

Novell Developer Kit

www.novell.com

June21, 2006

LDAP LIBRARIES FOR C



Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2006 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.novell.com/company/legal/patents/> and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

For Novell trademarks, see [Novell Trademark and Service Mark list \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html)

Third-Party Materials

All third-party trademarks are the property of their respective owners

Contents

Preface	13
1 Concepts	15
1.1 Getting Started	15
1.1.1 Dependencies	15
1.1.2 Platform Libraries and Header Files	16
1.1.3 Supported Platforms	17
1.1.4 Supported Compilers	17
1.1.5 Tutorials	18
1.1.6 Sample Code	18
1.1.7 LDAP Test Server	18
1.2 Using the LDAP Functions	18
1.2.1 Using Dynamic Memory with LDAP Functions	19
1.2.2 Selecting a Function for an LDAP Operation	21
1.2.3 Using Asynchronous or Synchronous Functions	21
1.2.4 Initializing a Session with LDAP v3	22
1.2.5 Setting Initial Connection Timeout	22
1.2.6 Setting and Getting the Cipher Level	23
1.2.7 LDAP URLs	23
1.2.8 Threads	24
1.2.9 Internationalization	25
1.3 Authentication and Security	26
1.3.1 Setting Up SSL Security	26
1.3.2 Authentication	27
1.3.3 SSL Certificates	31
1.3.4 Transport Layer Security	33
1.3.5 Recommendations	33
1.4 LDAP Searches	34
1.4.1 Setting the Search Parameters and Search Constraints	34
1.4.2 Using Search Filters	37
1.4.3 Operational Attributes	39
1.5 LDAP Based Backup	39
1.6 Referral Handling in LDAP v3	40
1.6.1 Configuring eDirectory to Return Complete Data	40
1.6.2 Configuring eDirectory to Return Referrals	40
1.6.3 Enabling Referral Handling in the Application	40
1.6.4 Creating a Rebind Process	41
1.6.5 Using the Rebind Process	41
1.6.6 Following Referrals Manually	42
1.6.7 Retrieving Referrals for Non-Search Operations	42
1.6.8 Limiting Referral Hops	42
1.7 eDirectory Event System	43
1.7.1 Registering to Monitor an Event	43
1.7.2 LBURP	43
1.8 Character Conversions	44
1.8.1 A Brief History of Character Encoding	44
1.8.2 UTF-8 Encoding	45
1.8.3 UTF-8	45
1.8.4 wchar_t Type	46
1.9 Time Formats	46
1.10 Controls and Extensions	46

1.10.1	Controls	47
1.10.2	Extensions	47
1.11	Runtime Version of the Library Files	49
1.11.1	Windows (NT, 95, 98, 2000, XP)	49
1.11.2	NetWare	50
1.11.3	UNIX (Solaris, Linux, AIX, HP-UX)	51
1.12	Internationalization	52
1.12.1	File Locations	52
1.12.2	Language Directory Names	52
2	Tasks	55
2.1	Establishing an SSL Connection	55
2.2	Reading the Root DSE	55
2.3	Adding an Entry	56
2.4	Modifying an Entry	57
2.5	Modifying an Entry's Password	57
2.6	Extending the Schema	58
3	Standard LDAP Functions	59
ber_alloc_t		60
ber_bvdup		61
ber_bvecfree		62
ber_bvfree		63
ber_first_element		64
ber_flatten		65
ber_free		66
ber_init		67
ber_next_element		68
ber_peek_tag		69
ber_printf		70
ber_scanf		72
ber_skip_tag		75
ldap_abandon		76
ldap_abandon_ext		78
ldap_add		80
ldap_add_ext		82
ldap_add_ext_s		84
ldap_add_s		87
ldap_bind		89
ldap_bind_digest_md5_start		91
ldap_bind_digest_md5_finish		93
ldap_bind_nmas_s		95
ldap_bind_s		97
ldap_cancel_ext		99
ldap_cancel_ext_s		101
ldap_compare		103
ldap_compare_ext		105
ldap_compare_ext_s		107
ldap_compare_s		109
ldap_control_free		111
ldap_controls_free		112

ldap_count_entries	113
ldap_count_messages	115
ldap_count_references	117
ldap_count_values	119
ldap_count_values_len	120
ldap_create_geteffective_control	121
ldap_create_persistentsearch_control	123
ldap_create_reference_control	125
ldap_create_sort_control	126
ldap_create_sort_keylist	128
ldap_create_sstatus_control	130
ldap_create_vlv_control	131
ldap_delete	133
ldap_delete_ext	134
ldap_delete_ext_s	136
ldap_delete_s	138
ldap_destroy	139
ldap_dn2ufn	140
ldap_dup	141
ldap_err2string	142
ldap_explode_dn	144
ldap_explode_rdn	146
ldap_extended_operation	147
ldap_extended_operation_s	149
ldap_first_attribute	151
ldap_first_entry	153
ldap_first_message	155
ldap_first_reference	156
ldap_free_sort_keylist	157
ldap_free_urldesc	158
ldap_get_dn	159
ldap_get_digest_md5_realms	160
ldapssl_install_routines	161
ldap_get_entry_controls	162
ldap_get_lderrno	164
ldap_get_option	166
ldap_get_values	167
ldap_get_values_len	169
ldap_gssbind	171
ldap_gss_error	173
ldap_init	174
ldap_is_ldap_url	176
ldap_is_ldaps_url	177
ldap_memfree	178
ldap_modify	179
ldap_modify_ext	181
ldap_modify_ext_s	183
ldap_modify_s	185
ldap_modrdn	186
ldap_modrdn_s	188
ldap_modrdn2	190
ldap_modrdn2_s	192

ldap_msgfree	194
ldap_msgid	195
ldap_msgtype	196
ldap_multisort_entries	197
ldap_next_attribute	199
ldap_next_entry	201
ldap_next_message	203
ldap_next_reference	204
ldap_open	205
ldap_parse_entrychange_control	206
ldap_parse_extended_result	208
ldap_parse_intermediate	210
ldap_parse_reference	212
ldap_parse_reference_control	214
ldap_parse_result	216
ldap_parse_sasl_bind_result	219
ldap_parse_sort_control	221
ldap_parse_sstatus_control	223
ldap_parse_vlv_control	224
ldap_perror	226
ldap_rename	227
ldap_rename_s	229
ldap_result	231
ldap_result2error	233
ldap_sasl_bind	235
ldap_sasl_bind_s	237
ldap_schema_fetch	239
ldap_schema_free	240
ldap_schema_get_by_name	241
ldap_schema_get_count	242
ldap_schema_get_by_index	243
ldap_schema_get_field_names	245
ldap_schema_get_field_values	246
ldap_schema_add	247
ldap_schema_modify	248
ldap_schema_delete	249
ldap_schema_save	250
ldap_search	251
ldap_search_ext	253
ldap_search_ext_s	256
ldap_search_s	259
ldap_search_st	261
ldap_set_lderrno	264
ldap_set_option	266
ldap_set_rebind_proc	268
ldap_simple_bind	270
ldap_simple_bind_s	272
ldap_sort_entries	274
ldap_sort_strcasecmp	276
ldap_sort_values	277
ldap_unbind, ldap_unbind_s	278
ldap_unbind_ext, ldap_unbind_ext_s	279

ldap_url_desc2str	281
ldap_url_parse	282
ldap_url_parse_ext	284
ldap_url_search	286
ldap_url_search_s	288
ldap_url_search_st	290
ldap_value_free	292
ldap_value_free_len	293
ldapssl_client_init	294
ldapssl_client_deinit	296
ldapssl_init	297
ldapssl_add_trusted_cert	299
ldapssl_get_cert	301
ldapssl_get_cert_attribute	303
ldapssl_set_verify_mode	305
ldapssl_set_client_cert	306
ldapssl_set_client_private_key	308
ldapssl_get_verify_mode	310
ldapssl_set_verify_callback	311
ldapssl_start_tls	313
ldapssl_stop_tls	314

