NAME

ldap_bind, ldap_bind_s, ldap_simple_bind, ldap_simple_bind_s, ldap_sasl_bind, ldap_sasl_bind_s, ldap_sasl_interactive_bind_s, ldap_parse_sasl_bind_result, ldap_unbind, ldap_unbind_s, ldap_unbind_ext, ldap_unbind_ext_s, ldap_set_rebind_proc - LDAP bind routines

LIBRARY

OpenLDAP LDAP (libldap, -lldap)

SYNOPSIS

```
#include <ldap.h>

int ldap_bind(LDAP *ld, const char *who, const char *cred,
    int method);

int ldap_bind_s(LDAP *ld, const char *who, const char *cred,
    int method);

int ldap_simple_bind(LDAP *ld, const char *who, const char *passwd);

int ldap_simple_bind_s(LDAP *ld, const char *who, const char *passwd);

int ldap_sasl_bind(LDAP *ld, const char *dn, const char *mechanism,
    struct berval *cred, LDAPControl *sctrls[],
    LDAPControl *cctrls[], int *msgidp);

int ldap_sasl_bind_s(LDAP *ld, const char *dn, const char *mechanism,
    struct berval *cred, LDAPControl *sctrls[],
    LDAPControl *cctrls[], struct berval **servercredp);

int ldap_parse_sasl_bind_result(LDAP *ld, LDAPMessage *res,
    struct berval **servercredp, int freeit);

int ldap_sasl_interactive_bind_s(LDAP *ld, const char *dn,
    const char *mechs,
    LDAPControl *sctrls[], LDAPControl *cctrls[],
    unsigned flags, LDAP_SASL_INTERACT_PROC *interact,
    void *defaults);

int ldap_sasl_interactive_bind(LDAP *ld, const char *dn,
    const char *mechs,
```

LDAPControl ****sctrls*[], LDAPControl ****cctrls***[],**
unsigned *flags***, LDAP_SASL_INTERACT_PROC ****interact***,**
void ****defaults*, LDAPMessage ****result***,**
const char *****rmechp*, int ****msgidp***);**

int (LDAP_SASL_INTERACT_PROC)(LDAP ****ld*, unsigned** *flags***, void ****defaults***, void ****sasl_interact***);**

int ldap_unbind(LDAP ****ld*);**

int ldap_unbind_s(LDAP ****ld*);**

int ldap_unbind_ext(LDAP ****ld*, LDAPControl ****sctrls***[],**
LDAPControl ****cctrls*[]);**

int ldap_unbind_ext_s(LDAP ****ld*, LDAPControl ****sctrls***[],**
LDAPControl ****cctrls*[]);**

int ldap_set_rebind_proc (LDAP ****ld*, LDAP_REBIND_PROC ****ldap_proc***, void ****params***);**

int (LDAP_REBIND_PROC)(LDAP ****ld*, LDAP_CONST char ****url***, ber_tag_t** *request***, ber_int_t** *msgid***, void ****par

DESCRIPTION

These routines provide various interfaces to the LDAP bind operation. After an association with an LDAP server is made using **ldap_init**(3), an LDAP bind operation should be performed before other operations are attempted over the connection. An LDAP bind is required when using Version 2 of the LDAP protocol; it is optional for Version 3 but is usually needed due to security considerations.

There are three types of bind calls, ones providing simple authentication, ones providing SASL authentication, and general routines capable of doing either simple or SASL authentication.

SASL (Simple Authentication and Security Layer) can negotiate one of many different kinds of authentication. Both synchronous and asynchronous versions of each variant of the bind call are provided. All routines take *ld* as their first parameter, as returned from **ldap_init**(3).

SIMPLE AUTHENTICATION

The simplest form of the bind call is **ldap_simple_bind_s()**. It takes the DN to bind as in *who*, and the userPassword associated with the entry in *passwd*. It returns an LDAP error indication (see **ldap_error**(3)). The **ldap_simple_bind()** call is asynchronous, taking the same parameters but only initiating the bind operation and returning the message id of the request it sent. The result of the operation can be obtained by a subsequent call to **ldap_result**(3). The **ldap_sasl_bind_s()** and

asynchronous **ldap_sasl_bind()** functions can also be used to make a simple bind by using LDAP_SASL_SIMPLE as the SASL mechanism.

GENERAL AUTHENTICATION

The **ldap_bind()** and **ldap_bind_s()** routines can be used when the authentication method to use needs to be selected at runtime. They both take an extra *method* parameter selecting the authentication method to use. It should be set to LDAP_AUTH_SIMPLE to select simple authentication. **ldap_bind()** returns the message id of the request it initiates. **ldap_bind_s()** returns an LDAP error indication.

SASL AUTHENTICATION

For SASL binds the server always ignores any provided DN, so the *dn* parameter should always be NULL. **ldap_sasl_bind_s**() sends a single SASL bind request with the given SASL *mechanism* and credentials in the *cred* parameter. The format of the credentials depends on the particular SASL mechanism in use. For mechanisms that provide mutual authentication the server's credentials will be returned in the *servercredp* parameter. The routine returns an LDAP error indication (see **ldap_error**(3)). The **ldap_sasl_bind**() call is asynchronous, taking the same parameters but only sending the request and returning the message id of the request it sent. The result of the operation can be obtained by a subsequent call to **ldap_result**(3). The result must be additionally parsed by **ldap_parse_sasl_bind_result**() to obtain any server credentials sent from the server.

