

NAME

ASN1_INTEGER_get_uint64, ASN1_INTEGER_set_uint64, ASN1_INTEGER_get_int64,
 ASN1_INTEGER_get, ASN1_INTEGER_set_int64, ASN1_INTEGER_set, BN_to_ASN1_INTEGER,
 ASN1_INTEGER_to_BN, ASN1_ENUMERATED_get_int64, ASN1_ENUMERATED_get,
 ASN1_ENUMERATED_set_int64, ASN1_ENUMERATED_set, BN_to_ASN1_ENUMERATED,
 ASN1_ENUMERATED_to_BN – ASN.1 INTEGER and ENUMERATED utilities

SYNOPSIS

```
#include <openssl/asn1.h>
```

```
int ASN1_INTEGER_get_int64(int64_t *pr, const ASN1_INTEGER *a);
long ASN1_INTEGER_get(const ASN1_INTEGER *a);
```

```
int ASN1_INTEGER_set_int64(ASN1_INTEGER *a, int64_t r);
int ASN1_INTEGER_set(ASN1_INTEGER *a, long v);
```

```
int ASN1_INTEGER_get_uint64(uint64_t *pr, const ASN1_INTEGER *a);
int ASN1_INTEGER_set_uint64(ASN1_INTEGER *a, uint64_t r);
```

```
ASN1_INTEGER *BN_to_ASN1_INTEGER(const BIGNUM *bn, ASN1_INTEGER *ai);
BIGNUM *ASN1_INTEGER_to_BN(const ASN1_INTEGER *ai, BIGNUM *bn);
```

```
int ASN1_ENUMERATED_get_int64(int64_t *pr, const ASN1_ENUMERATED *a);
long ASN1_ENUMERATED_get(const ASN1_ENUMERATED *a);
```

```
int ASN1_ENUMERATED_set_int64(ASN1_ENUMERATED *a, int64_t r);
int ASN1_ENUMERATED_set(ASN1_ENUMERATED *a, long v);
```

```
ASN1_ENUMERATED *BN_to_ASN1_ENUMERATED(const BIGNUM *bn, ASN1_ENUMERATED *ai);
BIGNUM *ASN1_ENUMERATED_to_BN(const ASN1_ENUMERATED *ai, BIGNUM *bn);
```

DESCRIPTION

These functions convert to and from **ASN1_INTEGER** and **ASN1_ENUMERATED** structures.

ASN1_INTEGER_get_int64() converts an **ASN1_INTEGER** into an **int64_t** type. If successful it returns 1 and sets **pr* to the value of *a*. If it fails (due to invalid type or the value being too big to fit into an **int64_t** type) it returns 0.

ASN1_INTEGER_get_uint64() is similar to **ASN1_INTEGER_get_int64_t()** except it converts to a **uint64_t** type and an error is returned if the passed integer is negative.

ASN1_INTEGER_get() also returns the value of *a* but it returns 0 if *a* is NULL and -1 on error (which is ambiguous because -1 is a legitimate value for an **ASN1_INTEGER**). New applications should use **ASN1_INTEGER_get_int64()** instead.

ASN1_INTEGER_set_int64() sets the value of **ASN1_INTEGER** *a* to the **int64_t** value *r*.

ASN1_INTEGER_set_uint64() sets the value of **ASN1_INTEGER** *a* to the **uint64_t** value *r*.

ASN1_INTEGER_set() sets the value of **ASN1_INTEGER** *a* to the *long* value *v*.

BN_to_ASN1_INTEGER() converts **BIGNUM** *bn* to an **ASN1_INTEGER**. If *ai* is NULL a new **ASN1_INTEGER** structure is returned. If *ai* is not NULL then the existing structure will be used instead.

ASN1_INTEGER_to_BN() converts **ASN1_INTEGER** *ai* into a **BIGNUM**. If *bn* is NULL a new **BIGNUM** structure is returned. If *bn* is not NULL then the existing structure will be used instead.

ASN1_ENUMERATED_get_int64(), **ASN1_ENUMERATED_set_int64()**,
ASN1_ENUMERATED_set(), **BN_to_ASN1_ENUMERATED()** and **ASN1_ENUMERATED_to_BN()**
 behave in an identical way to their **ASN1_INTEGER** counterparts except they operate on an **ASN1_ENUMERATED** value.

ASN1_ENUMERATED_get() returns the value of *a* in a similar way to **ASN1_INTEGER_get()** but it returns **0xffffffffL** if the value of *a* will not fit in a long type. New applications should use **ASN1_ENUMERATED_get_int64()** instead.

NOTES

In general an **ASN1_INTEGER** or **ASN1_ENUMERATED** type can contain an integer of almost arbitrary size and so cannot always be represented by a C **int64_t** type. However, in many cases (for example version numbers) they represent small integers which can be more easily manipulated if converted to an appropriate C integer type.

BUGS

The ambiguous return values of **ASN1_INTEGER_get()** and **ASN1_ENUMERATED_get()** mean these functions should be avoided if possible. They are retained for compatibility. Normally the ambiguous return values are not legitimate values for the fields they represent.

RETURN VALUES

ASN1_INTEGER_set_int64(), **ASN1_INTEGER_set()**, **ASN1_ENUMERATED_set_int64()** and **ASN1_ENUMERATED_set()** return 1 for success and 0 for failure. They will only fail if a memory allocation error occurs.

ASN1_INTEGER_get_int64() and **ASN1_ENUMERATED_get_int64()** return 1 for success and 0 for failure. They will fail if the passed type is incorrect (this will only happen if there is a programming error) or if the value exceeds the range of an **int64_t** type.

BN_to_ASN1_INTEGER() and **BN_to_ASN1_ENUMERATED()** return an **ASN1_INTEGER** or **ASN1_ENUMERATED** structure respectively or NULL if an error occurs. They will only fail due to a memory allocation error.

ASN1_INTEGER_to_BN() and **ASN1_ENUMERATED_to_BN()** return a **BIGNUM** structure of NULL if an error occurs. They can fail if the passed type is incorrect (due to programming error) or due to a memory allocation failure.

SEE ALSO

ERR_get_error(3)

HISTORY

ASN1_INTEGER_set_int64(), **ASN1_INTEGER_get_int64()**, **ASN1_ENUMERATED_set_int64()** and **ASN1_ENUMERATED_get_int64()** were added in OpenSSL 1.1.0.

COPYRIGHT

Copyright 2015–2020 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>>.