4 LDAP Extension Functions 315

ldap_abort_partition_operation	316
ldap_add_replica	318
ldap_backup_object	320
ldap_change_replica_type	322
ldap_create_partition	324
ldap_create_orphan_partition	326
ldap_event_free	328
ldap_get_bind_dn	329
ldap_get_effective_privileges	331
ldap_get_replication_filter	333
ldap_get_replica_info	335
ldap_lburp_end_request	337
ldap_lburp_operation_request	338
ldap_lburp_parse_operation_response	340
ldap_lburp_start_request	341
ldap_list_replicas	342
ldap_merge_partitions	344
ldap_monitor_events	346
ldap_monitor_events_filtered	348
ldap_parse_ds_event	350
ldap_parse_lburp_end_response	352
ldap_parse_lburp_start_response	354
ldap_parse_monitor_events_response	356
ldap_partition_entry_count	358
ldap_nds_to_ldap	360
ldap_receive_all_updates	362
ldap_refresh_server	364
ldap_remove_orphan_partition	366

ldap_remove_replica	368
ldap_request_partition_sync	370
ldap_request_schema_sync	372
ldap_restore_object	374
ldap_send_all_updates	376
ldap_set_replication_filter	378
ldap_split_orphan_partition	380
ldap_split_partition	382
ldap_trigger_back_process	384
ldapx_memfree	386
 5 UTF-8 Functions	 387
5.1 UTF-8 / Wide Character Conversions	387
ldap_x_utf8_to_wc	388
ldap_x_utf8s_to_wcs	389
ldap_x_wc_to_utf8	391
ldap_x_wcs_to_utf8s	392
5.2 UTF-8 Utility Functions	393
ldap_x_utf8_chars	394
ldap_x_utf8_charlen	395
ldap_x_utf8_charlen2	396
ldap_x_utf8_next	397
ldap_x_utf8_prev	398
ldap_x_utf8_copy	399
ldap_x_utf8_strchr	400
ldap_x_utf8_strspn	401
ldap_x_utf8_strcspn	402
ldap_x_utf8_strpbrk	403
ldap_x_utf8_strtok	404
 6 Values	 407
6.1 Object Access Control Rights	407
6.2 Attribute Access Control Rights	407
6.3 Certificate Attribute IDs	408
6.4 Inheritance Control Rights	409
6.5 Replica States	409
6.6 Replication Filters	411
6.7 Replica Types	411
6.8 Request Message Types	412
6.9 Result Message Types	412
6.10 Session Preference Options	413
6.11 Schema Element Types	418
6.11.1 LDAP_SCHEMA_ATTRIBUTE_TYPE	418
6.11.2 LDAP_SCHEMA_OBJECT_CLASS	420
6.11.3 LDAP_SCHEMA_MATCHING_RULE	421
6.11.4 LDAP_SCHEMA_MATCHING_RULE_USE	422
6.11.5 LDAP_SCHEMA_NAME_FORM	422
6.11.6 LDAP_SCHEMA_SYNTAX	423
6.11.7 LDAP_SCHEMA_DIT_CONTENT_RULE	423
6.11.8 LDAP_SCHEMA_DIT_STRUCTURE_RULE	423
6.12 SSL Certificate Status Codes	424

7 Structures	427
BerElement	428
berval	429
DB_binary	430
DB_netAddress	431
DB_Parameter	432
DB_timeStampVector	433
DB_value	434
EVT_BinderyObjectInfo	435
EVT_ChangeConfigParm	436
EVT_ChangeConnState	438
EVT_ChangeServerAddr	439
EVT_DebugInfo	440
EVT_EntryInfo	442
EVT_EventData	443
EVT_EventSpecifier	444
EVT_FilteredEventSpecifier	445
EVT_ModuleState	447
EVT_NetAddress	449
EVT_ReferralAddress	450
EVT_SEVInfo	451
EVT_TimeStamp	452
EVT_ValueInfo	453
LBURPUpdateResult	454
LBURPUpdateOperationList	455
LDAP	456
LDAP_DIGEST_MD5_CONTEXT	457
LDAPAPIFeatureInfo	458
LDAPAPIInfo	459
LDAPControl	460
LDAPMessage	461
LDAPMod	462
LDAPReplicaInfo	464
LDAPSchema	465
LDAPSchemaElement	466
LDAPSchemaMod	467
LDAPSortKey	468
LDAPSSL_Cert	469
LDAPSSL_Cert_VValidity_Period	470
LDAPURLDesc	472
LDAPVLVInfo	474
timeval	476
 A Source Code Contributors	 479
 B Revision History	 481

Preface

The LDAP Libraries for C enable you to write applications to access, manage, update, and search for information stored in Novell® eDirectory™ and other LDAP-aware directories.

LDAP (Lightweight Directory Access Protocol) is becoming an Internet standard for accessing directory information, allowing LDAP-enabled applications to access multiple directories. LDAP v3 supports such features as secure connections (through SSL and SASL), entry management, schema management, and LDAP controls and extensions for expanding the functionality of LDAP.

The LDAP Libraries for C are available for the following platforms:

- Windows* (NT*, 95, 98, 2000, XP)
- NetWare®
- Unix* (Solaris*, Linux*, AIX*, and HP-UX*)

This guide contains the following sections:

- **Concepts**
- **Tasks**
- **Standard LDAP Functions**
- **LDAP Extension Functions**
- **Values**
- **Structures**
- **Source Code Contributors**
- **Revision History**

Audience

This guide is intended for C developers who desire to write applications to access, manage, update, and search for information stored in Novell eDirectory and other LDAP-aware directories.

Feedback

We want to hear your comments and suggestions about this manual. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

Documentation Updates

For the most recent version of this guide, see [LDAP Libraries for C \(http://developer.novell.com/ndk/cldap.htm\)](http://developer.novell.com/ndk/cldap.htm).

Additional Documentation

For the most recent version of NDK guides, see [NDK Download Wibe site \(http://developer.novell.com/ndk/downloadaz.htm\)](http://developer.novell.com/ndk/downloadaz.htm).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol ([®], [™], etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX*, should use forward slashes as required by your software.

This manual assumes that you have a basic understanding of Novell® eDirectory™ and eDirectory and LDAP integration. For more information on these topics, see

- *Developer Kit.*
- *NDK: LDAP and eDirectory Integration.*

This manual consists of the following:

- Section 1.1, “Getting Started,” on page 15
- Section 1.2, “Using the LDAP Functions,” on page 18
- Section 1.3, “Authentication and Security,” on page 26
- Section 1.4, “LDAP Searches,” on page 34
- Section 1.5, “LDAP Based Backup,” on page 39
- Section 1.6, “Referral Handling in LDAP v3,” on page 40
- Section 1.7, “eDirectory Event System,” on page 43
- Section 1.8, “Character Conversions,” on page 44
- Section 1.9, “Time Formats,” on page 46
- Section 1.10, “Controls and Extensions,” on page 46
- Section 1.11, “Runtime Version of the Library Files,” on page 49
- Section 1.12, “Internationalization,” on page 52

1.1 Getting Started

The following sections cover a few basic requirements for getting set up and started with the LDAP Libraries for C:

- “Dependencies” on page 15
- “Platform Libraries and Header Files” on page 16
- “Supported Platforms” on page 17
- “Supported Compilers” on page 17
- “Tutorials” on page 18
- “Sample Code” on page 18

1.1.1 Dependencies

In addition to LDAP Libraries for C, you need the following to take full advantage of the functionality offered in the libraries:

- **LDAP Server.** The libraries can be used to access any LDAP server and its directory. If you are using them to access eDirectory, the LDAP server must be running on NDS® 7.xx or higher to access LDAP v3 functionality. Other servers in the tree can be running earlier versions of NDS;

only the LDAP server needs to be on NDS 7.xx or higher. For information on the functionality available in various versions of NDS/eDirectory, see **NDK: LDAP and eDirectory Integration**.

- **SSL.** To use SSL, the LDAP server and the LDAP client must be configured for SSL. For more information, see **Section 1.3, “Authentication and Security,”** on page 26.
- **LDAP Extensions for eDirectory.** To use the LDAP extensions for partition and replica management, and getting effective rights, the LDAP server must be running on eDirectory 8.5 or higher. To obtain a copy, see **Novell eDirectory evaluation site** (<http://www.novell.com/products/edirectory/evaluation.html>).

1.1.2 Platform Libraries and Header Files

The LDAP Libraries for C includes the following header files.

Table 1-1 Header File Description

Header File	Description
ldap.h	Contains the prototypes for all the standard LDAP functions
ldapx.h	Contains the prototypes for LDAP functions for extensions
ldapssl.h	Contains the prototypes for all of the LDAP SSL functions
ldaputf8.h	Contains the prototypes for all of the UTF-8 conversions routines

The following header files are included in the LDAP libraries for C, but are linked by the header files listed in the previous table:

- lber.h
- lber_types.h
- ldap_cdefs.h
- ldap_features.h

The LDAP Libraries for C have been compiled into the following libraries (UNIX* platforms add a lib prefix to the library names):

Table 1-2 Platform Libraries

Library	Platforms
ldapsdk.dll	Win32 platforms (Windows* 95, Windows 98, Windows 2000, Windows NT* server with SP 4 or newer, Windows NT workstation with SP 3 or SP 4)
ldapx.dll	
ldapssl.dll	
nmas.dll	NOTE: nmas.dll is used to perform a bind using Novell Modular Authentication Services .

Library	Platforms
ldapsdk.nlm	NetWare® 6 with eDirectory 8.6 (or higher), NetWare 5 with SP4 and NDS 8.2x, NetWare 5.1 with NDS 8.3x
ldapx.nlm	
ldapssl.nlm	
libldapsdk.so	Solaris* (2.6 and 2.7), Linux* (Red Hat 7.2), and AIX*
libldapx.so	
libldapssl.so	
libldapsdk.sl	HP-UX* (11.11)
libldapx.sl	
libldapssl.sl	

1.1.3 Supported Platforms

The LDAP Libraries for C SDK enables application developers to write applications to access, manage, update and search for information stored in eDirectory and other LDAP-aware directories.

