Oracle® Internet Directory

Application Developer's Guide

Release 2.0.6

April 2000

Part No. A83775-01

Oracle Internet Directory Application Developer’s Guide, Release 2.0.6

Part No. A83775-01

Primary Author: Richard Smith

Contributors: Saheli Dey, Rajinder Gupta, Stephen Lee, Radikha Moolky, David Saslav

Contents

Send Us Your Comments

Oracle Internet Directory Application Developer's Guide, Release 2.0.6

Part No. A83775-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail - infodev@us.oracle.com
- FAX - (650) 506-7228. Attn: Oracle Internet Directory Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Internet Directory Documentation Manager
  500 Oracle Parkway, 4op7
  Redwood Shores, CA 94065
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle Internet Directory Application Developer's Guide provides information for enabling applications to access Oracle Internet Directory by using the C API.

Audience

This book is intended primarily for application developers enabling applications to access Oracle Internet Directory by using the C API. It is also intended for anyone who wants to know how the Oracle Internet Directory C API works.

Book Organization

Chapter 1, "C Application Programming Interface"	Introduces the Oracle Internet Directory C API and provides examples of how to use it
Chapter 2, "Command Line Tools Syntax"	Provides syntax, usage notes, and examples for using LDAP Data Interchange Format (LDIF) and LDAP command line tools

Related Documentation

Oracle Internet Directory Administrator's Guide.

Oracle8*i* documentation set

Chadwick, David. *Understanding X.500 The Directory*. Thomson Computer Press, 1996. This book is now out of print, but is available online at: http://www.salford.ac.uk/its024/Version.Web/Contents.htm

Hodges, Jeff, Staff Scientist, Oblix, Inc.,
http://www.kingsmountain.com/ldapRoadmap.shtml

Howes, Tim and Mark Smith. *LDAP: Programming Directory-enabled Applications with Lightweight Directory Access Protocol.* Macmillan Technical Publishing, 1997.

Howes, Tim, Mark Smith and Gordon Good, *Understanding and Deploying LDAP Directory Services.* Macmillan Technical Publishing, 1999.

Kosiur, Dave, LDAP: "The next-generation directory?," *SunWorld Online,* October 1997.

Radicati, Sara, *X.500 Directory Services, Technology and Deployment,* International Thomson Computer Press, 1994.

University of Michigan LDAP Repository,
http://www.umich.edu/~dirsvcs/ldap/index.html

Conventions

The following conventions are used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command have been omitted.
bold	Boldface text indicates text you must type in a command, or a subheading.
italics	Italics indicate: ■ In a code example, a variable for which you must supply a value ■ In regular text, special emphasis ■ Book titles
`courier`	Courier is used for user input and code examples.
syntax	This typeface is used for syntax explanations in code examples.
< >	In code examples, angle brackets may enclose user-supplied names.
[ ]	Brackets enclose a choice of optional items from which you can choose one or none.

Convention	Meaning
{ }	Braces enclose a choice of required items from which you can choose one.

1 C Application Programming Interface

This chapter introduces the Oracle Internet Directory API and provides examples of how to use it.

This document covers topics in the following sections:

About the Oracle Internet Directory C API

The Oracle Internet Directory SDK C API Release 2.0.6 is based on:

- LDAP Version 2 C API
- Oracle extensions to support SSL

Oracle Internet Directory SDK C API Release 2.0.6 supports the following modes:

- SSL—All communication secured using SSL
- Non-SSL—Client-to-server communication not secure

To use the SSL mode, you must use the Oracle SSL call interface. You determine which mode you are using by the presence or absence of the SSL calls in the API usage. You can easily switch between SSL and non-SSL modes.

> **See Also:** "Sample C API Usage" on page 1-7 for more details on how to use the two modes

This section contains these topics:

- Oracle Internet Directory SDK C API SSL Extensions
- Summary of LDAP C API

Oracle Internet Directory SDK C API SSL Extensions

Oracle SSL extensions to the LDAP API are based on standard SSL protocol. The SSL extensions provide encryption and decryption of data over the wire, and authentication.

There are three modes of authentication:

- One-way—Only the server is authenticated by the client
- Two-way—Both the server and the client are authenticated by each other
- None—Neither client nor server is authenticated, and only SSL encryption is used

The type of authentication is indicated by a parameter in the SSL interface call.

SSL Interface Calls

There is only one call required to enable SSL:

```
int ldap_init_SSL(Sockbuf *sb, text *sslwallet, text *sslwalletpasswd, int
sslauthmode)
```

The ldap_init_SSL call performs the necessary handshake between client and server using the standard SSL protocol. If the call is successful, all subsequent communication happens over a secure connection.

Argument	Description
sb	Socket buffer handle returned by the *ldap_open* call as part of LDAP handle.
sslwallet	Location of the user wallet.
sslwalletpasswd	Password required to use the wallet.
sslauthmode	SSL authentication mode user wants to use. Possible values are: ▪ GSLC_SSL_NO_AUTH—No authentication required ▪ GSLC_SSL_ONEWAY_AUTH—Only server authentication required. ▪ GSLC_SSL_TWOWAY_AUTH—Both server and client authentication required. A return value of 0 indicates success. A non zero return value indicates an error. The error code can be decoded by using the function ldap_err2string.

See Also: See "Sample C API Usage" on page 1-7

Wallet Support

To use the SSL feature, both the server and the client may require wallets, depending on which authentication mode is being used. Release 2.0.6 of the API supports only Oracle Wallet. You can create wallets using Oracle Wallet Manager.

Summary of LDAP C API

This section lists all the calls available in the LDAP C API found in RFC 1823.

