSG *msg, size_t stride, size_t num_msg, uint64_t flags, size_t *msgs_processed)

```
callback_ex(b, BIO_CB_SENDMMSG, args, 0, 0, 0, 1, NULL)
```

or

```
callback(b, BIO_CB_SENDMMSG, args, 0, 0, 1)
```

is called before the call and

```
callback_ex(b, BIO_CB_SENDMSG | BIO_CB_RETURN, args, ret, 0, 0, ret, NULL)
or
callback(b, BIO_CB_SENDMSG | BIO_CB_RETURN, args, ret, 0, 0, ret)
after.
```

args is a pointer to a **BIO_MMSG_CB_ARGS** structure containing the arguments passed to **BIO_sendmmsg()**. **ret** is the return value of the **BIO_sendmmsg()** call. The return value of **BIO_sendmmsg()** is altered to the value returned by the **BIO_CB_SENDMSG | BIO_CB_RETURN** call.

BIO_recvmmsg(BIO *b, BIO_MSG *msg, size_t stride, size_t num_msg, uint64_t flags, size_t *msgs_processed)

See the documentation for **BIO_sendmmsg()**. **BIO_recvmmsg()** works identically except that **BIO_CB_RECVMSG** is used instead of **BIO_CB_SENDMSG**.

RETURN VALUES

BIO_get_callback_ex() and **BIO_get_callback()** return the callback function previously set by a call to **BIO_set_callback_ex()** and **BIO_set_callback()** respectively.

BIO_get_callback_arg() returns a **char** pointer to the value previously set via a call to **BIO_set_callback_arg()**.

BIO_debug_callback() returns 1 or **ret** if it's called after specific BIO operations.

EXAMPLES

The **BIO_debug_callback_ex()** function is an example, its source is in `crypto/bio/bio_cb.c`

HISTORY

The **BIO_debug_callback_ex()** function was added in OpenSSL 3.0.

BIO_set_callback(), **BIO_get_callback()**, and **BIO_debug_callback()** were deprecated in OpenSSL 3.0. Use the non-deprecated **_ex** functions instead.

COPYRIGHT

Copyright 2000–2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file `LICENSE` in the source distribution or at <https://www.openssl.org/source/license.html>.