NAME

bio − Basic I/O abstraction

SYNOPSIS

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

DESCRIPTION

A BIO is an I/O abstraction, it hides many of the underlying I/O details from an application. If an application uses a BIO for its I/O it can transparently handle SSL connections, unencrypted network connections and file I/O.

There are two types of BIO, a source/sink BIO and a filter BIO.

As its name implies a source/sink BIO is a source and/or sink of data, examples include a socket BIO and a file BIO.

A filter BIO takes data from one BIO and passes it through to another, or the application. The data may be left unmodified (for example a message digest BIO) or translated (for example an encryption BIO). The effect of a filter BIO may change according to the I/O operation it is performing: for example an encryption BIO will encrypt data if it is being written to and decrypt data if it is being read from.

BIOs can be joined together to form a chain (a single BIO is a chain with one component). A chain normally consists of one source/sink BIO and one or more filter BIOs. Data read from or written to the first BIO then traverses the chain to the end (normally a source/sink BIO).

Some BIOs (such as memory BIOs) can be used immediately after calling **BIO_new()**. Others (such as file BIOs) need some additional initialization, and frequently a utility function exists to create and initialize such BIOs.

If **BIO_free()** is called on a BIO chain it will only free one BIO resulting in a memory leak.

Calling **BIO_free_all()** on a single BIO has the same effect as calling **BIO_free()** on it other than the discarded return value.

Normally the *type* argument is supplied by a function which returns a pointer to a BIO_METHOD. There is a naming convention for such functions: a source/sink BIO typically starts with *BIO_s_* and a filter BIO with *BIO_f_*.

TCP Fast Open

TCP Fast Open (RFC7413), abbreviated "TFO", is supported by the BIO interface since OpenSSL 3.2. TFO is supported in the following operating systems:

- Linux kernel 3.13 and later, where TFO is enabled by default.
- Linux kernel 4.11 and later, using TCP_FASTOPEN_CONNECT.
- FreeBSD 10.3 to 11.4, supports server TFO only.
- FreeBSD 12.0 and later, supports both client and server TFO.
- macOS 10.14 and later.

Each operating system has a slightly different API for TFO. Please refer to the operating systems' API documentation when using sockets directly.

EXAMPLES

Create a memory BIO:

```
BIO *mem = BIO_new(BIO_s_mem());
```

SEE ALSO

BIO_ctrl (3), **BIO_f_base64** (3), **BIO_f_buffer** (3), **BIO_f_cipher** (3), **BIO_f_md** (3), **BIO_f_null** (3), **BIO_f_ssl** (3), **BIO_f_readbuffer** (3), **BIO_find_type** (3), **BIO_get_conn_mode** (3), **BIO_new** (3), **BIO_new_bio_pair** (3), **BIO_push** (3), **BIO_read_ex** (3), **BIO_s_accept** (3), **BIO_s_bio** (3), **BIO_s_connect** (3), **BIO_s_fd** (3), **BIO_s_file** (3), **BIO_s_mem** (3), **BIO_s_null** (3), **BIO_s_socket** (3), **BIO_set_callback** (3), **BIO_set_conn_mode** (3), **BIO_set_tfo** (3), **BIO_set_tfo_accept** (3),

BIO_should_retry (3)

COPYRIGHT