See Also: The following URL: http://www.ietf.org/rfc/rfc1823.txt for a more detailed explanation of these calls

This section contains these topics:

- Initializing and Ending LDAP Sessions
- Authenticating to an LDAP Server
- Getting Search Results
- Working with Distinguished Names
- Freeing Memory

Initializing and Ending LDAP Sessions

ldap_open()	Open a connection to an LDAP server
ldap_unbind()	End an LDAP session

Authenticating to an LDAP Server

ldap_ bind()	General authentication to an LDAP server
ldap_bind_s()	
ldap_simple_bind()	Simple authentication to an LDAP server
ldap_simple_bind_s()	
ldap_kerberos_bind()	Kerberos authentication to an LDAP server
ldap_kerberos_bind_s()	

Performing LDAP Operations

ldap_add() / ldap_add_s()	Add a new entry to the directory
ldap_modify() / ldap_modify_s()	Modify an entry in the directory

ldap_delete() / ldap_delete_s()	Delete an entry from the directory
ldap_modrdn() / ldap_modrdn_s()	Modify the RDN of an entry in the directory
ldap_search() / ldap_search_s()	Search the directory
ldap_search_st()	Search the directory with a timeout value
ldap_compare()/ ldap_compare_s()	Compare entries in the directory
ldap_result()	Check the results of an asynchronous operation
ldap_abandon()	Cancel an asynchronous operation

Getting Search Results

ldap_get_dn()	Get the distinguished name for an entry
ldap_first_entry()	Get the first entry in a chain of search results
ldap_next_entry()	Get the next entry in a chain of search results
ldap_count_ entries()	Count the number of entries in a chain of search results
ldap_first_ attribute()	Get the name of the first attribute in an entry
ldap_next_ attribute()	Get the name of the next attribute in an entry
ldap_get_values()	Get the string values of an attribute
ldap_get_values_ len()	Get the binary values of an attribute
ldap_count_ values()	Count the string values of an attribute
ldap_count_ values_len()	Count the binary values of an attribute

Working with Distinguished Names

ldap_get_dn()	Get the distinguished name for an entry
ldap_explode_dn()	Split up a distinguished name into its components
ldap_dn2ufn()	Converts the name into a more user friendly format

Handling Errors

ldap_result2error()	Returns the error code from result message.
ldap_err2string()	Get the error message for a specific error code
ldap_perror	Prints the message supplied in message.

Freeing Memory

ldap_memfree()	Free memory allocated by an LDAP API function call
ldap_msgfree()	Free the memory allocated for search results or other LDAP operation results
ldap_value_free()	Free the memory allocated for the string values of an attribute
ldap_value_free_len()	Free the memory allocated for the binary values of an attribute
ber_free()	Free the memory allocated for a BerElement structure

Sample C API Usage

The following examples show how to use the API both with and without SSL. More complete examples are given in RFC 1823. The sample code for the command line tool to perform LDAP search also demonstrates usage of the API in two modes.

This section contains these topics:

- API Usage with SSL
- API Usage Without SSL
- Sample Command Line Tool for Searching

API Usage with SSL

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <gsle.h>
#include <gslc.h>
#include <gsld.h>
#include "gslcc.h"

main()
{
    LDAP          *ld;
    int           ret = 0;
    ….
    /* open a connection */
    if ( (ld = ldap_open( "MyHost", 636 )) == NULL )
         exit( 1 );

    /* SSL initialization */
    ret = ldap_init_SSL(&ld->ld_sb, “file:/sslwallet”, “welcome”,
                                        GSLC_SSL_ONEWAY_AUTH );
    if(ret != 0)
    {
        printf(" %s \n", ldap_err2string(ret));
        exit(1);
    }

    /* authenticate as nobody */
    if ( ldap_bind_s( ld, NULL, NULL ) != LDAP_SUCCESS ) {
         ldap_perror( ld, "ldap_bind_s" );
         exit( 1 );
    }

    …..
    …..
}
```

Because the user is making the `ldap_init_SSL` call, the client-to-sever communication in the above example is secured by using SSL.

API Usage Without SSL

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <gsle.h>
#include <gslc.h>
#include <gsld.h>
#include "gslcc.h"

main()
{
    LDAP          *ld;
    int           ret = 0;
    ….

    /* open a connection */
    if ( (ld = ldap_open( "MyHost", LDAP_PORT )) == NULL )
         exit( 1 );

    /* authenticate as nobody */
    if ( ldap_bind_s( ld, NULL, NULL ) != LDAP_SUCCESS ) {
         ldap_perror( ld, "ldap_bind_s" );
          exit( 1 );
    }
    …..
    …..
}
```

In the above example, the user is not making the `ldap_init_SSL` call, and the client-to-server communication is therefore not secure.

Sample Command Line Tool for Searching

The Oracle Internet Directory SDK Release 2.0.6 provides a sample command line tool, ldapsearch, for showing users how to use the LDAP API to build applications. You can use ldapsearch to perform LDAP searches in either SSL or non-SSL mode. The source file (`ldapsearch.c`) and the make file (`Make.sh`) are included in the release. You can find them in the following directory: `ORACLE_HOME/ldap/demo`.

