

NAME

BIO_ctrl, BIO_callback_ctrl, BIO_ptr_ctrl, BIO_int_ctrl, BIO_reset, BIO_seek, BIO_tell, BIO_flush, BIO_eof, BIO_set_close, BIO_get_close, BIO_pending, BIO_wpending, BIO_ctrl_pending, BIO_ctrl_wpending, BIO_get_info_callback, BIO_set_info_callback, BIO_info_cb, BIO_get_ktls_send, BIO_get_ktls_recv - BIO control operations

SYNOPSIS

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

```
typedef int BIO_info_cb(BIO *b, int state, int res);
```

```
long BIO_ctrl(BIO *bp, int cmd, long larg, void *parg);  
long BIO_callback_ctrl(BIO *b, int cmd, BIO_info_cb *cb);  
void *BIO_ptr_ctrl(BIO *bp, int cmd, long larg);  
long BIO_int_ctrl(BIO *bp, int cmd, long larg, int iarg);
```

```
int BIO_reset(BIO *b);  
int BIO_seek(BIO *b, int ofs);  
int BIO_tell(BIO *b);  
int BIO_flush(BIO *b);  
int BIO_eof(BIO *b);  
int BIO_set_close(BIO *b, long flag);  
int BIO_get_close(BIO *b);  
int BIO_pending(BIO *b);  
int BIO_wpending(BIO *b);  
size_t BIO_ctrl_pending(BIO *b);  
size_t BIO_ctrl_wpending(BIO *b);
```

```
int BIO_get_info_callback(BIO *b, BIO_info_cb **cbp);  
int BIO_set_info_callback(BIO *b, BIO_info_cb *cb);
```

```
int BIO_get_ktls_send(BIO *b);  
int BIO_get_ktls_recv(BIO *b);
```

DESCRIPTION

BIO_ctrl(), **BIO_callback_ctrl()**, **BIO_ptr_ctrl()** and **BIO_int_ctrl()** are BIO "control" operations taking arguments of various types. These functions are not normally called directly, various macros are used instead. The standard macros are described below, macros specific to a particular type of BIO are described in the specific BIOs manual page as well as any special features of the standard calls.

BIO_reset() typically resets a BIO to some initial state, in the case of file related BIOs for example it rewinds the file pointer to the start of the file.

BIO_seek() resets a file related BIO's (that is file descriptor and FILE BIOs) file position pointer to **ofs** bytes from start of file.

BIO_tell() returns the current file position of a file related BIO.

BIO_flush() normally writes out any internally buffered data, in some cases it is used to signal EOF and that no more data will be written.

BIO_eof() returns 1 if the BIO has read EOF, the precise meaning of "EOF" varies according to the BIO type.

BIO_set_close() sets the BIO **b** close flag to **flag**. **flag** can take the value BIO_CLOSE or BIO_NOCLOSE. Typically BIO_CLOSE is used in a source/sink BIO to indicate that the underlying I/O stream should be closed when the BIO is freed.

BIO_get_close() returns the BIOs close flag.

BIO_pending(), **BIO_ctrl_pending()**, **BIO_wpending()** and **BIO_ctrl_wpending()** return the number of pending characters in the BIOs read and write buffers. Not all BIOs support these calls.

BIO_ctrl_pending() and **BIO_ctrl_wpending()** return a size_t type and are functions, **BIO_pending()** and **BIO_wpending()** are macros which call **BIO_ctrl()**.

BIO_get_ktls_send() returns 1 if the BIO is using the Kernel TLS data-path for sending. Otherwise, it returns zero. **BIO_get_ktls_recv()** returns 1 if the BIO is using the Kernel TLS data-path for receiving. Otherwise, it returns zero.

RETURN VALUES

BIO_reset() normally returns 1 for success and ≤ 0 for failure. File BIOs are an exception, they return 0 for success and -1 for failure.

BIO_seek() and **BIO_tell()** both return the current file position on success and -1 for failure, except file BIOs which for **BIO_seek()** always return 0 for success and -1 for failure.

BIO_flush() returns 1 for success and ≤ 0 for failure.

BIO_eof() returns 1 if EOF has been reached, 0 if not, or negative values for failure.

BIO_set_close() returns 1 on success or ≤ 0 for failure.

BIO_get_close() returns the close flag value: **BIO_CLOSE** or **BIO_NOCLOSE**. It also returns other negative values if an error occurs.

BIO_pending(), **BIO_ctrl_pending()**, **BIO_wpending()** and **BIO_ctrl_wpending()** return the amount of pending data. **BIO_pending()** and **BIO_wpending()** return negative value or 0 on error.

BIO_ctrl_pending() and **BIO_ctrl_wpending()** return 0 on error.

BIO_get_ktls_send() returns 1 if the BIO is using the Kernel TLS data-path for sending. Otherwise, it returns zero. **BIO_get_ktls_recv()** returns 1 if the BIO is using the Kernel TLS data-path for receiving. Otherwise, it returns zero.

NOTES

BIO_flush(), because it can write data may return 0 or -1 indicating that the call should be retried later in a similar manner to **BIO_write_ex()**. The **BIO_should_retry()** call should be used and appropriate action taken is the call fails.

The return values of **BIO_pending()** and **BIO_wpending()** may not reliably determine the amount of pending data in all cases. For example in the case of a file BIO some data may be available in the FILE structures internal buffers but it is not possible to determine this in a portably way. For other types of BIO they may not be supported.

Filter BIOs if they do not internally handle a particular **BIO_ctrl()** operation usually pass the operation to the next BIO in the chain. This often means there is no need to locate the required BIO for a particular operation, it can be called on a chain and it will be automatically passed to the relevant BIO. However, this can cause unexpected results: for example no current filter BIOs implement **BIO_seek()**, but this may still succeed if the chain ends in a FILE or file descriptor BIO.

Source/sink BIOs return an 0 if they do not recognize the **BIO_ctrl()** operation.

BUGS

Some of the return values are ambiguous and care should be taken. In particular a return value of 0 can be returned if an operation is not supported, if an error occurred, if EOF has not been reached and in the case of **BIO_seek()** on a file BIO for a successful operation.

In older versions of OpenSSL the **BIO_ctrl_pending()** and **BIO_ctrl_wpending()** could return values greater than **INT_MAX** on error.

HISTORY

The **BIO_get_ktls_send()** and **BIO_get_ktls_recv()** macros were added in OpenSSL 3.0. They were modified to never return -1 in OpenSSL 3.0.4.

COPYRIGHT

Copyright 2000-2022 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>.