

**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

Copyright 2016-2022 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>.