Any returned server credentials should be freed using **ber_bvfree**().

Many SASL mechanisms require multiple message exchanges to perform a complete authentication. Applications should generally use **ldap_sasl_interactive_bind_s**() rather than calling the basic **ldap_sasl_bind**() functions directly. The *mechs* parameter should contain a space-separated list of candidate mechanisms to use. If this parameter is NULL or empty the library will query the supportedSASLMechanisms attribute from the server's rootDSE for the list of SASL mechanisms the server supports. The *flags* parameter controls the interaction used to retrieve any necessary SASL authentication parameters and should be one of:

LDAP_SASL_AUTOMATIC
: use defaults if available, prompt otherwise

LDAP_SASL_INTERACTIVE
: always prompt

LDAP_SASL_QUIET
: never prompt

The *interact* function uses the provided *defaults* to handle requests from the SASL library for particular

authentication parameters. There is no defined format for the *defaults* information; it is up to the caller to use whatever format is appropriate for the supplied *interact* function. The *sasl_interact* parameter comes from the underlying SASL library. When used with Cyrus SASL this is an array of **sasl_interact_t** structures. The Cyrus SASL library will prompt for a variety of inputs, including:

SASL_CB_GETREALM
: the realm for the authentication attempt

SASL_CB_AUTHNAME
: the username to authenticate

SASL_CB_PASS
: the password for the provided username

SASL_CB_USER
: the username to use for proxy authorization

SASL_CB_NOECHOPROMPT
: generic prompt for input with input echoing disabled

SASL_CB_ECHOPROMPT
: generic prompt for input with input echoing enabled

SASL_CB_LIST_END
: indicates the end of the array of prompts

See the Cyrus SASL documentation for more details.

Applications which need to manage connections asynchronously may use **ldap_sasl_interactive_bind**() instead of the synchronous version. A valid mechs parameter must be supplied, otherwise the library will be forced to query the server for a list of supported mechanisms, and this query will be performed synchronously. The other parameters are the same as for the synchronous function, with three additional parameters. The actual SASL mechanism that was used, and the message ID for use with **ldap_result**() will be returned in rmechp and msgidp, respectively. The value in rmechp must not be modified by the caller and must be passed back on each subsequent call. The message obtained from **ldap_result**() must be passed in the result parameter. This parameter must be NULL when initiating a new Bind. The caller must free the result message after each call using **ldap_msgfree**(). The **ldap_sasl_interactive_bind**() function returns an LDAP result code. If the code is LDAP_SASL_BIND_IN_PROGRESS then the Bind is not complete yet, and this function must be called again with the next result from the server.

REBINDING

The **ldap_set_rebind_proc** function() sets the process to use for binding when an operation returns a referral. This function is used when an application needs to bind to another server in order to follow a referral or search continuation reference.

The function takes *ld*, the *rebind* function, and the *params*, the arbitrary data like state information which the client might need to properly rebind. The LDAP_OPT_REFERRALS option in the *ld* must be set to ON for the libraries to use the rebind function. Use the **ldap_set_option** function to set the value.

The rebind function parameters are as follows:

The *ld* parameter must be used by the application when binding to the referred server if the application wants the libraries to follow the referral.

The *url* parameter points to the URL referral string received from the LDAP server. The LDAP application can use the **ldap_url_parse**(3) function to parse the string into its components.

The *request* parameter specifies the type of request that generated the referral.

The *msgid* parameter specifies the message ID of the request generating the referral.

The *params* parameter is the same value as passed originally to the **ldap_set_rebind_proc**() function.

The LDAP libraries set all the parameters when they call the rebind function. The application should not attempt to free either the ld or the url structures in the rebind function.

The application must supply to the rebind function the required authentication information such as, user name, password, and certificates. The rebind function must use a synchronous bind method.

UNBINDING

The **ldap_unbind()** call is used to unbind from the directory, terminate the current association, and free the resources contained in the *ld* structure. Once it is called, the connection to the LDAP server is closed, and the *ld* structure is invalid. The **ldap_unbind_s()** call is just another name for **ldap_unbind()**; both of these calls are synchronous in nature.

The **ldap_unbind_ext()** and **ldap_unbind_ext_s()** allows the operations to specify controls.

ERRORS

Asynchronous routines will return -1 in case of error, setting the *ld_errno* parameter of the *ld* structure.

Synchronous routines return whatever *ld_errno* is set to. See **ldap_error**(3) for more information.

NOTES

If an anonymous bind is sufficient for the application, the rebind process need not be provided. The LDAP libraries with the LDAP_OPT_REFERRALS option set to ON (default value) will automatically follow referrals using an anonymous bind.

If the application needs stronger authentication than an anonymous bind, you need to provide a rebind process for that authentication method. The bind method must be synchronous.

SEE ALSO

ldap(3), **ldap_error**(3), **ldap_open**(3), **ldap_set_option**(3), **ldap_url_parse**(3) **RFC 4422** (http://www.rfc-editor.org), **Cyrus SASL** (http://asg.web.cmu.edu/sasl/)

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.