Client applications remotely access directory information stored on an LDAP server. The libraries currently support development of such applications on the following platforms:

- NetWare 6™
- NetWare 5.1™
- Windows NT* workstation 4.0 with SP 3 and SP 4
- Windows 95*
- Windows 98*
- Windows 2000*
- Windows XP*
- Solaris 2.8*
- Linux (tested on Red Hat 7.2*)
- AIX 4.3*
- HP-UX 11.11*

The Novell LDAP server is currently available on NetWare 6, NetWare 5.x, and eDirectory (for NetWare, NT, Solaris, Linux, and AIX).

1.1.4 Supported Compilers

The libraries can be used with the following C compilers:

- Microsoft Visual Studio C++ versions 5 and 6 for Windows
- Borland C Compiler for Windows
- CodeWarrior from Metrowerks for NLM
- Watcom for NLM

- GCC 2.95.2 for Linux
- Solaris vendor-supplied compiler (built using Sun Workshop Compiler 5.0)
- AIX vendor-supplied compiler (built using AIX Compiler version 5)
- CC compiler for HP-UX

1.1.5 Tutorials

DeveloperNet University has developed an LDAP tutorial that creates a White Pages application. This application is similar to looking up information in the white pages of a phone book and allows users to browse and search for employees in a company, view their pictures, and obtain phone numbers, titles, and other information.

To access this tutorial, see [Programming NDS with C LDAP \(http://developer.novell.com/education/tutorials/whitepages/index.htm\)](http://developer.novell.com/education/tutorials/whitepages/index.htm).

1.1.6 Sample Code

To access LDAP sample code, check the following sites:

- For source code examples that use the standard functions for LDAP operations (such as search, add, modify, and delete), see [the LDAP sample code for C \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).
- For source code examples that use the Novell LDAP extensions for LDAP partition operations (such as add, modify, and delete replicas or splitting and joining partitions), see [the LDAP sample code for extensions \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).
- For source code and task examples that authenticate, search, and read, see the LDAP examples in the Library and Laboratory departments of [DeveloperNet University \(http://developer.novell.com/education/index.html\)](http://developer.novell.com/education/index.html).

1.1.7 LDAP Test Server

Novell has set up an LDAP server that you can access over the Internet to test your LDAP application. This server's name is www.nldap.com, and it listens on the default LDAP port (389). To use authenticated access, you must set up your own account in your own eDirectory container. Your account is limited to 1MB of disk storage.

To access this site, go to the [NDS/LDAP Services Access Test Site \(http://www.nldap.com/NLDAP\)](http://www.nldap.com/NLDAP).

1.2 Using the LDAP Functions

The following sections discuss some general concepts for using the functions to accomplish a task and some principles for selecting the appropriate function for the task. It covers the following topics:

- [Section 1.2.1, "Using Dynamic Memory with LDAP Functions," on page 19](#)
- [Section 1.2.2, "Selecting a Function for an LDAP Operation," on page 21](#)
- [Section 1.2.3, "Using Asynchronous or Synchronous Functions," on page 21](#)
- [Section 1.2.4, "Initializing a Session with LDAP v3," on page 22](#)

- [Section 1.2.5, “Setting Initial Connection Timeout,” on page 22](#)
- [Section 1.2.6, “Setting and Getting the Cipher Level,” on page 23](#)
- [Section 1.2.7, “LDAP URLs,” on page 23](#)
- [Section 1.2.8, “Threads,” on page 24](#)
- [Section 1.2.9, “Internationalization,” on page 25](#)

1.2.1 Using Dynamic Memory with LDAP Functions

If your application allocates any memory for use with LDAP functions, that memory must be freed by your application. Do not free this memory using an LDAP function, for example `ldap_memfree`.

All memory allocated by LDAP functions must be freed by LDAP functions. The following table lists the LDAP functions which allocate memory and the LDAP function you should use to free the memory.

Table 1-3 *Dynamic Memory with LDAP Functions*

Functions That Allocate Memory	Parameter/Structure Member	Memory Free Functions
<code>ber_alloc_t</code>	<code>return(BerElement *)</code>	<code>ber_free</code>
<code>ber_bvdup</code>	<code>return(struct berval *)</code>	<code>ber_bvfree</code>
<code>ber_flatten</code>	<code>bvptr</code>	<code>ber_bvfree</code>
<code>ber_init</code>	<code>return(BerElement *)</code>	<code>ber_free</code>
<code>ber_scanf</code>	"a" format option	<code>ldap_memfree</code>
<code>ber_scanf</code>	"B" format option	<code>ldap_memfree</code>
<code>ber_scanf</code>	"O" format option	<code>ber_bvfree</code>
<code>ber_scanf</code>	"V" format option	<code>ber_bvecfree</code>
<code>ber_scanf</code>	"v" format option	<code>ldap_memfree</code>
<code>ldap_create_sort_control</code>	<code>ctrlp</code>	<code>ldap_control_free</code>
<code>ldap_create_sort_keylist</code>	<code>sortKeyList</code>	<code>ldap_free_sort_keylist</code>
<code>ldap_create_vlv_control</code>	<code>ctrlp</code>	<code>ldap_control_free</code>
<code>ldap_dn2ufn</code>	<code>return(char *)</code>	<code>ldap_memfree</code>
<code>ldap_explode_dn</code>	<code>return(char **)</code>	<code>ldap_value_free</code>
<code>ldap_explode_rdn</code>	<code>return(char **)</code>	<code>ldap_value_free</code>
<code>ldap_extended_operation_s</code>	<code>retdatap</code>	<code>ber_bvfree</code>
<code>ldap_extended_operation_s</code>	<code>retoidp</code>	<code>ldap_memfree</code>
<code>ldap_first_attribute</code>	<code>return (char *)</code>	<code>ldap_memfree</code>
<code>ldap_first_attribute</code>	<code>BerElement</code>	<code>ber_free (xxx, 0)</code>
<code>ldap_get_context_identity_name</code>	<code>identity</code>	<code>ldappx_memfree</code>

Functions That Allocate Memory	Parameter/Structure Member	Memory Free Functions
ldap_get_dn	return (char *)	ldap_memfree
ldap_get_entry_controls	LDAPControl	ldap_controls_free
ldap_get_option	LDAP_OPT_API_INFO / ldapai_extensions	ldap_value_free
ldap_get_option	LDAP_OPT_API_INFO/ ldapai_vendor_name	ldap_memfree
ldap_get_option	LDAP_OPT_MATCHED_DN / outvalue	ldap_memfree ldap_controls_free
ldap_get_values	return (char **)	ldap_value_free
ldap_get_values_len	return(struct berval **)	ldap_value_free_len
ldap_init	return(LDAP *)	ldap_unbind, ldap_unbind_s
ldap_list_replicas	replicaList	ldapx_memfree
ldap_nds_to_ldap_name	ldapName	ldapx_memfree
ldap_next_attribute	return (char *)	ldap_memfree
ldap_next_attribute	BerElement	ber_free (xxx, 0)
ldap_parse_extended_result	retdatap	ber_bvfree
ldap_parse_extended_result	retoidp	ldap_memfree
ldap_parse_reference	referralsp	ldap_value_free
ldap_parse_reference	serverctrlsp	ldap_controls_free
ldap_parse_result	errmsgsp	ldap_memfree
ldap_parse_result	matcheddn	ldap_memfree
ldap_parse_result	referralsp	ldap_value_free
ldap_parse_result	serverctrlsp	ldap_controls_free
ldap_parse_sasl_bind_result	servercredp	ber_bvfree
ldap_parse_sort_control	attribute	ldap_memfree
ldap_parse_vlv_control	contextp	ber_bvfree
ldap_result	res	ldap_msgfree
ldap_sasl_bind_s	servercredp	ber_bvfree
ldap_search_ext_s	res	ldap_msgfree
ldap_search_s	res	ldap_msgfree
ldap_search_st	res	ldap_msgfree
ldap_ssl_client_init		ldap_ssl_client_deinit
ldap_ssl_init	return(LDAP *)	ldap_unbind
ldap_url_parse	ludpp	ldap_free_urldesc

Functions That Allocate Memory	Parameter/Structure Member	Memory Free Functions
ldap_url_parse_ext	ludpp	ldap_free_urldesc
ldap_url_search_s	res	ldap_msgfree
ldap_url_search_st	res	ldap_msgfree
ldap_url_desc2str	return(char *)	ldap_memfree

The following functions free the memory allocated to the res parameter if the freeit parameter is set to true:

```

ldap_parse_extended_result
ldap_parse_reference
ldap_parse_result
ldap_parse_sasl_bind_result
ldap_result2error

```

1.2.2 Selecting a Function for an LDAP Operation

Most LDAP functions that perform operations (such as add, delete, modify) have four variations:

- LDAP v2 asynchronous. These take the format of *ldap_operation*, for example, *ldap_search*.
- LDAP v3 asynchronous. These take the format of *ldap_operation_ext*, for example, *ldap_search_ext*.
- LDAP v2 synchronous. These take the format of *ldap_operation_s*, for example, *ldap_search_s*.
- LDAP v3 synchronous. These take the format of *ldap_operation_ext_s*, for example, *ldap_search_ext_s*.

