NAME

BIO_lookup_type, BIO_ADDRINFO, BIO_ADDRINFO_next, BIO_ADDRINFO_free, BIO_ADDRINFO_family, BIO_ADDRINFO_socktype, BIO_ADDRINFO_protocol, BIO_ADDRINFO_address, BIO_lookup_ex, BIO_lookup - BIO_ADDRINFO type and routines

SYNOPSIS

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

typedef union bio_addrinfo_st BIO_ADDRINFO;

enum BIO_lookup_type {
    BIO_LOOKUP_CLIENT, BIO_LOOKUP_SERVER
};

int BIO_lookup_ex(const char *host, const char *service, int lookup_type,
                  int family, int socktype, int protocol, BIO_ADDRINFO **res);
int BIO_lookup(const char *host, const char *service,
               enum BIO_lookup_type lookup_type,
               int family, int socktype, BIO_ADDRINFO **res);

const BIO_ADDRINFO *BIO_ADDRINFO_next(const BIO_ADDRINFO *bai);
int BIO_ADDRINFO_family(const BIO_ADDRINFO *bai);
int BIO_ADDRINFO_socktype(const BIO_ADDRINFO *bai);
int BIO_ADDRINFO_protocol(const BIO_ADDRINFO *bai);
const BIO_ADDR *BIO_ADDRINFO_address(const BIO_ADDRINFO *bai);
void BIO_ADDRINFO_free(BIO_ADDRINFO *bai);
```

DESCRIPTION

The **BIO_ADDRINFO** type is a wrapper for address information types provided on your platform.

BIO_ADDRINFO normally forms a chain of several that can be picked at one by one.

BIO_lookup_ex() looks up a specified **host** and **service**, and uses **lookup_type** to determine what the default address should be if **host** is **NULL**. **family**, **socktype** and **protocol** are used to determine what protocol family, socket type and protocol should be used for the lookup. **family** can be any of AF_INET, AF_INET6, AF_UNIX and AF_UNSPEC. **socktype** can be SOCK_STREAM, SOCK_DGRAM or 0. Specifying 0 indicates that any type can be used. **protocol** specifies a protocol such as IPPROTO_TCP, IPPROTO_UDP or IPPORTO_SCTP. If set to 0 than any protocol can be used. **res** points at a pointer to hold the start of a **BIO_ADDRINFO** chain.

For the family **AF_UNIX**, **BIO_lookup_ex()** will ignore the **service** parameter and expects the **host** parameter to hold the path to the socket file.

BIO_lookup() does the same as **BIO_lookup_ex()** but does not provide the ability to select based on the protocol (any protocol may be returned).

BIO_ADDRINFO_family() returns the family of the given **BIO_ADDRINFO**. The result will be one of the constants AF_INET, AF_INET6 and AF_UNIX.

BIO_ADDRINFO_socktype() returns the socket type of the given **BIO_ADDRINFO**. The result will be one of the constants SOCK_STREAM and SOCK_DGRAM.

BIO_ADDRINFO_protocol() returns the protocol id of the given **BIO_ADDRINFO**. The result will be one of the constants IPPROTO_TCP and IPPROTO_UDP.

BIO_ADDRINFO_address() returns the underlying **BIO_ADDR** of the given **BIO_ADDRINFO**.

BIO_ADDRINFO_next() returns the next **BIO_ADDRINFO** in the chain from the given one.

BIO_ADDRINFO_free() frees the chain of **BIO_ADDRINFO** starting with the given one.

RETURN VALUES

BIO_lookup_ex() and **BIO_lookup()** return 1 on success and 0 when an error occurred, and will leave an error indication on the OpenSSL error stack in that case.

All other functions described here return 0 or **NULL** when the information they should return isn't available.

NOTES

The **BIO_lookup_ex()** implementation uses the platform provided **getaddrinfo()** function. On Linux it is known that specifying 0 for the protocol will not return any SCTP based addresses when calling **getaddrinfo()**. Therefore, if an SCTP address is required then the **protocol** parameter to **BIO_lookup_ex()** should be explicitly set to IPPROTO_SCTP. The same may be true on other platforms.

HISTORY

The **BIO_lookup_ex()** function was added in OpenSSL 1.1.1.

COPYRIGHT