NAME

BIO_s_connect, BIO_new_connect, BIO_set_conn_hostname, BIO_set_conn_port, BIO_set_conn_ip, BIO_set_conn_int_port, BIO_get_conn_hostname, BIO_get_conn_port, BIO_get_conn_ip, BIO_get_conn_int_port, BIO_set_nbio, BIO_do_connect - connect BIO

SYNOPSIS

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

BIO_METHOD * BIO_s_connect(void);

BIO *BIO_new_connect(char *name);

long BIO_set_conn_hostname(BIO *b, char *name);
long BIO_set_conn_port(BIO *b, char *port);
long BIO_set_conn_ip(BIO *b, char *ip);
long BIO_set_conn_int_port(BIO *b, char *port);
char *BIO_get_conn_hostname(BIO *b);
char *BIO_get_conn_port(BIO *b);
char *BIO_get_conn_ip(BIO *b, dummy);
long BIO_get_conn_int_port(BIO *b, int port);

long BIO_set_nbio(BIO *b, long n);

int BIO_do_connect(BIO *b);
```

DESCRIPTION

BIO_s_connect() returns the connect BIO method. This is a wrapper round the platform's TCP/IP socket connection routines.

Using connect BIOs, TCP/IP connections can be made and data transferred using only BIO routines. In this way any platform specific operations are hidden by the BIO abstraction.

Read and write operations on a connect BIO will perform I/O on the underlying connection. If no connection is established and the port and hostname (see below) is set up properly then a connection is established first.

Connect BIOs support *BIO_puts()* but not *BIO_gets()*.

If the close flag is set on a connect BIO then any active connection is shutdown and the socket closed when the BIO is freed.

Calling *BIO_reset()* on a connect BIO will close any active connection and reset the BIO into a state where it can connect to the same host again.

BIO_get_fd() places the underlying socket in **c** if it is not NULL, it also returns the socket . If **c** is not NULL it should be of type (int *).

BIO_set_conn_hostname() uses the string **name** to set the hostname. The hostname can be an IP address. The hostname can also include the port in the form hostname:port . It is also acceptable to use the form ''hostname/any/other/path'' or ''hostname:port/any/other/path''.

BIO_set_conn_port() sets the port to **port**. **port** can be the numerical form or a string such as ''http''. A string will be looked up first using *getservbyname()* on the host platform but if that fails a standard table of port names will be used. Currently the list is http, telnet, socks, https, ssl, ftp, gopher and wais.

BIO_set_conn_ip() sets the IP address to **ip** using binary form, that is four bytes specifying the IP address in big-endian form.

BIO_set_conn_int_port() sets the port using **port**. **port** should be of type (int *).

BIO_get_conn_hostname() returns the hostname of the connect BIO or NULL if the BIO is

initialized but no hostname is set. This return value is an internal pointer which should not be modified.

BIO_get_conn_port() returns the port as a string.

BIO_get_conn_ip() returns the IP address in binary form.

BIO_get_conn_int_port() returns the port as an int.

BIO_set_nbio() sets the non blocking I/O flag to **n**. If **n** is zero then blocking I/O is set. If **n** is 1 then non blocking I/O is set. Blocking I/O is the default. The call to *BIO_set_nbio()* should be made before the connection is established because non blocking I/O is set during the connect process.

BIO_new_connect() combines *BIO_new()* and *BIO_set_conn_hostname()* into a single call: that is it creates a new connect BIO with **name**.

BIO_do_connect() attempts to connect the supplied BIO. It returns 1 if the connection was established successfully. A zero or negative value is returned if the connection could not be established, the call *BIO_should_retry()* should be used for non blocking connect BIOs to determine if the call should be retried.

NOTES

If blocking I/O is set then a non positive return value from any I/O call is caused by an error condition, although a zero return will normally mean that the connection was closed.

If the port name is supplied as part of the host name then this will override any value set with *BIO_set_conn_port()*. This may be undesirable if the application does not wish to allow connection to arbitrary ports. This can be avoided by checking for the presence of the ':' character in the passed hostname and either indicating an error or truncating the string at that point.

The values returned by *BIO_get_conn_hostname()*, *BIO_get_conn_port()*, *BIO_get_conn_ip()* and *BIO_get_conn_int_port()* are updated when a connection attempt is made. Before any connection attempt the values returned are those set by the application itself.

Applications do not have to call *BIO_do_connect()* but may wish to do so to separate the connection process from other I/O processing.

If non blocking I/O is set then retries will be requested as appropriate.

It addition to *BIO_should_read()* and *BIO_should_write()* it is also possible for *BIO_should_io_special()* to be true during the initial connection process with the reason BIO_RR_CONNECT. If this is returned then this is an indication that a connection attempt would block, the application should then take appropriate action to wait until the underlying socket has connected and retry the call.

BIO_set_conn_hostname(), *BIO_set_conn_port()*, *BIO_set_conn_ip()*, *BIO_set_conn_int_port()*, *BIO_get_conn_hostname()*, *BIO_get_conn_port()*, *BIO_get_conn_ip()*, *BIO_get_conn_int_port()*, *BIO_set_nbio()* and *BIO_do_connect()* are macros.

RETURN VALUES

BIO_s_connect() returns the connect BIO method.

BIO_get_fd() returns the socket or -1 if the BIO has not been initialized.

BIO_set_conn_hostname(), *BIO_set_conn_port()*, *BIO_set_conn_ip()* and *BIO_set_conn_int_port()* always return 1.

BIO_get_conn_hostname() returns the connected hostname or NULL is none was set.

BIO_get_conn_port() returns a string representing the connected port or NULL if not set.

BIO_get_conn_ip() returns a pointer to the connected IP address in binary form or all zeros if not set.

BIO_get_conn_int_port() returns the connected port or 0 if none was set.

BIO_set_nbio() always returns 1.

BIO_do_connect() returns 1 if the connection was successfully established and 0 or -1 if the connection failed.

EXAMPLE

This is example connects to a webserver on the local host and attempts to retrieve a page and copy the result to standard output.

```
BIO *cbio, *out;
int len;
char tmpbuf[1024];
ERR_load_crypto_strings();
cbio = BIO_new_connect("localhost:http");
out = BIO_new_fp(stdout, BIO_NOCLOSE);
if(BIO_do_connect(cbio) <= 0) {
fprintf(stderr, "Error connecting to server\n");
ERR_print_errors_fp(stderr);
/* whatever ... */
}
BIO_puts(cbio, "GET / HTTP/1.0\n\n");
for(;;) {
len = BIO_read(cbio, tmpbuf, 1024);
if(len <= 0) break;
BIO_write(out, tmpbuf, len);
}
BIO_free(cbio);
BIO_free(out);
```

SEE ALSO

TBA