If you are developing a new application, you should use the LDAP v3 version of the functions. The LDAP library supports the LDAP v2 versions for backwards compatibility with earlier releases.

1.2.3 Using Asynchronous or Synchronous Functions

Blocking versus Non-Blocking. Synchronous functions block and do not return until the LDAP server has serviced the request and returned a result. Asynchronous functions return as soon as the LDAP client processes the request, and the application is then free to do other work. However, the application is responsible to use the returned message ID to check on the status of the operation.

Return Values. The synchronous functions return both client and server error codes. The asynchronous *ldap_*_ext* functions return only the client error codes. The subsequent results must be parsed for the server error codes. The asynchronous *ldap_** functions return a -1 for the client error codes, and the *ldap_get_option* function must be used to retrieve the client error codes from the LDAP session handle.

1.2.4 Initializing a Session with LDAP v3

By default, the LDAP v2 functions set up an LDAP v2 session because the session handle is configured for an LDAP v2 session. To ensure that your application sets up an LDAP v3 session, call the following functions in the order specified. The following example uses `ldap_simple_bind`.

- 1 Call the `ldap_set_option` function with the `ld` parameter set to `NULL` and the option parameter set to `LDAP_OPT_PROTOCOL_VERSION`, and the invalue parameter set to `LDAP_VERSION3`.

This sets the value in the global session handle to LDAP v3 and all subsequent session handles assume these values.

- 2 Call the `ldap_init` function.
- 3 Call the `ldap_simple_bind` or `ldap_simple_bind_s` function.

NOTE: This example uses clear text passwords. When you are ready to set up a secure connection, see [“Authentication” on page 27](#).

1.2.5 Setting Initial Connection Timeout

Setting an initial connection timeout enables you to control the amount of time your application will wait for an initial connection to succeed. If a server does not respond and no initial connection timeout option is specified, timeout depends upon the underlying socket timeout setting of the operating system.

By setting an initial connection timeout, you can control how long your application will wait for an initial connection, then possibly attempt a connection to another server or wait and attempt a connection at another time.

An initial connection timeout is set using the `LDAP_OPT_NETWORK_TIMEOUT` option (set by calling [`ldap_set_option` \(page 266\)](#)). The initial connection usually happens on the Bind command, whether it's synchronous or asynchronous; simple, SASL, NMAS, or digest-md5. If no bind command is given, the initial connection happens on the first LDAP operation. An initial connection may also occur during a referral or rebind operation.

The following example sets an initial connection timeout of 10 seconds:

```
struct timeval timeOut = {10, 0};
ldap_set_option( NULL, LDAP_OPT_NETWORK_TIMEOUT, &timeOut);
```

Passing `NULL` for the `ld` parameter to `ldap_set_option` will set this as the default connection timeout for subsequent session handles created with `ldap_init()` or `ldapssl_init()`. To clear the timeout, pass `NULL` for the timeout argument to `ldap_set_option`.

A connection timeout will cause an `LDAP_SERVER_DOWN` error (81) "Can't contact LDAP server".

Using the connection timeout, you can specify multiple hosts separated by spaces in a bind call, and use this timeout to determine how long your application waits for an initial response before attempting a connection to the next host in the list. The following example sets an initial connection timeout of 5 seconds and multiple hosts in the bind call:

```
struct timeval timeOut = {5,0};
ldap_set_option( NULL, LDAP_OPT_NETWORK_TIMEOUT, &timeOut);
ld = ldap_init("www.acme.com 129.233.80.5 127.0.0.1", 389);
```

1.2.6 Setting and Getting the Cipher Level

There are four possible combinations of cipher that can be set. The following table provides the details:

Table 1-4 Details of the Cipher Level

Cipher Value	Key Strength	Algorithm
LDAP_TLS_CIPHER_LOW	56	Single DES
LDAP_TLS_CIPHER_MEDIUM	128	RSA
LDAP_TLS_CIPHER_HIGH	168	Triple DES
LDAP_TLS_CIPHER_EXPORT	56	SHA

By default, the cipher is set to high (triple DES). If the you want to set any of the above mentioned cipher value, call the following functions in the order specified. The following example uses `ldap_simple_bind`.

- 1 Call the `ldap_set_option` function with the `ld` parameter set to `NULL` and the option parameter set to `LDAP_OPT_TLS_CIPHER_LEVEL`, and the invalue parameter set to `LDAP_TLS_CIPHER_MEDIUM`.

This sets the value in the global session handle to key strength 128, algorithm RSA, and all subsequent session handles assume these values.

- 2 Call the `ldapsl_init` function.
- 3 Call the `ldap_simple_bind` or `ldap_simple_bind_s` function.

NOTE: After the bind operation is complete, the application can retrieve the cipher settings used during SSL connection by using the `ldap_get_option`.

1.2.7 LDAP URLs

LDAP URLs provide a uniform method to access information on an LDAP server. Defined in RFC 2255, LDAP URLs begin with the prefix `ldap://` or `ldaps://`. The following provides the syntax and descriptions of an LDAP URL.

```
ldap[s]://<hostname>:<port>/
<base_dn>?<attributes>?<scope>?<filter>?<extension>
```

TIP: `ldaps` is a common enhancement used to denote SSL, and is not defined in an RFC.

In the LDAP Libraries for C, LDAP URLs are used to:

- Return referrals or search references from a server
- Perform searches ([ldap_url_search](#) (page 286))

Table 1-5 Field descriptions for an LDAP URL

URL Element	Default Value	Description
hostname	none	DNS name or IP address of the LDAP server.
port	389	Port of the LDAP server.
base_dn	root	Base DN for the LDAP operation.
attributes	all attributes	A comma-delimited list of attributes to return.
scope	base	Search scope: base, one, sub.
filter	objectClass=*	Search filter.
extension	none	LDAP extended operations.

NOTE: An attribute list is required if you want to provide a scope (even if the attribute list is blank). To return all attributes within a specific scope you must include <base_dn>??<scope>.

Determining if a URL is an LDAP URL

To determine if a URL is a valid ldap:// or ldaps:// URL use one of the following functions:

- [ldap_is_ldap_url \(page 176\)](#)
- [ldap_is_ldaps_url \(page 177\)](#)

Both functions take a URL as the parameter and return 1 if the URL is a valid LDAP URL, and 0 if it is not valid.

Parsing an LDAP URL

The [ldap_url_parse \(page 282\)](#) function parses an LDAP URL and returns an [LDAPURLDesc \(page 472\)](#) structure to your application. You can then retrieve the individual parameters from the URL, or you can pass this URL to a search function.

Searching with an LDAP URL

The [ldap_url_search \(page 286\)](#) functions allow you to pass an [LDAPURLDesc](#) structure to perform an LDAP search.

1.2.8 Threads

The LDAP libraries for C APIs are operation thread safe. This allows different threads within an application to concurrently use the same LDAP session handle for different operations.

Applications using this feature need to duplicate the session handle using the [ldap_dup \(page 141\)](#) function. The returned session handle may be used concurrently with the original session handle. To destroy the session handle use the [ldap_destroy \(page 139\)](#) function.

The following example uses [ldap_dup](#) and [ldap_destory](#).

1. Call the [ldap_init \(page 174\)](#) function.

2. Call the `ldap_simple_bind` (page 270) or `ldap_simple_bind_s` (page 272) function.
3. Duplicate the session handle using `ldap_dup` (page 141).
4. Use the duplicated session handle in a separated thread to do any LDAP operation like add, search, or modify.
5. Close the duplicated session handle using `ldap_destroy` (page 139) in the same thread.
6. Use the original LDAP handle in the main thread to do any LDAP operation like add, search, or modify.
7. Use the `LDAP_OPT_SESSION_REFCNT` to get reference count associated with the supplied session handle.
8. Call the `ldap_unbind` function.

For more information, refer to the `theadSafe.c` sample program.

1.2.9 Internationalization

The LDAP libraries have been enabled for internationalization. However, the messages are currently available only in English. For cross-platform support, the English messages have been placed in the following files:

Table 1-6 *Internationalization File Name on Different Platforms*

File Name	Platform
<code>ldapsdk.msg</code>	NetWare
<code>ldapsdkmsg.dll</code>	Windows (NT, 95, 98, 2000)
<code>ldapsdk.mo</code>	Solaris (2.6, 2.7, 2.8), Linux (RedHat 7.2), AIX and HP-UX (11.11)

Depending upon the platform, the message file is installed in the following locations:

- On a NetWare server in the `sys:\system\nls\4` directory.
- On an eDirectory for NT server in the `winnt\system32\nls\english` directory.
- On a Windows client in the `Novell\ndk\cldapsdk\bin\win32\nls\english` directory.
- On a Unix platform (Solaris, Linux, or HP-UX) in the `[install directory]/cldapsdk/lib/locale/C/LC_MESSAGES` directory.

