Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'BIO_meth_get_read.3ossl' command

$ man BIO_meth_get_read.3ossl

BIO_METH_NEW(3ossl) OpenSSL BIO_METH_NEW(3ossl)

NAME

BIO_get_new_index, BIO_meth_new, BIO_meth_free, BIO_meth_get_read_ex,
BIO_meth_set_read_ex, BIO_meth_get_write_ex, BIO_meth_set_write_ex,
BIO_meth_get_write, BIO_meth_set_write, BIO_meth_get_read,
BIO_meth_set_read, BIO_meth_get_puts, BIO_meth_set_puts,
BIO_meth_get_gets, BIO_meth_set_gets, BIO_meth_get_ctrl,
BIO_meth_set_ctrl, BIO_meth_get_create, BIO_meth_set_create,
BIO_meth_get_destroy, BIO_meth_set_destroy, BIO_meth_get_callback_ctrl,
BIO_meth_set_callback_ctrl - Routines to build up BIO methods

SYNOPSIS

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

int BIO_get_new_index(void);

BIO_METHOD *BIO_meth_new(int type, const char *name);

void BIO_meth_free(BIO_METHOD *biom);

int (*BIO_meth_get_write_ex(const BIO_METHOD *biom))(BIO *, const char *, size_t,
                                                     size_t *);
```

```
int (*BIO_meth_get_write(const BIO_METHOD *biom))(BIO *, const char *, int);
int BIO_meth_set_write_ex(BIO_METHOD *biom,
                          int (*bwrite)(BIO *, const char *, size_t, size_t *));
int BIO_meth_set_write(BIO_METHOD *biom,
                       int (*write)(BIO *, const char *, int));

int (*BIO_meth_get_read_ex(const BIO_METHOD *biom))(BIO *, char *, size_t, size_t *);
int (*BIO_meth_get_read(const BIO_METHOD *biom))(BIO *, char *, int);
int BIO_meth_set_read_ex(BIO_METHOD *biom,
                         int (*bread)(BIO *, char *, size_t, size_t *));
int BIO_meth_set_read(BIO_METHOD *biom, int (*read)(BIO *, char *, int));

int (*BIO_meth_get_puts(const BIO_METHOD *biom))(BIO *, const char *);
int BIO_meth_set_puts(BIO_METHOD *biom, int (*puts)(BIO *, const char *));

int (*BIO_meth_get_gets(const BIO_METHOD *biom))(BIO *, char *, int);
int BIO_meth_set_gets(BIO_METHOD *biom,
                      int (*gets)(BIO *, char *, int));

long (*BIO_meth_get_ctrl(const BIO_METHOD *biom))(BIO *, int, long, void *);
int BIO_meth_set_ctrl(BIO_METHOD *biom,
                      long (*ctrl)(BIO *, int, long, void *));

int (*BIO_meth_get_create(const BIO_METHOD *bion))(BIO *);
int BIO_meth_set_create(BIO_METHOD *biom, int (*create)(BIO *));

int (*BIO_meth_get_destroy(const BIO_METHOD *biom))(BIO *);
int BIO_meth_set_destroy(BIO_METHOD *biom, int (*destroy)(BIO *));

