NAME

ber_sockbuf_alloc, ber_sockbuf_free, ber_sockbuf_ctrl, ber_sockbuf_add_io, ber_sockbuf_remove_io, Sockbuf_IO - OpenLDAP LBER I/O infrastructure

LIBRARY

OpenLDAP LBER (liblber, -llber)

SYNOPSIS

```
#include <lber.h>

Sockbuf *ber_sockbuf_alloc( void );

void ber_sockbuf_free(Sockbuf *sb);

int ber_sockbuf_ctrl(Sockbuf *sb, int opt, void *arg);

int ber_sockbuf_add_io(Sockbuf *sb, Sockbuf_IO *sbio, int layer, void *arg);

int ber_sockbuf_remove_io(Sockbuf *sb, Sockbuf_IO *sbio, int layer);

typedef struct sockbuf_io_desc {
int sbiod_level;
Sockbuf *sbiod_sb;
Sockbuf_IO *sbiod_io;
void *sbiod_pvt;
struct sockbuf_io_desc *sbiod_next;
} Sockbuf_IO_Desc;

typedef struct sockbuf_io {
int (*sbi_setup)(Sockbuf_IO_Desc *sbiod, void *arg);
int (*sbi_remove)(Sockbuf_IO_Desc *sbiod);
int (*sbi_ctrl)(Sockbuf_IO_Desc *sbiod, int opt, void *arg);
ber_slen_t (*sbi_read)(Sockbuf_IO_Desc *sbiod, void *buf, ber_len_t len);
ber_slen_t (*sbi_write)(Sockbuf_IO_Desc *sbiod, void *buf, ber_len_t len);
int (*sbi_close)(Sockbuf_IO_Desc *sbiod);
} Sockbuf_IO;
```

DESCRIPTION

These routines are used to manage the low level I/O operations performed by the Lightweight BER

library. They are called implicitly by the other libraries and usually do not need to be called directly from applications. The I/O framework is modularized and new transport layers can be supported by appropriately defining a **Sockbuf_IO** structure and installing it onto an existing **Sockbuf**. **Sockbuf** structures are allocated and freed by **ber_sockbuf_alloc**() and **ber_sockbuf_free**(), respectively. The **ber_sockbuf_ctrl**() function is used to get and set options related to a **Sockbuf** or to a specific I/O layer of the **Sockbuf**. The **ber_sockbuf_add_io**() and **ber_sockbuf_remove_io**() functions are used to add and remove specific I/O layers on a **Sockbuf**.

Options for **ber_sockbuf_ctrl**() include:

LBER_SB_OPT_HAS_IO
Takes a **Sockbuf_IO *** argument and returns 1 if the given handler is installed on the **Sockbuf**, otherwise returns 0.

LBER_SB_OPT_GET_FD
Retrieves the file descriptor associated to the **Sockbuf**; **arg** must be a **ber_socket_t ***. The return value will be 1 if a valid descriptor was present, -1 otherwise.

LBER_SB_OPT_SET_FD
Sets the file descriptor of the **Sockbuf** to the descriptor pointed to by **arg**; **arg** must be a **ber_socket_t ***. The return value will always be 1.

LBER_SB_OPT_SET_NONBLOCK
Toggles the non-blocking state of the file descriptor associated to the **Sockbuf**. **arg** should be NULL to disable and non-NULL to enable the non-blocking state. The return value will be 1 for success, -1 otherwise.

LBER_SB_OPT_DRAIN
Flush (read and discard) all available input on the **Sockbuf**. The return value will be 1.

LBER_SB_OPT_NEEDS_READ
Returns non-zero if input is waiting to be read.

LBER_SB_OPT_NEEDS_WRITE
Returns non-zero if the **Sockbuf** is ready to be written.

LBER_SB_OPT_GET_MAX_INCOMING
Returns the maximum allowed size of an incoming message; **arg** must be a **ber_len_t ***. The return value will be 1.

LBER_SB_OPT_SET_MAX_INCOMING
Sets the maximum allowed size of an incoming message; **arg** must be a **ber_len_t ***. The return value will be 1.

Options not in this list will be passed down to each **Sockbuf_IO** handler in turn until one of them processes it. If the option is not handled **ber_sockbuf_ctrl**() will return 0.

Multiple **Sockbuf_IO** handlers can be stacked in multiple layers to provide various functionality. Currently defined layers include

LBER_SBIOD_LEVEL_PROVIDER
the lowest layer, talking directly to a network

LBER_SBIOD_LEVEL_TRANSPORT
an intermediate layer

LBER_SBIOD_LEVEL_APPLICATION
a higher layer

Currently defined **Sockbuf_IO** handlers in liblber include

ber_sockbuf_io_tcp
The default stream-oriented provider

ber_sockbuf_io_fd
A stream-oriented provider for local IPC sockets

ber_sockbuf_io_dgram
A datagram-oriented provider. This handler is only present if the liblber library was built with LDAP_CONNECTIONLESS defined.

ber_sockbuf_io_readahead
A buffering layer, usually used with a datagram provider to hide the datagram semantics from upper layers.

ber_sockbuf_io_debug
A generic handler that outputs hex dumps of all traffic. This handler may be inserted multiple times at arbitrary layers to show the flow of data between other handlers.

Additional handlers may be present in libldap if support for them was enabled:

ldap_pvt_sockbuf_io_sasl
An application layer handler for SASL encoding/decoding.

sb_tls_sbio
A transport layer handler for SSL/TLS encoding/decoding. Note that this handler is private to the library and is not exposed in the API.

The provided handlers are all instantiated implicitly by libldap, and applications generally will not need to directly manipulate them.

SEE ALSO

lber-decode(3), **lber-encode**(3), **lber-types**(3), **ldap_get_option**(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.