NetWare NLMs

If you wish to translate the messages to another language for the NetWare platform, you will need to use the internationalization tools included in the [NLM User Interface Developer Components \(http://developer.novell.com/ndk/unsupported.htm#nwsnut\)](http://developer.novell.com/ndk/unsupported.htm#nwsnut). Use the `ldapmsg.h` file and the tools to create an `errormsg.mdb` file. Use the tools to translate the `errormsg.mdb` file. Use the translated file and the tools to create an `ldapsdk.msg` file.

Windows

If you wish to translate the messages to another language for the Windows platform, translate the `errmsg.rc` file. When you save the file, the `resource.h` file is created. Build the code to convert the `errmsg.rc` and `resource.h` files to an `errmsg.dll` file.

Unix

If you wish to translate the messages to another language for a Unix platform, complete the following steps:

- 1 Translate the messages in the `ldapsdk.po` file to the target language. This file is located in the `<install_directory>/cldapsdk/lib/locale/C/LC_MESSAGES` directory. The following steps assume that French is the target language.
- 2 Use the `msgfmt` command to convert the `ldapsdk.po` file to an `ldapsdk.mo` file.
- 3 Create the directory for the messages. For French, the directory would be the following:
`<install_directory>/cldapsdk/lib/locale/fr/LC_MESSAGES`
- 4 Copy the `ldapsdk.mo` file to the directory created in Step 3.
- 5 Export the following:
`NLDAPSDK_ROOT=<install_directory>`

1.3 Authentication and Security

This section contains an overview of authentication and security in the LDAP Libraries for C. This section provides the information you need to set up SSL security, effectively authenticate servers and clients, examine certificates, and securely transport information across your network.

- “[Setting Up SSL Security](#)” on page 26 contains instructions on configuring eDirectory for use with SSL, as well as information on exporting server and client SSL certificates.
- “[Authentication](#)” on page 27 contains an overview of the different authentication mechanisms available in the LDAP Libraries for C.
- “[SSL Certificates](#)” on page 31 contains an explanation of the different methods available to effectively examine then accept or reject SSL Certificates.
- “[Transport Layer Security](#)” on page 33 contains instructions on starting and stopping Transport Layer Security (TLS).

1.3.1 Setting Up SSL Security

The LDAP Libraries for C are independent of Novell client software, and they perform their own authentication. For SSL authentication to work, the LDAP server must have a certificate to use with SSL, and the LDAP libraries must be configured to trust the LDAP server's certificate. Thus, the following two components must be set up to use SSL:

- “[LDAP Server](#)” on page 27
- “[Server Certificate](#)” on page 27

Additionally, to use Client-Based Certificate Authentication (CBCA, sometimes referred to as mutual authentication), you must have a client certificate. See the following for additional information:

- “Client Certificate” on page 27

LDAP Server

In eDirectory 8 and higher, the LDAP server is installed and started automatically with eDirectory. The LDAP server is set up to service anonymous binds by default.

To enable secure connections over SSL, the LDAP server must be set up with a digital certificate from a Certificate Authority.

The steps for setting up SSL on the LDAP server are slightly different for each release of eDirectory. For specific information, see one of the following:

- eDirectory Documentation (<http://www.novell.com/documentation/lg/edir87/edir87/data/a2iii88.html>)
- NetWare 5 for PKI (<http://www.novell.com/documentation/lg/nw5/ussecur/crndsenu/data/h0000014.html>) and for the LDAP server (http://www.novell.com/documentation/lg/nw5/usnds/ldap_enu/data/h0000012.html)

Server Certificate

The LDAP libraries perform SSL server authentication using SSL certificates.

To export an eDirectory server certificate use ConsoleOne. For step-by-step instructions for this procedure, see the [Novell Certificate Server Documentation](http://www.novell.com/documentation/lg/crt221ad/index.html). (<http://www.novell.com/documentation/lg/crt221ad/index.html>)

For details on using this certificate see “SSL Certificates” on page 31.

Client Certificate

The LDAP Libraries for C perform SSL client authentication using SSL certificates.

To export a client certificate, use ConsoleOne. For step-by-step instructions for this procedure see [Novell Certificate Server Documentation](http://www.novell.com/documentation/lg/crt221ad/index.html) (<http://www.novell.com/documentation/lg/crt221ad/index.html>).

TIP: When exporting a client certificate using ConsoleOne, you can place the client private key and certificate in the same file, then secure this file with a password. This password helps prevent unauthorized use of this file.

For details on using this certificate see the Client-Based Certificate Authentication and the SASL External sections in “Authentication” on page 27.

1.3.2 Authentication

The LDAP Libraries for C provide two methods for authentication: Simple Bind and Simple Authentication Security Layer (SASL). Simple Bind enables you to authenticate using a

distinguished name and password, whereas SASL defines a standard method to support any number of different authentication mechanisms.

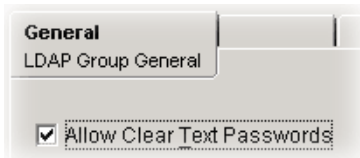
- “Simple Bind” on page 28
- “Simple Authentication Security Layer” on page 29

Simple Bind

Simple bind enables you to authenticate to an LDAP server using a distinguished name and password. Simple bind can be used with or without SSL security, and with or without client-based certificate authentication (CBCA).

Simple Bind Without SSL

In order to use simple bind without SSL, eDirectory must be configured to accept clear text passwords:



WARNING: Enabling eDirectory to accept clear text passwords means that any password you send in clear text is not encrypted as it is transported. Clear text passwords should not be used outside of a secure environment.

For an example, see [bind.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/bind.c.html) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/bind.c.html). For information on client certificates, see “Client Certificate” on page 27.

Simple Bind With SSL

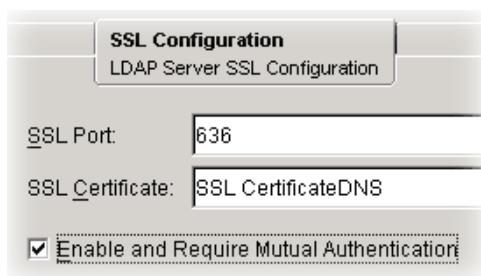
To use simple bind with SSL, the session is initialized using `ldapssl_init` (instead of `ldap_init`), which returns an SSL-enabled context handle to your application. calling `ldap_simple_bind` with this handle encrypts your bind call using SSL.

For an example using simple bind with SSL, see [sslbind.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/sslbind.c.html) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/sslbind.c.html).

Client-Based Certificate Authentication

Optionally, the LDAP client can present an SSL certificate during authentication, and eDirectory can be configured to require this. This feature is called client-based certificate authentication

(CBCA, sometimes referred to as mutual authentication), and can be enabled on the LDAP server object using ConsoleOne:



To use CBCA, specify a client private key and a client certificate by calling the `ldapssl_set_client_private_key` and the `ldapssl_set_client_cert` functions. Once you have specified a private key and certificate, call `ldap_simple_bind` to perform the bind.

For an example using client based certificate authentication with simple bind, see [mutual.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/mutual.c.html) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/mutual.c.html).

Simple Authentication Security Layer

Simple Authentication Security Layer (SASL) is a standard way for adding authentication support to connection-based protocols.

SASL is used by LDAP to provide modular authentication by defining a standard method for a client and server to use common or custom mechanisms for authentication. Several SASL mechanisms are currently defined by IETF RFCs and Internet drafts.

Although generic SASL support is provided by the `ldap_sasl_bind` function, the LDAP Libraries for C have been enhanced to simplify using many SASL mechanisms.

NOTE: Support for SASL was added in eDirectory 8.7. To determine which SASL mechanisms are supported by any LDAP server query the rootDSE.

SASL is defined in RFC 2222. The following SASL mechanisms are currently supported by eDirectory 8.7:

Digest MD5

Digest MD5 uses a hash algorithm to encrypt and ensure the integrity of transferred data without using SSL. During Digest MD5 authentication, the client sends a request to the server, to which the server responds with a digest-challenge (unique data that is verified by the client). The client then sends a response to the server with digest information and login credentials. If the server successfully verifies the response the user is authenticated.

Digest MD5 is defined in RFC 2831.

To use Digest MD5, call the `ldap_bind_digest_md5_start` function. Once that call completes successfully, call the `ldap_bind_md5_finish` function specifying your login credentials.

Optionally, before calling the `ldap_bind_md5_finish` function, you can call `ldap_get_digest_md5_realms` to retrieve the MD5 realms. In eDirectory, there is only one realm returned which is the tree to which you sent the bind request.

For an example, see [md5bind.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/md5bind.c.html\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/md5bind.c.html).

NOTE: To use SASL Digest MD5 you do not need to call the `ldap_sasl_bind` function directly.

Mechanism-specific dependencies:

- ❑ The `simplePassword` attribute must have been set for the user attempting authentication. To set a `simplePassword`, use the Novell Import Convert Export utility or the `simplePassword` snap-in for ConsoleOne.