long (*BIO_meth_get_callback_ctrl(const BIO_METHOD *biom))(BIO *, int, BIO_info_cb *);
int BIO_meth_set_callback_ctrl(BIO_METHOD *biom,
                               long (*callback_ctrl)(BIO *, int, BIO_info_cb *));
```

DESCRIPTION

The BIO_METHOD type is a structure used for the implementation of new BIO types. It provides a set of functions used by OpenSSL for the implementation of the various BIO capabilities. See the bio(7) page for more information.

BIO_meth_new() creates a new BIO_METHOD structure. It should be given a unique integer type and a string that represents its name. Use BIO_get_new_index() to get the value for type.

The set of standard OpenSSL provided BIO types is provided in <openssl/bio.h>. Some examples include BIO_TYPE_BUFFER and BIO_TYPE_CIPHER. Filter BIOs should have a type which have the "filter" bit set (BIO_TYPE_FILTER). Source/sink BIOs should have the "source/sink" bit set (BIO_TYPE_SOURCE_SINK). File descriptor based BIOs (e.g. socket, fd, connect, accept etc) should additionally have the "descriptor" bit set (BIO_TYPE_DESCRIPTOR). See the BIO_find_type(3) page for more information.

BIO_meth_free() destroys a BIO_METHOD structure and frees up any memory associated with it.

BIO_meth_get_write_ex() and BIO_meth_set_write_ex() get and set the function used for writing arbitrary length data to the BIO respectively. This function will be called in response to the application calling BIO_write_ex() or BIO_write(). The parameters for the function have the same meaning as for BIO_write_ex(). Older code may call BIO_meth_get_write() and BIO_meth_set_write() instead. Applications should not call both BIO_meth_set_write_ex() and BIO_meth_set_write() or call BIO_meth_get_write() when the function was set with BIO_meth_set_write_ex().

BIO_meth_get_read_ex() and BIO_meth_set_read_ex() get and set the

function used for reading arbitrary length data from the BIO respectively. This function will be called in response to the application calling BIO_read_ex() or BIO_read(). The parameters for the function have the same meaning as for BIO_read_ex(). Older code may call BIO_meth_get_read() and BIO_meth_set_read() instead. Applications should not call both BIO_meth_set_read_ex() and BIO_meth_set_read() or call BIO_meth_get_read() when the function was set with BIO_meth_set_read_ex().

BIO_meth_get_puts() and BIO_meth_set_puts() get and set the function used for writing a NULL terminated string to the BIO respectively. This function will be called in response to the application calling BIO_puts(). The parameters for the function have the same meaning as for BIO_puts().

BIO_meth_get_gets() and BIO_meth_set_gets() get and set the function typically used for reading a line of data from the BIO respectively (see the BIO_gets(3) page for more information). This function will be called in response to the application calling BIO_gets(). The parameters for the function have the same meaning as for BIO_gets().

BIO_meth_get_ctrl() and BIO_meth_set_ctrl() get and set the function used for processing ctrl messages in the BIO respectively. See the BIO_ctrl(3) page for more information. This function will be called in response to the application calling BIO_ctrl(). The parameters for the function have the same meaning as for BIO_ctrl().

BIO_meth_get_create() and BIO_meth_set_create() get and set the function used for creating a new instance of the BIO respectively. This function will be called in response to the application calling BIO_new() and passing in a pointer to the current BIO_METHOD. The BIO_new() function will allocate the memory for the new BIO, and a pointer to this newly allocated structure will be passed as a parameter

to the function. If a create function is set, BIO_new() will not mark the BIO as initialised on allocation. BIO_set_init(3) must then be called either by the create function, or later, by a BIO ctrl function, once BIO initialisation is complete.

BIO_meth_get_destroy() and BIO_meth_set_destroy() get and set the function used for destroying an instance of a BIO respectively. This function will be called in response to the application calling BIO_free(). A pointer to the BIO to be destroyed is passed as a parameter. The destroy function should be used for BIO specific clean up. The memory for the BIO itself should not be freed by this function.

BIO_meth_get_callback_ctrl() and BIO_meth_set_callback_ctrl() get and set the function used for processing callback ctrl messages in the BIO respectively. See the BIO_callback_ctrl(3) page for more information. This function will be called in response to the application calling BIO_callback_ctrl(). The parameters for the function have the same meaning as for BIO_callback_ctrl().

RETURN VALUES

BIO_get_new_index() returns the new BIO type value or -1 if an error occurred.

BIO_meth_new(int type, const char *name) returns a valid BIO_METHOD or NULL if an error occurred.

The BIO_meth_set functions return 1 on success or 0 on error.

The BIO_meth_get functions return the corresponding function pointers.

SEE ALSO

bio(7), BIO_find_type(3), BIO_ctrl(3), BIO_read_ex(3), BIO_new(3)

HISTORY

The functions described here were added in OpenSSL 1.1.0.

COPYRIGHT