The following is sample code for ldapsearch:

```
/*
   NAME
     s0gsldsearch.c - <one-line expansion of the name>
   DESCRIPTION
     <short description of component this file declares/defines>
   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line descriptions>
   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line descriptions>
   RETURNS
     <function return values, for .c file with single function>
   NOTES
     <other useful comments, qualifications, etc.>
*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include "ldap.h"

#define DEFSEP"="
#define LDAPSEARCH_BINDDN        NULL
#define LDAPSEARCH_BASE          DEFAULT_BASE
#define DEFAULT_BASE    "o=oracle, c=US"

#ifdef LDAP_DEBUG
extern int ldap_debug, lber_debug;
#endif /* LDAP_DEBUG */

usage( s )
char*s;
{
    fprintf( stderr, "usage: %s [options] filter [attributes...]\nwhere:\n", s
);
    fprintf( stderr, "    filter\tRFC-1558 compliant LDAP search filter\n" );
    fprintf( stderr, "    attributes\twhitespace-separated list of attributes to
retrieve\n" );
    fprintf( stderr, "\t\t(if no attribute list is given, all are retrieved)\n"
);
    fprintf( stderr, "options:\n" );
    fprintf( stderr, "    -n\t\tshow what would be done but don't actually
search\n" );
    fprintf( stderr, "    -v\t\trun in verbose mode (diagnostics to standard
output)\n" );
    fprintf( stderr, "    -t\t\twrite values to files in /tmp\n" );
    fprintf( stderr, "    -u\t\tinclude User Friendly entry names in the
output\n" );
    fprintf( stderr, "    -A\t\tretrieve attribute names only (no values)\n" );
    fprintf( stderr, "    -B\t\tdo not suppress printing of non-ASCII values\n"
);
    fprintf( stderr, "    -L\t\tprint entries in LDIF format (-B is implied)\n"
);
#ifdef LDAP_REFERRALS
    fprintf( stderr, "    -R\t\tdo not automatically follow referrals\n" );
#endif /* LDAP_REFERRALS */
    fprintf( stderr, "    -d level\tset LDAP debugging level to `level'\n" );
    fprintf( stderr, "    -F sep\tprint `sep' instead of `=' between attribute
names and values\n" );
    fprintf( stderr, "    -S attr\tsort the results by attribute `attr'\n" );
    fprintf( stderr, "    -f file\tperform sequence of searches listed in
`file'\n" );
    fprintf( stderr, "    -b basedn\tbase dn for search\n" );
    fprintf( stderr, "    -s scope\tone of base, one, or sub (search scope)\n"
);
    fprintf( stderr, "    -a deref\tone of never, always, search, or find (alias
dereferencing)\n" );
    fprintf( stderr, "    -l time lim\ttime limit (in seconds) for search\n" );
    fprintf( stderr, "    -z size lim\tsize limit (in entries) for search\n" );
    fprintf( stderr, "    -D binddn\tbind dn\n" );
    fprintf( stderr, "    -w passwd\tbind passwd (for simple authentication)\n"
);
#ifdef KERBEROS
    fprintf( stderr, "    -k\t\tuse Kerberos instead of Simple Password
authentication\n" );
#endif
    fprintf( stderr, "    -h host\tldap server\n" );
    fprintf( stderr, "    -p port\tport on ldap server\n" );
    fprintf( stderr, "    -W Wallet\tWallet location\n" );
    fprintf( stderr, "    -P Wpasswd\tWallet Password\n" );
    fprintf( stderr, "    -U SSLAuth\tSSL Authentication Mode\n" );
    return;
}

static char*binddn = LDAPSEARCH_BINDDN;
static char*passwd = NULL;
static char*base = LDAPSEARCH_BASE;
static char*ldaphost = NULL;
static intldapport = LDAP_PORT;
static char*sep = DEFSEP;
static char*sortattr = NULL;
static intskipsortattr = 0;
static intverbose, not, includeufn, allow_binary, vals2tmp, ldif;
/* TEMP */

main( argc, argv )
intargc;
char**argv;
{
    char*infile, *filtpattern, **attrs, line[ BUFSIZ ];
    FILE*fp;
    intrc, i, first, scope, kerberos, deref, attrsonly;
    intldap_options, timelimit, sizelimit, authmethod;
    LDAP*ld;
    extern char*optarg;
    extern intoptind;
    charlocalHostName[MAXHOSTNAMELEN + 1];
    char *sslwrl = NULL;
    char*sslpasswd = NULL;
int sslauth=0,err=0;

    infile = NULL;
    deref = verbose = allow_binary = not = kerberos = vals2tmp =
    attrsonly = ldif = 0;
#ifdef LDAP_REFERRALS
    ldap_options = LDAP_OPT_REFERRALS;
#else /* LDAP_REFERRALS */
    ldap_options = 0;
#endif /* LDAP_REFERRALS */
    sizelimit = timelimit = 0;
    scope = LDAP_SCOPE_SUBTREE;

    while (( i = getopt( argc, argv,
#ifdef KERBEROS
    "KknuvtRABLD:s:f:h:b:d:p:F:a:w:l:z:S:"
#else
    "nuvtRABLD:s:f:h:b:d:p:F:a:w:l:z:S:W:P:U:"
#endif
    )) != EOF ) {
switch( i ) {
case 'n':/* do Not do any searches */
    ++not;
    break;
case 'v':/* verbose mode */
```

```
    ++verbose;
    break;
case 'd':
#ifdef LDAP_DEBUG
    ldap_debug = lber_debug = atoi( optarg );/* */
#else /* LDAP_DEBUG */
    fprintf( stderr, "compile with -DLDAP_DEBUG for debugging\n" );
#endif /* LDAP_DEBUG */
    break;
#ifdef KERBEROS
case 'k':/* use kerberos bind */
    kerberos = 2;
    break;
case 'K':/* use kerberos bind, 1st part only */
    kerberos = 1;
    break;
#endif
case 'u':/* include UFN */
    ++includeufn;
    break;
case 't':/* write attribute values to /tmp files */
    ++vals2tmp;
    break;
case 'R':/* don't automatically chase referrals */
#ifdef LDAP_REFERRALS
    ldap_options &= ~LDAP_OPT_REFERRALS;
#else /* LDAP_REFERRALS */
    fprintf( stderr,
    "compile with -DLDAP_REFERRALS for referral support\n" );
#endif /* LDAP_REFERRALS */
    break;
case 'A':/* retrieve attribute names only -- no values */
    ++attrsonly;
    break;
case 'L':/* print entries in LDIF format */
    ++ldif;
    /* fall through -- always allow binary when outputting LDIF */
case 'B':/* allow binary values to be printed */
    ++allow_binary;
    break;
case 's':/* search scope */
    if ( strncasecmp( optarg, "base", 4 ) == 0 ) {
scope = LDAP_SCOPE_BASE;
    } else if ( strncasecmp( optarg, "one", 3 ) == 0 ) {
scope = LDAP_SCOPE_ONELEVEL;
```

```
    } else if ( strncasecmp( optarg, "sub", 3 ) == 0 ) {
scope = LDAP_SCOPE_SUBTREE;
    } else {
fprintf( stderr, "scope should be base, one, or sub\n" );
usage( argv[ 0 ] );
                exit(1);
    }
    break;

case 'a':/* set alias deref option */
    if ( strncasecmp( optarg, "never", 5 ) == 0 ) {
deref = LDAP_DEREF_NEVER;
    } else if ( strncasecmp( optarg, "search", 5 ) == 0 ) {
deref = LDAP_DEREF_SEARCHING;
    } else if ( strncasecmp( optarg, "find", 4 ) == 0 ) {
deref = LDAP_DEREF_FINDING;
    } else if ( strncasecmp( optarg, "always", 6 ) == 0 ) {
deref = LDAP_DEREF_ALWAYS;
    } else {
fprintf( stderr, "alias deref should be never, search, find, or always\n" );
usage( argv[ 0 ] );
                exit(1);
    }
    break;

case 'F':/* field separator */
    sep = (char *)strdup( optarg );
    break;
case 'f':/* input file */
    infile = (char *)strdup( optarg );
    break;
case 'h':/* ldap host */
    ldaphost = (char *)strdup( optarg );
    break;
case 'b':/* searchbase */
    base = (char *)strdup( optarg );
    break;
case 'D':/* bind DN */
    binddn = (char *)strdup( optarg );
    break;
case 'p':/* ldap port */
    ldapport = atoi( optarg );
    break;
case 'w':/* bind password */
    passwd = (char *)strdup( optarg );
    break;
case 'l':/* time limit */
