

**NAME**

`ne_iaddr_make`, `ne_iaddr_cmp`, `ne_iaddr_print`, `ne_iaddr_typeof`, `ne_iaddr_parse`, `ne_iaddr_raw`, `ne_iaddr_reverse`, `ne_iaddr_free` - functions to manipulate network addresses

**SYNOPSIS**

```
#include <ne_socket.h>
```

```
typedef enum {  
    ne_iaddr_ipv4 = 0,  
    ne_iaddr_ipv6  
} ne_iaddr_type;
```

```
ne_inet_addr *ne_iaddr_make(ne_iaddr_type type, const unsigned char *raw);
```

```
int ne_iaddr_cmp(const ne_inet_addr *ia1, const ne_inet_addr *ia2);
```

```
char *ne_iaddr_print(const ne_inet_addr *ia, char *buffer, size_t bufsiz);
```

```
ne_iaddr_type ne_iaddr_typeof(const ne_inet_addr *ia);
```

```
ne_inet_addr *ne_iaddr_parse(const char *address, ne_iaddr_type type);
```

```
unsigned char *ne_iaddr_raw(const ne_inet_addr *ia, unsigned char *buffer);
```

```
int ne_iaddr_reverse(const ne_inet_addr *ia, char *buffer, size_t buflen);
```

```
void ne_iaddr_free(const ne_inet_addr *ia);
```

**DESCRIPTION**

**ne\_iaddr\_make** creates an **ne\_inet\_addr** object from a raw binary network address; for instance the four bytes `0x7f 0x00 0x00 0x01` represent the IPv4 address `127.0.0.1`. The object returned is suitable for passing to **ne\_sock\_connect**. A binary IPv4 address contains four bytes; a binary IPv6 address contains sixteen bytes; addresses passed must be in network byte order.

**ne\_iaddr\_cmp** compares two network address objects; returning zero only if they are identical. The objects need not have the same address type; if the addresses are not of the same type, the return value is guaranteed to be non-zero.

**ne\_iaddr\_print** prints a human-readable string representation of a network address into a buffer, for instance the string `"127.0.0.1"`.

**ne\_iaddr\_typeof** returns the type of the given network address object.

**ne\_iaddr\_parse** parses a string representation of a network address (such as "127.0.0.1" and creates a network address object to represent the parsed address.

**ne\_iaddr\_raw** writes the raw byte representation of a network address to the provided buffer. The bytes are written in network byte order; the buffer must be of suitable length for the type of address (4 bytes for an IPv4 address, 16 bytes for an IPv6 address).

**ne\_iaddr\_reverse** performs a reverse name lookup on the address object, writing the (first) hostname associated with the IP address to the provided buffer. If the hostname is longer than the buffer it will be silently truncated; on success the string written to the buffer is always NUL-terminated.

**ne\_iaddr\_free** releases the memory associated with a network address object.

## RETURN VALUE

**ne\_iaddr\_make** returns NULL if the address type passed is not supported (for instance on a platform which does not support IPv6).

**ne\_iaddr\_print** returns the *buffer* pointer, and never NULL.

**ne\_iaddr\_parse** returns a network address object on success, or NULL on failure to parse the *address* parameter.

**ne\_iaddr\_reverse** returns zero on success or non-zero if no hostname is associated with the address.

**ne\_iaddr\_raw** returns the *buffer* parameter, and never NULL.

## EXAMPLES

The following example connects a socket to port 80 at the address 127.0.0.1.

```
unsigned char addr[] = "\0x7f\0x00\0x00\0x01";
ne_inet_addr *ia;

ia = ne_iaddr_make(ne_iaddr_ipv4, addr);
if (ia != NULL) {
    ne_socket *sock = ne_sock_connect(ia, 80);
    ne_iaddr_free(ia);
    /* ... */
} else {
```

```
    /* ... */  
}
```

**SEE ALSO**

ne\_addr\_resolve

**AUTHOR**

**Joe Orton**

Author.

**COPYRIGHT**