External

SASL External is used in conjunction with the client-based certificate authentication (CBCA) feature of eDirectory. This enables you to require any client attempting a connection to your sever to present an SSL certificate for verification. With this mechanism, the client and server exchange SSL certificates and each determine whether or not to accept the connection.

When using CBCA, instead of passing credentials, the client can use the SASL External mechanism to authenticate to the server based on the information in the SSL certificate.

To use SASL external, specify a client private key and a client certificate by calling the `ldapsl_set_client_private_key` and the `ldapsl_set_client_cert` functions. Once you have specified a private key and certificate, call `ldap_sasl_bind` specifying EXTERNAL as the mechanism parameter to perform the bind.

For an example, see [saslExternal.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/saslExternal.c.html\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/saslExternal.c.html).

Mechanism-specific dependencies:

- ❑ Client-based certificate authentication must be enabled and required by eDirectory (see “LDAP Server” on page 27).

Novell Modular Authentication Services

Novell Modular Authentication Services (NMAS) provides the ability for developers to add various login and authentication methods to their applications. Possible methods include face recognition, fingerprints, voice recognition, signature, iris recognition, tokens, and smart cards as well as standard passwords. The NMAS SASL mechanism enables you to use these methods to bind to eDirectory using the LDAP protocol.

To use the SASL NMAS mechanism, call `ldap_bind_nmas_s` specifying the requested NMAS sequence and clearance.

NOTE: To use SASL NMAS authentication you do not need to call the `ldap_sasl_bind` function directly.

Mechanism-specific dependencies:

- ❑ Microsoft Windows (NMAS functionality is limited to Windows)
- ❑ The NMAS library, `nmas.dll`, which is included with the LDAP Libraries for C. For additional information on NMAS see *Novell Modular Authentication Service* (<http://>

developer.novell.com/ndk/doc/nmas/index.html?page=/ndk/doc/nmas/nmas_enu/data/a3012t8.html).

GSSAPI

The SASL-GSSAPI mechanism enables you to authenticate to eDirectory through LDAP using a Kerberos ticket and you are not required to enter the eDirectory user password. The Kerberos ticket should be obtained by authenticating to a Kerberos server.

This feature is primarily useful for LDAP application users in environments that already has a Kerberos infrastructure in place. Therefore, these users should be able to authenticate to the LDAP server without providing a separate LDAP user password.

To facilitate this, eDirectory introduces the SASL-GSSAPI mechanism.

The current implementation of SASL-GSSAPI is compliant with RFC 2222 (<http://www.ietf.org/rfc/rfc2222.txt?number=2222> (<http://www.ietf.org/rfc/rfc2222.txt?number=2222>)) and supports only Kerberos v5 as the authentication mechanism.

Mechanism-specific dependencies:

- ☐ We assume that SASL-GSSAPI is already installed on your machine. If not, you might want to download and install SASL-GSSAPI.
- ☐ On Windows, SSPI is used for Kerberos authentication

1.3.3 SSL Certificates

The LDAP Libraries for C can be configured to handle server SSL certificates in one of three ways:

- **Add trusted certificates.** Your application individually adds each trusted server certificate and does not accept any other certificates. This is the most secure way to handle SSL certificates and is the default mode.
- **Interactive verification.** Your application provides a callback mechanism that is called when non-trusted certificates are encountered. This method provides functions to determine the characteristics of the certificate so your application can decide whether or not to trust the certificate.

Add Trusted Certificates

If your application is designed to work with a known set of LDAP servers, the most secure way to handle SSL certificates is to individually add each server certificate.

Only trusted certificates are accepted in this mode unless you specify a callback function using interactive verification, in which case your callback function is called to handle the certificate. This is the default mode for SSL certificate verification.

To add trusted certificates, use the `ldapsl_add_trusted_cert` (page 299) function to add each certificate from a DER or Base64 encoded file. For instruction on exporting encoded certificates using ConsoleOne see “[Setting Up SSL Security](#)” on page 26.

Interactive Server Verification

The LDAP libraries for C provide an interactive mechanism to handle SSL certificates, called Interactive SSL.

Interactive SSL is used in conjunction with the add trusted certificates mode to provide a callback function when an un-trusted certificate is encountered. If a certificate is not found in the list of trusted certificates, your callback function is called to review the certificate. Your callback function can then choose to accept or reject the certificate.

Interactive server verification mode is set by calling the `ldap_ssl_set_verify_callback` function and specifying a callback function. If no callback function is specified, certificates are handled as described in “[Add Trusted Certificates](#)” on page 31.

For an example of a complete certificate callback routine, see `sslbindi.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/sslbindi.c.html).

Creating a Certificate Callback Function

To create your own certificate callback function you need to do the following three things:

- 1 Before coding, determine your criteria for accepting or rejecting certificates based on the certificate status, issuer, subject, and validity period.
- 2 In your code, retrieve the certificate status and other certificate information and determine if the certificate meets your acceptance criteria.
- 3 Return either `LDAPSSL_CERT_ACCEPT` (to accept the certificate) or `LDAPSSL_CERT_REJECT` (to reject the certificate).

Certificate Status

The SSL certificate status codes are defined in [Section 6.12, “SSL Certificate Status Codes,” on page 424](#). The status code indicates the reason your callback function was called. For example, the certificate might be un-trusted, contain an invalid date, or a formatting error.

The certificate status is retrieved by calling `ldapssl_get_cert_attribute` (page 303) specifying `LDAPSSL_CERT_GET_STATUS` as the attribute ID you would like returned.

Of the sixteen status codes, only three indicate a valid certificate:

`LDAPSSL_CERT_STATUS_ERR_CERT_UNTRUSTED`, `LDAPSSL_CERT_STATUS_ERR_DEPTH_ZERO_SELF_SIGNED_CERT`, and `LDAPSSL_CERT_STATUS_ERR_SELF_SIGNED_CERT_IN_CHAIN`. The first status simply means that this certificate is not trusted, and the other two indicate a self-signed certificate.

All other status codes indicate a problem with the certificate, such as an invalid date or a format error. In most cases you will reject invalid certificates, though you can find out more about the certificate and decide based on other factors.

Other Certificate Information

The `ldapssl_get_cert_attribute` function can also retrieve the certificate issuer (`LDAPSSL_CERT_ATTR_ISSUER`), the certificate subject (`LDAPSSL_CERT_ATTR_SUBJECT`), and the certificate validity period (`LDAPSSL_CERT_ATTR_VALIDITY_PERIOD`) to help you determine whether or not to accept the certificate. For example, you might want to check the issuer and validity period on all un-trusted certificates before accepting them.

Accept or Reject the Certificate

Once you have determined whether or not the certificate meets your criteria for acceptance your callback function returns either `LDAPSSL_CERT_ACCEPT` to accept the certificate or `LDAPSSL_CERT_REJECT` to reject the certificate.

Helper Functions

In addition to `ldapssl_get_cert_attribute`, the LDAP Libraries for C provide other functions to help you handle SSL certificates, outlined below:

- `ldapssl_get_cert` (page 301) enables you to place the certificate in a buffer encoded in DER or Base64 format. You can then pass this buffer directly to `ldapssl_add_trusted_cert` (page 299) to add this certificate to the list of trusted certificates.
- `ldapssl_add_trusted_cert` (page 299) enables you to add a certificate to a list of trusted certificates. The certificate will remain trusted for the duration of the session.
- `ldapssl_get_verify_mode` (page 310) returns the current server verification mode.

Accept Any Certificate

This function will be deprecated in the future C LDAP SDK releases. For this release, the `LDAPSSL_VERIFY_NONE` option will not be supported in both `ldapssl_set_verify_mode` and `ldapssl_get_verify_mode`.

1.3.4 Transport Layer Security

When you perform SSL authentication, SSL security is used to encrypt data transfers for the duration of the session.

TIP: Transport Layer Security (TLS) is the open-standards equivalent of SSL. When the IETF standardized SSL, this standardized security layer was named TLS.

Because of the overhead of encryption, there are times when a client might want to disable SSL security and send information un-encrypted. Additionally, you might want to perform a clear text or non-SSL SASL authentication, then enable SSL security to transfer a piece of sensitive information.

The LDAP Libraries for C provide `startTLS` and `stopTLS` functions to enable and disable TLS.

For an example of starting and stopping TLS see [starttls.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/starttls.c.html) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/starttls.c.html).

1.3.5 Recommendations

We recommend you do the following for maximizing the security:

- Do a simple bind over encrypted channel.
- Do not accept any certificates without validation.
- Do a check for the SSL authentication failure. As LDAP Libraries for C does not check this, but your application should do it.
- Do set the cipher to high.

1.4 LDAP Searches

In LDAP, search functions are used for both searching and reading information from the directory. The LDAP Libraries for C provide the following asynchronous and synchronous functions:

Function	Description
<code>ldap_search</code> <code>ldap_search_s</code>	Uses a search filter and specified attributes to search the directory either synchronously or asynchronously.
<code>ldap_search_ext</code> <code>ldap_search_ext_s</code>	Use a search filter, specified attributes, time limit, and LDAP controls to search the directory either synchronously or asynchronously.
<code>ldap_search_st</code>	Uses a search filter, specified attributes, and a time limit to synchronously search the directory.