    timelimit = atoi( optarg );
    break;
case 'z':/* size limit */
    sizelimit = atoi( optarg );
    break;
case 'S':/* sort attribute */
    sortattr = (char *)strdup( optarg );
    break;
case 'W':/* Wallet URL */
    sslwrl = (char *)strdup( optarg );
    break;
case 'P':/* Wallet password */
    sslpasswd = (char *)strdup( optarg );
    break;
case 'U':/* SSL Authentication Mode */
    sslauth = atoi( optarg );
    break;
default:
    usage( argv[0] );
            exit(1);
            break;
}
    }

    if ( argc - optind < 1 ) {
usage( argv[ 0 ] );
        exit(1);
    }
    filtpattern = (char *)strdup( argv[ optind ] );
    if ( argv[ optind + 1 ] == NULL ) {
attrs = NULL;
    } else if ( sortattr == NULL || *sortattr == '\0' ) {
        attrs = &argv[ optind + 1 ];
    } else {
for ( i = optind + 1; i < argc; i++ ) {
    if ( strcasecmp( argv[ i ], sortattr ) == 0 ) {
break;
    }
}
if ( i == argc ) {
skipsortattr = 1;
argv[ optind ] = sortattr;
} else {
```

```
optind++;
}
        attrs = &argv[ optind ];
    }

    if ( infile != NULL ) {
if ( infile[0] == '-' && infile[1] == '\0' ) {
    fp = stdin;
} else if (( fp = fopen( infile, "r" )) == NULL ) {
    perror( infile );
    exit( 1 );
}
    }

    if (ldaphost == NULL) {
        if (gethostname(localHostName, MAXHOSTNAMELEN) != 0) {
                perror("gethostname");
                exit(1);
        }
        ldaphost = localHostName;
    }

    if ( verbose ) {
printf( "ldap_open( %s, %d )\n", ldaphost, ldapport );
    }

    if (( ld = ldap_open( ldaphost, ldapport )) == NULL ) {
perror( ldaphost );
exit( 1 );
    }

    if (sslauth > 1)
    {
       if (!sslwrl || !sslpasswd)
   {
          printf ("Null Wallet or password given\n");
          exit (0);
       }
    }
    if (sslauth > 0)
    {
        if (sslauth == 1)
            sslauth = GSLC_SSL_NO_AUTH;
        else if (sslauth == 2)
            sslauth = GSLC_SSL_ONEWAY_AUTH;
    else if (sslauth == 3)
      sslauth = GSLC_SSL_TWOWAY_AUTH;
    else
    {
  printf(" Wrong SSL Authenication Mode Value\n");
  exit(0);
    }

  err = ldap_init_SSL(&ld->ld_sb,sslwrl,sslpasswd,sslauth);
  if(err != 0)
{
  printf(" %s\n", ldap_err2string(err));
  exit(0);
}
  }

  ld->ld_deref = deref;
  ld->ld_timelimit = timelimit;
  ld->ld_sizelimit = sizelimit;
  ld->ld_options = ldap_options;

  if ( !kerberos ) {
authmethod = LDAP_AUTH_SIMPLE;
  } else if ( kerberos == 1 ) {
authmethod = LDAP_AUTH_KRBV41;
  } else {
authmethod =  LDAP_AUTH_KRBV4;
  }
  if ( ldap_bind_s( ld, binddn, passwd, authmethod ) != LDAP_SUCCESS ) {
ldap_perror( ld, "ldap_bind" );
exit( 1 );
  }

  if ( verbose ) {
printf( "filter pattern: %s\nreturning: ", filtpattern );
if ( attrs == NULL ) {
  printf( "ALL" );
} else {
  for ( i = 0; attrs[ i ] != NULL; ++i ) {
printf( "%s ", attrs[ i ] );
  }
}
putchar( '\n' );
  }
```

```
    if ( infile == NULL ) {
rc = dosearch( ld, base, scope, attrs, attrsonly, filtpattern, "" );
    } else {
rc = 0;
first = 1;
while ( rc == 0 && fgets( line, sizeof( line ), fp ) != NULL ) {
    line[ strlen( line ) - 1 ] = '\0';
    if ( !first ) {
putchar( '\n' );
    } else {
first = 0;
    }
    rc = dosearch( ld, base, scope, attrs, attrsonly, filtpattern,
    line );
}
if ( fp != stdin ) {
    fclose( fp );
}
    }

    ldap_unbind( ld );
    exit( rc );
}

dosearch( ld, base, scope, attrs, attrsonly, filtpatt, value )
    LDAP*ld;
    char*base;
    intscope;
    char**attrs;
    intattrsonly;
    char*filtpatt;
    char*value;
{
    charfilter[ BUFSIZ ], **val;
    intrc, first, matches;
    LDAPMessage*res, *e;

    sprintf( filter, filtpatt, value );

    if ( verbose ) {
printf( "filter is: (%s)\n", filter );
    }

    if ( not ) {
return( LDAP_SUCCESS );
```

```
    }

    if ( ldap_search( ld, base, scope, filter, attrs, attrsonly ) == -1 ) {
ldap_perror( ld, "ldap_search" );
return( ld->ld_errno );
    }

    matches = 0;
    first = 1;
    while ( (rc = ldap_result( ld, LDAP_RES_ANY, sortattr ? 1 : 0, NULL, &res ))
    == LDAP_RES_SEARCH_ENTRY ) {
matches++;
e = ldap_first_entry( ld, res );
if ( !first ) {
    putchar( '\n' );
} else {
    first = 0;
}
print_entry( ld, e, attrsonly );
ldap_msgfree( res );
    }
    if ( rc == -1 ) {
ldap_perror( ld, "ldap_result" );
return( rc );
    }
    if (( rc = ldap_result2error( ld, res, 0 )) != LDAP_SUCCESS ) {
        ldap_perror( ld, "ldap_search" );
    }
    if ( sortattr != NULL ) {
    extern intstrcasecmp();

    (void) ldap_sort_entries( ld, &res,
    ( *sortattr == '\0' ) ? NULL : sortattr, strcasecmp );
    matches = 0;
    first = 1;
    for ( e = ldap_first_entry( ld, res ); e != NULLMSG;
    e = ldap_next_entry( ld, e ) ) {
matches++;
if ( !first ) {
    putchar( '\n' );
} else {
    first = 0;
}
print_entry( ld, e, attrsonly );
    }
    }

    if ( verbose ) {
        printf( "%d matches\n", matches );
    }

    ldap_msgfree( res );
    return( rc );
}


print_entry( ld, entry, attrsonly )
    LDAP*ld;
    LDAPMessage*entry;
    intattrsonly;
{
    char*a, *dn, *ufn, tmpfname[ 64 ];
    inti, j, notascii;
    BerElement*ber;
    struct berval**bvals;
    FILE*tmpfp;
    extern char*mktemp();

    dn = ldap_get_dn( ld, entry );
    if ( ldif ) {
write_ldif_value( "dn", dn, strlen( dn ));
    } else {
printf( "%s\n", dn );
    }
    if ( includeufn ) {
ufn = ldap_dn2ufn( dn );
if ( ldif ) {
    write_ldif_value( "ufn", ufn, strlen( ufn ));
} else {
    printf( "%s\n", ufn );
}
free( ufn );
    }
    free( dn );

    for ( a = ldap_first_attribute( ld, entry, &ber ); a != NULL;
    a = ldap_next_attribute( ld, entry, ber ) ) {
if ( skipsortattr && strcasecmp( a, sortattr ) == 0 ) {
    continue;
}
```

```
if ( attrsonly ) {
    if ( ldif ) {
write_ldif_value( a, "", 0 );
    } else {
printf( "%s\n", a );
    }
} else if (( bvals = ldap_get_values_len( ld, entry, a )) != NULL ) {
    for ( i = 0; bvals[i] != NULL; i++ ) {
if ( vals2tmp ) {
    sprintf( tmpfname, "/tmp/ldapsearch-%s-XXXXXX", a );
    tmpfp = NULL;

    if ( mktemp( tmpfname ) == NULL ) {
perror( tmpfname );
    } else if (( tmpfp = fopen( tmpfname, "w")) == NULL ) {
perror( tmpfname );
    } else if ( fwrite( bvals[ i ]->bv_val,
    bvals[ i ]->bv_len, 1, tmpfp ) == 0 ) {
perror( tmpfname );
    } else if ( ldif ) {
write_ldif_value( a, tmpfname, strlen( tmpfname ));
    } else {
printf( "%s%s%s\n", a, sep, tmpfname );
    }

    if ( tmpfp != NULL ) {
fclose( tmpfp );
    }
} else {
    notascii = 0;
    if ( !allow_binary ) {
for ( j = 0; j < bvals[ i ]->bv_len; ++j ) {
    if ( !isascii( bvals[ i ]->bv_val[ j ] )) {
notascii = 1;
break;
    }
}
    }

    if ( ldif ) {
write_ldif_value( a, bvals[ i ]->bv_val,
bvals[ i ]->bv_len );
    } else
{
printf( "%s%s%s\n", a, sep,
notascii ? "NOT ASCII" : (char *)bvals[ i ]->bv_val );
    }
}
    }
    gsledePBerBvecfree( bvals );
}
    }
}


int
write_ldif_value( char *type, char *value, unsigned long vallen )
{
    char*ldif;

    if (( ldif = gsldlDLdifTypeAndValue( type, value, (int)vallen )) == NULL ) {
return( -1 );
    }

    fputs( ldif, stdout );
    free( ldif );

    return( 0 );
}
```

