

**NAME**

ber\_int\_t, ber\_uint\_t, ber\_len\_t, ber\_slen\_t, ber\_tag\_t, struct berval, BerValue, BerVarray, BerElement, ber\_bvfree, ber\_bvecfree, ber\_bvecadd, ber\_bvarray\_free, ber\_bvarray\_add, ber\_bvdup, ber\_dupbv, ber\_bvstr, ber\_bvstrdup, ber\_str2bv, ber\_alloc\_t, ber\_init, ber\_init2, ber\_free - OpenLDAP LBER types and allocation functions

**LIBRARY**

OpenLDAP LBER (liblber, -llber)

**SYNOPSIS**

```
#include <lber.h>
```

```
typedef impl_tag_t ber_tag_t;
typedef impl_int_t ber_int_t;
typedef impl_uint_t ber_uint_t;
typedef impl_len_t ber_len_t;
typedef impl_slen_t ber_slen_t;
```

```
typedef struct berval {
    ber_len_t bv_len;
    char *bv_val;
} BerValue, *BerVarray;
```

```
typedef struct berelement BerElement;
```

```
void ber_bvfree(struct berval *bv);
```

```
void ber_bvecfree(struct berval **bvec);
```

```
void ber_bvecadd(struct berval ***bvec, struct berval *bv);
```

```
void ber_bvarray_free(struct berval *bvarray);
```

```
void ber_bvarray_add(BerVarray *bvarray, BerValue *bv);
```

```
struct berval *ber_bvdup(const struct berval *bv);
```

```
struct berval *ber_dupbv(const struct berval *dst, struct berval *src);
```

```
struct berval *ber_bvstr(const char *str);
```

```
struct berval *ber_bvstrdup(const char *str);
```

```
struct berval *ber_str2bv(const char *str, ber_len_t len, int dup, struct berval *bv);
```

```
BerElement *ber_alloc_t(int options);
```

```
BerElement *ber_init(struct berval *bv);
```

```
void ber_init2(BerElement *ber, struct berval *bv, int options);
```

```
void ber_free(BerElement *ber, int freebuf);
```

## DESCRIPTION

The following are the basic types and structures defined for use with the Lightweight BER library.

**ber\_int\_t** is a signed integer of at least 32 bits. It is commonly equivalent to **int**. **ber\_uint\_t** is the unsigned variant of **ber\_int\_t**.

**ber\_len\_t** is an unsigned integer of at least 32 bits used to represent a length. It is commonly equivalent to a **size\_t**. **ber\_slen\_t** is the signed variant to **ber\_len\_t**.

**ber\_tag\_t** is an unsigned integer of at least 32 bits used to represent a BER tag. It is commonly equivalent to a **unsigned long**.

The actual definitions of the integral **impl\_TYPE\_t** types are platform specific.

**BerValue**, commonly used as **struct berval**, is used to hold an arbitrary sequence of octets. **bv\_val** points to **bv\_len** octets. **bv\_val** is not necessarily terminated by a NULL (zero) octet. **ber\_bvfree()** frees a BerValue, pointed to by *bv*, returned from this API. If *bv* is NULL, the routine does nothing.

**ber\_bvecfree()** frees an array of BerValues (and the array), pointed to by *bvec*, returned from this API. If *bvec* is NULL, the routine does nothing. **ber\_bvecadd()** appends the *bv* pointer to the *bvec* array. Space for the array is allocated as needed. The end of the array is marked by a NULL pointer.

**ber\_bvarray\_free()** frees an array of BerValues (and the array), pointed to by *bvarray*, returned from this API. If *bvarray* is NULL, the routine does nothing. **ber\_bvarray\_add()** appends the contents of the BerValue pointed to by *bv* to the *bvarray* array. Space for the new element is allocated as needed. The end of the array is marked by a BerValue with a NULL *bv\_val* field.

**ber\_bvdup()** returns a copy of a BerValue. The routine returns NULL upon error (e.g. out of memory).

The caller should use **ber\_bvfree()** to deallocate the resulting BerValue. **ber\_dupbv()** copies a BerValue from *src* to *dst*. If *dst* is NULL a new BerValue will be allocated to hold the copy. The routine returns NULL upon error, otherwise it returns a pointer to the copy. If *dst* is NULL the caller should use **ber\_bvfree()** to deallocate the resulting BerValue, otherwise **ber\_memfree()** should be used to deallocate the *dst->bv\_val*. (The **ber\_bvdup()** function is internally implemented as **ber\_dupbv(NULL, bv)**. **ber\_bvdup()** is provided only for compatibility with an expired draft of the LDAP C API; **ber\_dupbv()** is the preferred interface.)

**ber\_bvstr()** returns a BerValue containing the string pointed to by *str*. **ber\_bvstrdup()** returns a BerValue containing a copy of the string pointed to by *str*. **ber\_str2bv()** returns a BerValue containing the string pointed to by *str*, whose length may be optionally specified in *len*. If *dup* is non-zero, the BerValue will contain a copy of *str*. If *len* is zero, the number of bytes to copy will be determined by **strlen(3)**, otherwise *len* bytes will be copied. If *bv* is non-NULL, the result will be stored in the given BerValue, otherwise a new BerValue will be allocated to store the result. NOTE: Both **ber\_bvstr()** and **ber\_bvstrdup()** are implemented as macros using **ber\_str2bv()** in this version of the library.

**BerElement** is an opaque structure used to maintain state information used in encoding and decoding. **ber\_alloc\_t()** is used to create an empty BerElement structure. If **LBER\_USE\_DER** is specified for the *options* parameter then data lengths for data written to the BerElement will be encoded in the minimal number of octets required, otherwise they will always be written as four byte values. **ber\_init()** creates a BerElement structure that is initialized with a copy of the data in its *bv* parameter. **ber\_init2()** initializes an existing BerElement *ber* using the data in the *bv* parameter. The data is referenced directly, not copied. The *options* parameter is the same as for **ber\_alloc\_t()**. **ber\_free()** frees a BerElement pointed to by *ber*. If *ber* is NULL, the routine does nothing. If *freebuf* is zero, the internal buffer is not freed.

## SEE ALSO

**lber-encode(3)**, **lber-decode(3)**, **lber-memory(3)**

## ACKNOWLEDGEMENTS

**OpenLDAP Software** is developed and maintained by The OpenLDAP Project

<<http://www.openldap.org/>>. **OpenLDAP Software** is derived from the University of Michigan LDAP 3.3 Release.