



*Full credit is given to the above companies including the OS that this PDF file was generated!*

## **Red Hat Enterprise Linux Release 9.2 Manual Pages on 'BN\_rand\_range\_ex.3ossl' command**

**\$ man BN\_rand\_range\_ex.3ossl**

BN\_RAND(3ossl)                    OpenSSL                    BN\_RAND(3ossl)

### NAME

BN\_rand\_ex, BN\_rand, BN\_priv\_rand\_ex, BN\_priv\_rand, BN\_pseudo\_rand,  
BN\_rand\_range\_ex, BN\_rand\_range, BN\_priv\_rand\_range\_ex,  
BN\_priv\_rand\_range, BN\_pseudo\_rand\_range - generate pseudo-random  
number

### SYNOPSIS

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

int BN_rand_ex(BIGNUM *rnd, int bits, int top, int bottom,
               unsigned int strength, BN_CTX *ctx);

int BN_rand(BIGNUM *rnd, int bits, int top, int bottom);

int BN_priv_rand_ex(BIGNUM *rnd, int bits, int top, int bottom,
                   unsigned int strength, BN_CTX *ctx);

int BN_priv_rand(BIGNUM *rnd, int bits, int top, int bottom);

int BN_rand_range_ex(BIGNUM *rnd, const BIGNUM *range, unsigned int strength,
                    BN_CTX *ctx);

int BN_rand_range(BIGNUM *rnd, const BIGNUM *range);

int BN_priv_rand_range_ex(BIGNUM *rnd, const BIGNUM *range, unsigned int strength,
                          BN_CTX *ctx);

int BN_priv_rand_range(BIGNUM *rnd, const BIGNUM *range);
```

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining OPENSSL\_API\_COMPAT with a suitable version value, see openssl\_user\_macros(7):

```
int BN_pseudo_rand(BIGNUM *rnd, int bits, int top, int bottom);  
int BN_pseudo_rand_range(BIGNUM *rnd, const BIGNUM *range);
```

## DESCRIPTION

`BN_rand_ex()` generates a cryptographically strong pseudo-random number of bits in length and security strength at least strength bits using the random number generator for the library context associated with `ctx`. The function stores the generated data in `rnd`. The parameter `ctx` may be `NULL` in which case the default library context is used. If `bits` is less than zero, or too small to accommodate the requirements specified by the `top` and `bottom` parameters, an error is returned. The `top` parameters specifies requirements on the most significant bit of the generated number. If it is `BN RAND TOP ANY`, there is no constraint. If it is `BN RAND TOP ONE`, the top bit must be one. If it is `BN RAND TOP TWO`, the two most significant bits of the number will be set to 1, so that the product of two such random numbers will always have  $2 * \text{bits}$  length. If `bottom` is `BN RAND BOTTOM ODD`, the number will be odd; if it is `BN RAND BOTTOM ANY` it can be odd or even. If `bits` is 1 then `top` cannot also be `BN RAND TOP TWO`.

`BN_rand()` is the same as `BN_rand_ex()` except that the default library context is always used.

`BN_rand_range_ex()` generates a cryptographically strong pseudo-random number `rnd`, of security strength at least strength bits, in the range  $0 \leq \text{rnd} < \text{range}$  using the random number generator for the library context associated with `ctx`. The parameter `ctx` may be `NULL` in which case the default library context is used.

`BN_rand_range()` is the same as `BN_rand_range_ex()` except that the default library context is always used.

`BN_priv_rand_ex()`, `BN_priv_rand()`, `BN_priv_rand_range_ex()` and `BN_priv_rand_range()` have the same semantics as `BN_rand_ex()`, `BN_rand()`, `BN_rand_range_ex()` and `BN_rand_range()` respectively. They are intended to be used for generating values that should remain private, and mirror the same difference between `RAND_bytes(3)` and `RAND_priv_bytes(3)`.

## NOTES

Always check the error return value of these functions and do not take randomness for granted: an error occurs if the CSPRNG has not been seeded with enough randomness to ensure an unpredictable byte sequence.

## RETURN VALUES

The functions return 1 on success, 0 on error. The error codes can be obtained by `ERR_get_error(3)`.

## SEE ALSO

`ERR_get_error(3)`, `RAND_add(3)`, `RAND_bytes(3)`, `RAND_priv_bytes(3)`, `RAND(7)`, `EVP RAND(7)`

## HISTORY

? Starting with OpenSSL release 1.1.0, `BN_pseudo_rand()` has been identical to `BN_rand()` and `BN_pseudo_rand_range()` has been identical to `BN_rand_range()`. The `BN_pseudo_rand()` and `BN_pseudo_rand_range()` functions were deprecated in OpenSSL 3.0.

? The `BN_priv_rand()` and `BN_priv_rand_range()` functions were added in OpenSSL 1.1.1.

? The `BN_rand_ex()`, `BN_priv_rand_ex()`, `BN_rand_range_ex()` and `BN_priv_rand_range_ex()` functions were added in OpenSSL 3.0.

## COPYRIGHT

Copyright 2000-2021 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>.

3.0.7

2023-07-13

BN\_RAND(3openssl)