See Also: ldapsearch Syntax on page 2-17 for instructions on using ldapsearch

Dependencies and Limitations

This API can work against any release of Oracle Internet Directory server or a third party LDAP server.

To use the different authentication modes in SSL, the directory server requires corresponding configuration settings.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for details on how to set the Oracle directory server in various SSL authentication modes

Oracle Wallet Manager is required for creating wallets if you are using the C API in SSL mode.

TCP/IP Socket Library is required.

The following Oracle libraries are required:

- Oracle SSL-related libraries
- Oracle system libraries

Sample libraries are included in the release for the sample command line tool. You should replace these libraries with your own versions of the libraries.

The product supports only those authentication mechanisms described in LDAP SDK specifications (RFC 1823).

In SSL mode, if the server is set for two-way authentication and the client is set for one-way authentication, the `ldap_init_SSL` should fail. However, due to a known bug, it succeeds.

2

Command Line Tools Syntax

This chapter provides syntax, usage notes, and examples for using LDAP Data Interchange Format (LDIF) and LDAP command line tools. It contains these topics:

Command Line Tools Syntax

This section tells you how to use the following tools:

- ldapadd Syntax
- ldapaddmt Syntax
- ldapbind Syntax
- ldapcompare Syntax
- ldapdelete Syntax
- ldapmoddn Syntax
- ldapmodify Syntax
- ldapmodifymt Syntax
- ldapsearch Syntax

ldapadd Syntax

The ldapadd command line tool enables you to add entries, their object classes, attributes, and values to the directory. To add attributes to an existing entry, use the ldapmodify command, explained in "ldapmodify Syntax" on page 2-11.

ldapadd uses this syntax:

```
ldapadd [arguments] -f filename
```

where *filename* is the name of an LDIF file written with the specifications explained in the section "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21.

The following example adds the entry specified in the LDIF file `my_ldif_file.ldi`:

```
ldapadd -p 389 -h myhost -f my_ldif_file.ldi
```

Optional Arguments	Descriptions
-b	Specifies that you have included binary file names in the file, which are preceded by a forward slash character. The tool retrieves the actual values from the file referenced.
-c	Tells ldapadd to proceed in spite of errors. The errors will be reported. (If you do not use this option, ldapadd stops when it encounters an error.)
-D *binddn*	When authenticating to the directory, specifies doing so as the entry specified in *binddn*. Use this with the `-w` *password* option.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-f *filename*	Specifies the input name of the LDIF format import data file. For a detailed explanation of how to format an LDIF file, see "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21.
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-K	Same as `-k`, but performs only the first step of the Kerberos bind
-k	Authenticates using Kerberos authentication instead of simple authentication. To enable this option, you must compile with KERBEROS defined. You must already have a valid ticket granting ticket.
-n	Shows what would occur without actually performing the operation
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *password*	Provides the password required to connect
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapaddmt Syntax

ldapaddmt is like ldapadd: it enables you to add entries, their object classes, attributes, and values to the directory. It is unlike ldapadd in that it supports multiple threads for adding entries concurrently.

While it is processing LDIF entries, ldapaddmt logs errors in the `add.log` file in the current directory.

ldapaddmt uses this syntax:

```
ldapaddmt -T number_of_threads -h host -p port -f filename
```

where *`filename`* is the name of an LDIF file written with the specifications explained in the section "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21.