A search results can contain entry, references, or search result messages. The last message in the results is always a search result message. Each message type has its own set of functions for reading the results. If you use the `ldap_first_message` and `ldap_next_message` functions, the following message types are returned:

- **LDAP_RES_SEARCH_ENTRY**
Returns the directory entries from the search.
- **LDAP_RES_SEARCH_REFERENCE**
Returns a sequence of one or more LDAP URLs. An **LDAP_RES_SEARCH_REFERENCE** is returned for each area not explored by this LDAP server during the search.
If the LDAP server is configured to follow referrals automatically, the LDAP server will never return **LDAP_RES_SEARCH_REFERENCE** to the application.
- **LDAP_RES_SEARCH_RESULT**
Returns a search result message.

If you are only interested in the entry messages from the search, you can use the `ldap_first_entry` and `ldap_next_entry` to read just the entry results and to skip any other type of message.

If you are only interested in search references returned from the search, you can use the `ldap_first_reference` and `ldap_next_reference` functions to read the references and to skip any other type of message.

1.4.1 Setting the Search Parameters and Search Constraints

The search parameters set up the criteria for where the search begins, what entries to find, and what information to return with the matching entries. Search constraints determine how many entries to return and set time limits on looking for the entries. These same parameters and constraints can be set up to perform read operations.

Base. The base parameter specifies the container in the directory where the search begins. You can specify the root of the directory tree, or any container or branch of the directory tree. For quick searches, be as specific as you can because a branch search is always faster than a full search from the tree root.

Scope. The scope parameter specifies how deep to search. It allows three levels to be set:

- LDAP_SCOPE_BASE (0x00)—searches only the entry specified by the base parameter. If the base parameter is set to the entry and the scope parameter set to this level, the search becomes a read of this entry.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with, and including, the entry specified by the base parameter. If the eDirectory server does not contain replicas for all the containers in the specified subtree, the server can automatically follow referrals to other servers. A session option, LDAP_OPT_REFERRALS, allows you to specify whether referrals are followed automatically or whether search references should be returned to where the additional entries are located. For more information, see [Section 6.10, “Session Preference Options,” on page 413](#)
- LDAP_SCOPE_SUBORDINATESUBTREE (0x03)—searches all subordinates of the specified base object, but does not include the base object, as the subtree scope does.

Filter. The search filter specifies what you are searching for. The following is a simple filter that searches for all entries with the last name of Smith.

```
"sn=Smith"
```

The default filter, if no filter is specified, is "(objectClass=*)". This filter searches every entry in the directory since the objectClass attribute is a required attribute of all entries in the directory.

These simple filters are strings with the following format:

```
attribute_name operator value
```

For example, if you specified (cn=Kim Smith), the search would return entries with a common name of Kim Smith.

For information about the grammar required to create more complex search filters, see [“Using Search Filters” on page 37](#).

Attributes. The attribute parameter specifies which attributes to return with each matching entry. The parameter accepts the following types of values:

- To return specific attributes, you pass a NULL-terminated array of attribute names in the parameter.
- To return only entry names (and no attributes), set the first, and only, string in the array to LDAP_NO_ATTRS.
- To return all attributes, set this parameter to NULL

Attributes Only. This parameter determines whether values are returned with the attributes specified in the attribute parameter. Set this parameter to zero (0) to return attributes and values. Set it to a non-zero value to return only attribute names and no values.

Time Limit. The time limit determines how long the server should wait for search results before returning to the client. The time limit is approximate because the client passes the value to the LDAP server with the search request and is dependent upon the LDAP server's interpretation of the limit.

The `ldap_search_ext`, `ldap_search_ext_s`, and `ldap_search_st` functions allow you to specify the time limit with a timeout parameter. This parameter points to a `timeval` structure that specifies the maximum time to wait for the results of a search to complete. The structure specifies both the time the server waits for the operation to complete as well as the time the local function waits for the server to respond. If the timeout parameter is set to `NULL`, the client timeout is infinite and the server uses the timeout value specified in the `LDAP_OPT_TIMELIMIT` option.

The other search functions do not have a timeout parameter and use the `LDAP_OPT_TIMELIMIT` option. This option determines the number of seconds an LDAP server will spend on a search. A value of `LDAP_NO_LIMIT` (0) means no limit. The default is `LDAP_NO_LIMIT`.

To get the option's current value, use [ldap_get_option \(page 166\)](#).

To set the option's value, use [ldap_set_option \(page 266\)](#).

Search Result Limits. This parameter or constraint determines how many entries are returned in a search results. Two functions, `ldap_search_ext` and `ldap_search_ext_s`, have a `sizelimit` parameter. To specify no limit, set this parameter to `LDAP_NO_LIMIT` (0). To use the value in the `LDAP_OPT_SIZELIMIT` option, set this parameter to -1.

The other search functions use the `LDAP_OPT_SIZELIMIT` option to determine how many entries are returned from a search. A value of `LDAP_NO_LIMIT` (0) means no limit. The default is `LDAP_NO_LIMIT`.

To get the current value, use [ldap_get_option \(page 166\)](#).

To set the value, use [ldap_set_option \(page 266\)](#).

Alias Dereferencing. The `LDAP_OPT_DEREF` option determines how aliases are handled during a search. It supports the following values:

- `LDAP_DEREF_NEVER` (0x00)
- `LDAP_DEREF_SEARCHING` (0x01)
- `LDAP_DEREF_FINDING` (0x02)
- `LDAP_DEREF_ALWAYS` (0x03)

The default is `LDAP_DEREF_NEVER`.

The `LDAP_DEREF_SEARCHING` flag indicates that aliases are dereferenced during the search but not when locating the base object of the search.

The `LDAP_DEREF_FINDING` flag indicates that aliases are dereferenced when locating the base object but not during the search.

The `LDAP_DEREF_ALWAYS` flag indicates that aliases are dereferenced when locating the base object and when finding entries.

The `LDAP_DEREF_NEVER` flag indicates that aliases are not dereferenced.

To get the current value, use [ldap_get_option \(page 166\)](#).

To set the value, use [ldap_set_option \(page 266\)](#).

1.4.2 Using Search Filters

The LDAP search filter grammar is specified in RFC 2254 and 2251. The grammar uses ABNF notation.

```
filter = " ( " filtercomp " ) "  
filtercomp = and / or /not /item  
  
and = "&" filterlist  
    filterlist = 1*filter  
  
or = "|" filterlist  
    filterlist = 1*filter  
  
not = "!" filterlist  
    filterlist = 1*filter  
  
item = simple/present/substring/extensible  
  
simple = attr filtertype value  
    attr = name | name;binary  
    filtertype = equal/approx/greater/less  
    value = data valid for the attribute's syntax  
  
equal = "="  
approx = "~="   
greater = ">="   
less = "<="   
  
present = attr "=*"   
    attr = name | name;binary  
  
substing = attr "=" [initial] any [final]   
    attr = name | name;binary  
    initial = value  
    any = "*" *(value "*")  
    final = value  
  
extensible = attr [":dn"] [": matchingrule] ":value  
    / [":dn] [": matchingrule] "value  
    /matchingrule = name | OID
```

For additional options for the attr option, see Section 4.1.5 of RFC 2251.

For additional information on the value option, see Section 4.1.6 of RFC 2251.

IMPORTANT: eDirectory does not support LDAP approximate (~=) matching or extensible matching rules.

Operators

Table 1-7 LDAP Filter Operators

Operator	Description
=	<p>Used for presence and equality matching. To test if an attribute exists in the directory, use (attributename=*). All entries that have the specified attribute will be returned. To test for equality, use (attributename=value). All entries that have attributename=value are returned.</p> <p>For example, (cn=Kim Smith) would return entries with Kim Smith as the common name attribute. (cn=*) would return all entries that contained a cn attribute. The = operator can also be used with wildcards to find a substring, (cn=*ary*) would return mary, hillary, and gary.</p>
>=	<p>Used to return attributes that are greater than or equal to the specified value. For this to work, the syntax type of the attribute must have defined a mechanism to make this comparison.</p> <p>For example, (cn>=Kim Smith) would return all entries from Kim Smith to Z.</p>
<=	<p>Used to return attributes that are less than or equal to the specified value. For this to work, the syntax type of the attribute must have defined a mechanism to make this comparison.</p> <p>For example, (cn<=Kim Smith) would return all entries from A to Kim Smith.</p>
~=	<p>Used for approximate matching. The algorithm used for approximate matching varies with different LDAP implementations.</p>

The following boolean operators can be combined with the standard operators to form more complex filters. Note that boolean operator syntax is used different in search filters than in the C and Java programming languages, but the concepts are the same.

