NAME

BN_bn2binpad, BN_bn2bin, BN_bin2bn, BN_bn2lebinpad, BN_lebin2bn, BN_bn2hex, BN_bn2dec, BN_hex2bn, BN_dec2bn, BN_print, BN_print_fp, BN_bn2mpi, BN_mpi2bn − format conversions

SYNOPSIS

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

int BN_bn2bin(const BIGNUM *a, unsigned char *to);
int BN_bn2binpad(const BIGNUM *a, unsigned char *to, int tolen);
BIGNUM *BN_bin2bn(const unsigned char *s, int len, BIGNUM *ret);

int BN_bn2lebinpad(const BIGNUM *a, unsigned char *to, int tolen);
BIGNUM *BN_lebin2bn(const unsigned char *s, int len, BIGNUM *ret);

char *BN_bn2hex(const BIGNUM *a);
char *BN_bn2dec(const BIGNUM *a);
int BN_hex2bn(BIGNUM **a, const char *str);
int BN_dec2bn(BIGNUM **a, const char *str);

int BN_print(BIO *fp, const BIGNUM *a);
int BN_print_fp(FILE *fp, const BIGNUM *a);

int BN_bn2mpi(const BIGNUM *a, unsigned char *to);
BIGNUM *BN_mpi2bn(unsigned char *s, int len, BIGNUM *ret);
```

DESCRIPTION

BN_bn2bin() converts the absolute value of **a** into big-endian form and stores it at **to**. **to** must point to BN_num_bytes(**a**) bytes of memory.

BN_bn2binpad() also converts the absolute value of **a** into big-endian form and stores it at **to**. **tolen** indicates the length of the output buffer **to**. The result is padded with zeros if necessary. If **tolen** is less than BN_num_bytes(**a**) an error is returned.

BN_bin2bn() converts the positive integer in big-endian form of length **len** at **s** into a **BIGNUM** and places it in **ret**. If **ret** is NULL, a new **BIGNUM** is created.

BN_bn2lebinpad() and **BN_lebin2bn()** are identical to **BN_bn2binpad()** and **BN_bin2bn()** except the buffer is in little-endian format.

BN_bn2hex() and **BN_bn2dec()** return printable strings containing the hexadecimal and decimal encoding of **a** respectively. For negative numbers, the string is prefaced with a leading '−'. The string must be freed later using **OPENSSL_free()**.

BN_hex2bn() takes as many characters as possible from the string **str**, including the leading character '−' which means negative, to form a valid hexadecimal number representation and converts them to a **BIGNUM** and stores it in ***a**. If ***a** is NULL, a new **BIGNUM** is created. If **a** is NULL, it only computes the length of valid representation. A "negative zero" is converted to zero. **BN_dec2bn()** is the same using the decimal system.

BN_print() and **BN_print_fp()** write the hexadecimal encoding of **a**, with a leading '−' for negative numbers, to the **BIO** or **FILE fp**.

BN_bn2mpi() and **BN_mpi2bn()** convert **BIGNUM**s from and to a format that consists of the number's length in bytes represented as a 4−byte big-endian number, and the number itself in big-endian format, where the most significant bit signals a negative number (the representation of numbers with the MSB set is prefixed with null byte).

BN_bn2mpi() stores the representation of **a** at **to**, where **to** must be large enough to hold the result. The size can be determined by calling BN_bn2mpi(**a**, NULL).

BN_mpi2bn() converts the **len** bytes long representation at **s** to a **BIGNUM** and stores it at **ret**, or in a

newly allocated **BIGNUM** if **ret** is NULL.

RETURN VALUES

BN_bn2bin() returns the length of the big-endian number placed at **to**. **BN_bin2bn()** returns the **BIGNUM**, NULL on error.

BN_bn2binpad() returns the number of bytes written or −1 if the supplied buffer is too small.

BN_bn2hex() and **BN_bn2dec()** return a null-terminated string, or NULL on error. **BN_hex2bn()** and **BN_dec2bn()** return the number of characters used in parsing, or 0 on error, in which case no new **BIGNUM** will be created.

BN_print_fp() and **BN_print()** return 1 on success, 0 on write errors.

BN_bn2mpi() returns the length of the representation. **BN_mpi2bn()** returns the **BIGNUM**, and NULL on error.

The error codes can be obtained by **ERR_get_error(3)**.

SEE ALSO

ERR_get_error(3), **BN_zero(3)**, **ASN1_INTEGER_to_BN(3)**, **BN_num_bytes(3)**

COPYRIGHT