The following example uses five concurrent threads to process the entries in the file `myentries.ldif`.

```
ldapaddmt -T 5 -h node1 -p 3000 -f myentries.ldif
```

Note: Increasing the number of concurrent threads improves the rate at which LDIF entries are created, but consumes more system resources.

Optional Arguments	Descriptions
-b	Specifies that you have included binary file names in the data file, which are preceded by a forward slash character. The tool retrieves the actual values from the file referenced.
-c	Tells the tool to proceed in spite of errors. The errors will be reported. (If you do not use this option, the tool stops when it encounters an error.)
-D *binddn*	When authenticating to the directory, specifies doing so as the entry is specified in *binddn*. Use this with the `-w` *`password`* option.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-K	Same as -k, but performs only the first step of the kerberos bind
-k	Authenticates using Kerberos authentication instead of simple authentication. To enable this option, you must compile with KERBEROS defined. You must already have a valid ticket granting ticket.
-n	Shows what would occur without actually performing the operation.
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-T	Sets the number of threads for concurrently processing entries
-U *SSLAuth*	Specifies SSL Authentication Mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *password*	Provides the password required to connect
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapbind Syntax

The ldapbind command line tool enables you to see whether you can authenticate a client to a server.

ldapbind uses this syntax:

```
ldapbind [arguments]
```

Optional Arguments	Descriptions
-D *binddn*	When authenticating to the directory, specifies doing so as the entry specified in *binddn*. Use this with the `-w` *password* option.
-E ".*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-n	Shows what would occur without actually performing the operation
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies the wallet password required for one-way or two-way SSL connections
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-w *password*	Provides the password required to connect
-W *wallet_location*	Specifies wallet location (required for one-way or two-way SSL connections)

ldapcompare Syntax

The ldapcompare command line tool enables you to match attribute values you specify in the command line with the attribute values in the directory entry.

ldapcompare uses this syntax:

```
ldapcompare [arguments]
```

The following example tells you whether `Person Nine`'s title is `associate`.

```
ldapcompare -p 389 -h myhost -b "cn=Person Nine, ou=EuroSInet Suite, o=IMC,
c=US" -a title -v associate
```

Mandatory Arguments	**Descriptions**
-a *attribute name*	Specifies the attribute on which to perform the compare
-b *basedn*	Specifies the distinguished name of the entry on which to perform the compare
-v *attribute value*	Specifies the attribute value to compare

Optional Arguments	**Descriptions**
-D *binddn*	When authenticating to the directory, specifies doing so as the entry is specified in *binddn*. Use this with the `-w` *password* option.
-d *debug-level*	Sets the debugging level. See *Oracle Internet Directory Administrator's Guide*.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-f *filename*	Specifies the input filename
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password (required for one-way or two-way SSL connections)

Optional Arguments	Descriptions
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-w *password*	Provides the password required to connect
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapdelete Syntax

The ldapdelete command line tool enables you to remove entire entries from the directory that you specify in the command line.

ldapdelete uses this syntax:

```
ldapdelete [arguments] "entry_DN"
```

The following example uses port 389 on a host named myhost.

```
ldapdelete -p 389 -h myhost ou=EuroSInet Suite, o=IMC, c=US”
```

Optional Arguments	Descriptions
-D *binddn*	When authenticating to the directory, uses a full DN for the *binddn* parameter; typically used with the `-w` *password* option.
-d *debug-level*	Sets the debugging level. See *Oracle Internet Directory Administrator's Guide*.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-f *filename*	Specifies the input filename
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-k	Authenticates using authentication instead of simple authentication. To enable this option, you must compile with Kerberos defined. You must already have a valid ticket granting ticket.
-n	Shows what would be done, but doesn't actually delete
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *password*	Provides the password required to connect.
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapmoddn Syntax

The ldapmoddn command line tool enables you to modify the DN or RDN of an entry.

ldapmoddn uses this syntax:

```
ldapmoddn [arguments]
```

The following example uses ldapmoddn to modify the RDN component of a DN from `"cn=dcpl"` to `"cn=thanh mai"`. It uses port 389, and a host named myhost.

```
ldapmoddn -p 389 -h myhost -b "cn=dcpl,dc=Americas,dc=imc,dc=com" -R "cn=thanh
mai"
```

Mandatory Argument	Description
-b *basedn*	Specifies DN of the entry to be moved

Optional Arguments	Descriptions
-D *binddn*	When authenticating to the directory, do so as the entry is specified in *binddn*. Use this with the `-w` *password* option.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-f *filename*	Specifies the input filename
-h *ldaphost*	Specifies name of the host node of the directory server
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-N *newparent*	Specifies new parent of the RDN
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-r	Specifies that the old RDN is not retained as a value in the modified entry. If this argument is not included, the old RDN is retained as an attribute in the modified entry.
-R *newrdn*	Specifies new RDN

Optional Arguments	Descriptions
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-w *password*	Provides the password required to connect.
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapmodify Syntax

The ldapmodify tool enables you to act on attributes.

ldapmodify uses this syntax:

```
ldapmodify [arguments] -f filename
```

where *`filename`* is the name of an LDIF file written with the specifications explained the section "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21.

The list of arguments in the following table is not exhaustive.

Optional Arguments	Description
-a	Denotes that entries are to be added, and that the input file is in LDIF format.
-b	Specifies that you have included binary file names in the data file, which are preceded by a forward slash character.
-c	Tells ldapmodify to proceed in spite of errors. The errors will be reported. (If you do not use this option, ldapmodify stops when it encounters an error.)
-D *binddn*	When authenticating to the directory, specifies doing so as the entry is specified in *binddn*. Use this with the `-w` *password* option.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-n	Shows what would occur without actually performing the operation.
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *password*	Overrides the default, unauthenticated, null bind. To force authentication, use this option with the `-D` option.
-W *wallet_location*	Specifies wallet location (required for one-way or two-way SSL connections)

To run `modify`, `delete`, and `modifyrdn` operations using the `-f` flag, use LDIF for the input file format (see "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21) with the specifications noted below:

Always separate entries with a blank line.

Unnecessary space characters in the LDIF input file, such as a space at the end of an attribute value, will cause the LDAP operations to fail.

Line 1: Every change record has, as its first line, the literal `dn:` followed by the DN value for the entry, for example:

```
dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
```

Line 2: Every change record has, as its second line, the literal “`changetype:`” followed by the type of change (`add, delete, modify, modrdn`), for example:

```
changetype:modify
```

or

```
changetype:modrdn
```

Format the remainder of each record according to the following requirements for each type of change:

- `changetype:add`

  Uses LDIF format (see "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21).

- `changetype:modify`

  The lines that follow this changetype consist of changes to attributes belonging to the entry that you identified in Line 1 above. You can specify three different types of attribute modifications—add, delete, and replace—which are explained next:

  – **Add attribute values**. This option to changetype modify adds more values to an existing multi-valued attribute. If the attribute does not exist, it adds the new attribute with the specified values:

    ```
    add: attribute name
    attribute name: value1
    attribute name: value2...
    ```

    For example:

    ```
    dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
    changetype:modify
    add: work-phone
    work-phone:510/506-7000
    work-phone:510/506-7001
    ```

- **Delete values**. If you supply only the “delete” line, all the values for the specified attribute are deleted. Otherwise, if you specify an attribute line, you can delete specific values from the attribute:

  ```
  delete: attribute name
  [attribute name: value1]
  ```

  For example:

  ```
  dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
  changetype:delete
  delete: home-fax
  ```

- **Replace values.** Use this option to replace all the values belonging to an attribute with the new, specified set:

  ```
  replace:attribute name
  [attribute name:value1 ...]
  ```

  If you do not provide any attributes with "replace," the directory adds an empty set. It then interprets the empty set as a delete request, and complies by deleting the attribute from the entry. This is useful if you want to delete attributes that may or may not exist.

  For example:

  ```
  dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
  changetype:modify
  replace: work-phone
  work-phone:510/506-7002
  ```

- `changetype:delete`

  This change type deletes entries. It requires no further input, since you identified the entry in Line 1 and specified a changetype of delete in Line 2.

  For example:

  ```
  dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
  changetype:delete
  ```

- `changetype:modrdn`

  The line following the change type provides the new relative distinguished name using this format:

  ```
  newrdn: RDN
  ```

For example:

```
dn:cn=Barbara Fritchy,ou=Sales,o=Oracle,c=US
changetype:modrdn
newrdn: cn=Barbara Fritchy-Blomberg
```

ldapmodifymt Syntax

The ldapmodifymt command line tool enables you to modify several entries concurrently.

ldapmodifymt uses this syntax:

```
ldapmodifymt -T number_of_threads [arguments] -f filename
```

where *filename* is the name of an LDIF file written with the specifications explained the section "LDAP Data Interchange Format (LDIF) Syntax" on page 2-21.

> **See Also:** "ldapmodify Syntax" on page 2-11 for additional formatting specifications used by ldapmodifymt

For example:

```
ldapmodifymt -T 5 -h node1 -p 3000 -f myentries.ldif
```

Optional Arguments	Descriptions
-a	Denotes that entries are to be added, and that the input file is in LDIF format. (If you are running ldapadd, this flag is not required.)
-b	Specifies that you have included binary file names in the data file, which are preceded by a forward slash character.
-c	Tells ldapmodify to proceed in spite of errors. The errors will be reported. (If you do not use this option, ldapmodify stops when it encounters an error.)
-D *binddn*	When authenticating to the directory, specifies doing so as the entry is specified in binddn. Use this with the `-w` *password* option.
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-h *ldaphost*	Tells ldapmodify to connect to *ldaphost*, rather than to the default directory. *ldaphost* can be an IP address.
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-n	Shows what would occur without actually performing the operation.
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password required for one-way or two-way SSL connections
-T	Sets the number of threads for concurrently processing entries
-U *SSLAuth*	Specifies SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *password*	Overrides the default, unauthenticated, null bind. To force authentication, use this option with the -D option.
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections

ldapsearch Syntax

The ldapsearch command line tool enables you to search for and retrieve specific entries in the directory.

ldapsearch uses this syntax:

```
ldapsearch [arguments] filter [attributes]
```

The *filter* format must be compliant with RFC-2254. For further information about this standard, search for the standard at: http://www.ietf.org/rfc/rfc2254.txt

Separate attributes with a space. If you do not list any attributes, all attributes are retrieved.

Mandatory Arguments	Descriptions
-b *basedn*	Specifies base dn for search
-s *scope*	Specifies search scope: base, one, or sub.

Optional Arguments	Descriptions
-A	Retrieves attribute names only (no values)
-a *deref*	Specifies alias dereferencing: never, always, search, or find
-B	Allows printing of non-ASCII values
-D *binddn*	When authenticating to the directory, specifies doing so as the entry specified in *binddn*. Use this with the `-w` *password* option.
-d *debug level*	Sets debugging level to the level specified. See *Oracle Internet Directory Administrator's Guide*
-E "*character_set*"	Specifies native character set encoding. See *Oracle Internet Directory Administrator's Guide*.
-f *file*	Performs sequence of searches listed in *file*
-F sep	Prints '`sep`' instead of '`=`' between attribute names and values
-h *ldaphost*	Connects to *ldaphost*, rather than to the default host, that is, your local computer. *ldaphost* can be a computer name or an IP address.
-L	Prints entries in LDIF format (`-B` is implied)
-l *timelimit*	Specifies maximum time (in seconds) to wait for ldapsearch command to complete

Optional Arguments	Descriptions
-n	Shows what would be done without actually searching
-p *ldapport*	Connects to the directory on TCP port *ldapport*. If you do not specify this option, the tool connects to the default port (389).
-P *wallet_password*	Specifies wallet password (required for one-way or two-way SSL connections)
-S *attr*	Sorts the results by attribute *attr*
-t	Writes to files in `/tmp`
-u	Includes user friendly entry names in the output
-U *SSLAuth*	Specifies the SSL authentication mode: ■ 1 for no authentication required ■ 2 for one way authentication required ■ 3 for two way authentication required
-v	Specifies verbose mode
-w *passwd*	Specifies bind passwd for simple authentication
-W *wallet_location*	Specifies wallet location required for one-way or two-way SSL connections
-z *sizelimit*	Specifies maximum number of entries to retrieve

Examples of ldapsearch Filters

Study the following examples to see how to build your own search commands.

Example 1: Base Object Search

The following example performs a base-level search on the directory from the root.

```
ldapsearch -p 389 -h myhost -b "" -s base -v "objectclass=*"
```

- `-b` specifies base dn for search, root in this case.
- `-s` specifies whether the search is a base search (`base`), one level search (`one`) or subtree search (`sub`).
- `"objectclass=*"` specifies the filter for search.

Example 2: One-Level Search

The following example performs a one level search starting at `"ou=HR, ou=Americas, o=IMC, c=US"`.

```
ldapsearch -p 389 -h myhost -b "ou=HR, ou=Americas, o=IMC, c=US" -s one -v
"objectclass=*"
```

Example 3: Sub-Tree Search

The following example performs a sub-tree search and returns all entries having a DN starting with `"cn=Person"`.

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "cn=Person*"
```

Example 4: Search Using Size Limit

The following example actually retrieves only two entries, even if there are more than two matches.

```
ldapsearch -h myhost -p 389 -z 2 -b "ou=Benefits,ou=HR,ou=Americas,o=IMC,c=US"
-s one "objectclass=*"
```

Example 5: Search with Required Attributes

The following example returns only the `dn` attribute values of the matching entries.

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "objectclass=*" dn
```

The following example retrieves only the distinguished name (`dn`) along with the surname (`sn`) and description (`description`) attribute values.

```
ldapsearch -p 389 -h myhost -b "c=US" -s sub -v "cn=Person*" dn sn description
```

Other Examples: Each of the following examples searches on port 389 of host sun1, and searches the whole subtree starting from the DN `"ou=hr,o=acme,c=us"`.

The following example searches for all entries with any value for the objectclass attribute.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree "objectclass=*"
```

The following example searches for all entries that have `orcle` at the beginning of the value for the `objectclass` attribute.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree
"objectclass=orcle*"
```

The following example searches for entries where the `objectclass` attribute begins with `orcle` and `cn` begins with foo.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree
"(&(objectclass=orcle*)(cn=foo*))"
```

The following example searches for entries in which the common name (`cn`) is not `foo`.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree "(!(cn=foo))"
```

The following example searches for entries in which `cn` begins with `foo` or `sn` begins with `bar`.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree
"(|(cn=foo*)(sn=bar*))"
```

The following example searches for entries in which `employeenumber` is less than or equal to 10000.

```
ldapsearch -p 389 -h sun1 -b "ou=hr, o=acme, c=us" -s subtree
"employeenumber<=10000"
```

LDAP Data Interchange Format (LDIF) Syntax

The standardized file format for directory entries is as follows:

Property	Value	Description
dn:	*RDN*,*RDN*,*RDN*, ...	Separate RDNs with commas.
attribute:	*attribute_value*	This line repeats for every attribute in the entry, and for every attribute value in multi-valued attributes.
objectClass:	*object_class_ value*	This line repeats for every object class.

The following example shows a file entry for an employee. In this example, the first line contains the DN. The lines that follow the DN begin with the mnemonic for an attribute, followed by the value to be associated with that attribute. Note that each entry ends with lines defining the object classes for the entry.

```
dn: cn=Suzie Smith,ou=Server Technology,o=Acme, c=US
cn: Suzie Smith
cn: SuzieS
sn: Smith
email: ssmith@us.Acme.com
telephoneNumber: 69332
photo:/ORACLE_HOME/empdir/photog/ssmith.jpg
objectClass: organizational person
objectClass: person
objectClass: top
```

The next example shows a file entry for an organization.

```
dn: o=Acme,c=US
o: Oracle
ou: Financial Applications
objectClass: organization
objectClass: top
```

LDIF Formatting Notes

A list of formatting rules follows. This list is not exhaustive.

- All mandatory attributes belonging to an entry being added must be included with non-null values in the LDIF file.

  Tip: To see the mandatory and optional attribute types for an object class, use Oracle Directory Manager. See *Oracle Internet Directory Administrator's Guide* for details.

- Non-printing characters and tabs are represented in attribute values by base-64 encoding.
- The entries in your file must be separated from each other by a blank line.
- A file must contain at least one entry.
- Lines can be continued to the next line by beginning the continuation line with a space or a tab.
- Add a blank line between separate entries.
- Reference binary files, such as photographs, with the absolute address of the file, preceded by a forward slash ("/").
- The DN contains the full, unique directory address for the object.
- The lines listed after the DN contain both the attributes and their values. DNs and attributes used in the input file must match the existing structure of the DIT. Do not use attributes in the input file that you have not implemented in your DIT.
- Sequence the entries in an LDIF file so that the DIT is created from the top down. If an entry relies on an earlier entry for its DN, make sure that the earlier entry is added before its child entry.
- When you define schema within an LDIF file, insert a white space between the opening parenthesis and the beginning of the text, and between the end of the text and the ending parenthesis.

Catalog Management Tool Syntax

Oracle Internet Directory uses indexes to make attributes available for searches. When Oracle Internet Directory is installed, the entry `cn=catalogs` lists available attributes that can be used in a search. Only those attributes that have an equality matching rule can be indexed.

If you want to use additional attributes in search filters, you must add them to the catalog entry. You can do this at the time you create the attribute by using Oracle Directory Manager. However, if the attribute already exists, then you can index it only by using the Catalog Management tool.

Before running the Catalog Management tool, unset the LANG variable. After you finish running Catalog Management tool, set the LANG variable back to its original value.

To unset LANG:

- Using Korn shell:

  ```
  UNSET LANG
  ```

- Using C shell:

  ```
  UNSETENV LANG
  ```

The Catalog Management tool uses this syntax:

```
catalog.sh -connect net_service_name {add|delete} {-attr attr_name|-file
filename}
```

Mandatory Argument	Description
- connect *net_service_name*	Specifies the net service name to connect to the directory database **See Also:** *Net8 Administrator's Guide*

Optional Arguments	Descriptions
- add -attr *attr_name*	Indexes the specified attribute
- delete -attr *attr_name*	Drops the index from the specified attribute
- add -file *filename*	Indexes attributes (one per line) in the specified file
-delete -file *filename*	Drops the indexes from the attributes in the specified file

When you enter the `catalog.sh` command, the following message appears:

```
This tool can only be executed if you know the OiD user password.
Enter OiD password:
```

If you enter the correct password, the command is executed. If you give an incorrect password, the following message is displayed:

```
Cannot execute this tool
```

After you finish running the Catalog Management tool, set the LANG variable back to its original value.

To set LANG:

- Using Korn shell:

  ```
  SET LANG=appropriate_language, EXPORT LANG
  ```

- Using C shell:

  ```
  SETENV LANG appropriate_language
  ```

To effect the changes after running the Catalog Management tool, stop, then restart, the Oracle directory server.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for instructions on starting and restarting directory servers

Index