Table 1-8 LDAP Filter Boolean Operators

Boolean Operators	Description
&	<p>And. For example, (&(cn=Kim Smith)(telephonenumber=555-5555)) would return entries with common name of Kim Smith and a telephone number of 555-5555.</p>
	<p>Or. For example, ((cn=Kim Smith)(cn=Kimberly Smith)) would return entries with common name Kim Smith or Kimberly Smith.</p>
!	<p>Not. For example, (!(cn=Kim Smith)) would return entries with any cn other than Kim Smith. Note that the ! operator is unary.</p>

Examples:

Table 1-9 *Examples for Different Filters*

Filter and Description
<code>(cn = Kim Smith)</code> Returns entries with a common name of Kim Smith.
<code>(&(cn=Kim Smith)(telephonenumber=555*)(emailaddress=*acme.com))</code> Returns entries with a common name of Kim Smith, a telephone number that starts with 555, and an e-mail address that ends in acme.com
<code>!(cn = Chris Jones)</code> Returns entries that do not have a common name of Chris Jones.
<code>(&(objectClass=inetOrgPerson) ((sn=Smith) (cn=Chris S*)))</code> Returns entries that are of type inetOrgPerson with a surname of Smith or a common name beginning with Chris S.
<code>(&(o=acme)(objectclass=Country)!((c=spain)(c=us))</code> Returns entries that are of type Country from the organization Acme, that are not countries spain or us.

1.4.3 Operational Attributes

Operational attributes are not automatically returned in search results; they must be requested by name in the search operation. For a list of supported operational attributes in eDirectory 8.5, see “**LDAP Operational Attributes**” in *LDAP and eDirectory*. The LDAP servers in previous releases of eDirectory do not support operational attributes.

1.5 LDAP Based Backup

LDAP based backup is used to backup the attributes and attribute values for one object at a time.

This feature enables you to make an incremental backup wherein the object is backed up only if there are changes to the object.

LDAP based backup provides a set of interfaces for backup and restore of eDirectory objects exposed through the LDAP libraries for C through LDAP extended operations. See [ldap_backup_object \(page 320\)](#) and [ldap_restore_object \(page 374\)](#).

The LDAP based backup tries to resolve the problems with the current backup and restore. The problems that this feature resolves are:

- Gives a consistent interface using which any third party backup applications or developers can backup eDirectory on all the supported platforms.
- Provides a backup solution to backup objects incrementally.

1.6 Referral Handling in LDAP v3

Because of the distributed nature of directory services, operations sent to an LDAP server often result in a referral to another LDAP server that might contain the requested data or entries.

When an LDAP server does not contain the requested data and a referral is necessary, eDirectory and your application can be configured to handle them in one of four ways.:

- Configure eDirectory to return complete data and never referrals to the client (always chain).
- Send referrals to the client only for eDirectory servers that do not support chaining.
- Always send referrals to the client (never chain).
- If the second or third option is selected and your application will receive referrals from eDirectory, your application can be configured to have the API automatically follow referrals (anonymous by default or authenticated using a rebind process), or your application can perform its own manual referral handling.

1.6.1 Configuring eDirectory to Return Complete Data

In eDirectory, the LDAP server can be configured to return complete data and not return referrals. This is done through the LDAP Group Object using ConsoleOne. For possible configurations in eDirectory, see the documentation for the [LDAP Group Object \(http://www.novell.com/documentation/lg/edir87/edir87/data/agy2a0m.html\)](http://www.novell.com/documentation/lg/edir87/edir87/data/agy2a0m.html).

1.6.2 Configuring eDirectory to Return Referrals

The LDAP server in eDirectory can also be configured to return referrals to your application. This is done through the LDAP Group Object using ConsoleOne. For possible configurations in Novell eDirectory, see the documentation for the [LDAP Group Object \(http://www.novell.com/documentation/lg/edir87/edir87/data/agy2a0m.html\)](http://www.novell.com/documentation/lg/edir87/edir87/data/agy2a0m.html)

1.6.3 Enabling Referral Handling in the Application

The LDAP Libraries for C are initially set up to automatically follow referrals. This feature is controlled through the LDAP_OPT_REFERRALS option in the ld session handle.

- When set to ON (the default value), the libraries follow the referrals and perform an anonymous bind to the referred servers. In eDirectory, this bind is equivalent to the [Public] user and grants minimal rights to entries in the directory.

If you want your application to follow referrals but to perform a stronger authentication than an anonymous bind, you must supply a rebind process (see [“Creating a Rebind Process” on page 41](#)).

- When set to OFF, the libraries return LDAP_REFERRAL status (10) on LDAP operations and continuation references on search operations as part of the search results. When you receive LDAP_REFERRAL status the referrals can be retrieved using `ldap_get_option` and specifying LDAP_OPT_REFERRAL_LIST as the requested value. This returns a NULL-terminated list of string pointers containing the referrals.

TIP: To change the setting of the LDAP_OPT_REFERRALS option, call the `ldap_set_option` function with the option parameter set to LDAP_OPT_REFERRALS (see [ldap_set_option \(page 266\)](#)).

1.6.4 Creating a Rebind Process

The `rebind` function must use the following syntax.

```
int LIBCALL rebind_function (
    LDAP      *ld,
    const char *url,
    int        request,
    ber_int_t   msgid)
{
    /* the body must perform a synchronous bind */
}
```

The `ld` parameter must be used by the application to bind 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 \(page 282\)](#) function to parse the string into its components.

The `request` parameter specifies the request operation that generated the referral. For possible values, see [Section 6.8, “Request Message Types,” on page 412](#).

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

The 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 is responsible for obtaining the required authentication information (user name, password, and certificates) associated with the `ld` and passing this information to the `rebind` function. The `rebind` function is responsible for performing the synchronous bind.

You must design your application to handle the possibility that the `rebind` process cannot bind to any of the referrals (for example, the servers are down or the authentication information is invalid). When this happens, the LDAP libraries return either

- results with referrals
- search results with search references

1.6.5 Using the Rebind Process

For the libraries to use a `rebind` process, the application must configure the `ld` to the following values:

- LDAP_OPT_REFERRALS option must be set to ON (the default value). For configuration information, see [ldap_set_option \(page 266\)](#).
- LDAP_REBIND_PROC must be set to the `rebind` function (see [ldap_set_rebind_proc \(page 268\)](#)).

When the ld has the proper settings, the referrals are processed according to the following algorithm.

1. The LDAP server sends a referral back to the libraries.
2. The libraries call the rebind function, setting the ld and url parameters.
3. The application supplies the logic for determining the type of bind.
For example, if the referral is to a server outside of a firewall, the application could decide to do an anonymous bind rather than a secure bind.
4. The application supplies the bind credentials associated with the ld (user name, password, and certificates) and with the bind method (such as simple, SSL, or SASL)
5. The libraries process the rebind function. If successful, the rebind function returns LDAP_SUCCESS.

If any other value is returned, the referral process stops and either LDAP_REFERRAL is returned as a result code for the original LDAP operation, or if a search operation, a search continuation is returned with the search results.

1.6.6 Following Referrals Manually

When eDirectory is configured to return referrals and automatic referral handling is turned off in your application, the libraries return LDAP_REFERRAL status (10) on LDAP operations and continuation references on search operations as part of the search results. When you receive LDAP_REFERRAL status, referrals can be retrieved using ldap_get_option and specifying **LDAP_OPT_REFERRALS** as the requested value. This returns a NULL-terminated list of string pointers containing the referrals.

If a referral is returned with no DN field, the library inserts the DN of the original request in the referral before returning it.

Your application can then determine how to handle each returned referral.

1.6.7 Retrieving Referrals for Non-Search Operations

eDirectory 8.7 can return referrals for non-search operations. See **“Enabling Referral Handling in the Application” on page 40** for details on handling these referrals.

1.6.8 Limiting Referral Hops

Your application can specify the maximum number of referral hops the LDAP libraries will follow. For example, suppose you set the limit to two. Your application does a search, and the search refers you to the following:

Server 1 refers you to Server 2—Hop 1
Server 2 refers you to Server 3—Hop 2
Server 3 refers you to Server 4—Hop 3

The libraries will follow the referral through Server 3, but they will not continue to Server 4 because Server 4 exceeds the hop limit of 2. They return an result code of LDAP_REFERRAL_LIMIT_EXCEEDED.

To set the referral hop limit, call the ldap_set_option function with the option parameter set to LDAP_OPT_REFERRAL_HOP_LIMIT (see **ldap_set_option (page 266)**).

1.7 eDirectory Event System

The eDirectory Event System provides a way for applications to monitor the activity of eDirectory on an individual server over LDAP. LDAP Event Services are available on eDirectory 8.7.

For additional information on the eDirectory Event System and for a complete listing of LDAP events, see “[LDAP Event Services](#)” in the “LDAP and eDirectory Integration Guide.”

1.7.1 Registering to Monitor an Event

The LDAP Libraries for C provide functions to simplify registering to receive event data. To register to monitor an event, you call the [ldap_monitor_events](#) (page 346) or [ldap_monitor_events_filtered](#) (page 348) function passing an array of [EVT_EventSpecifier](#) (page 444) or [EVT_FilteredEventSpecifier](#) (page 445) structures specifying the events you wish to monitor, and an event filter if calling [ldap_monitor_events_filtered](#).

These functions send a MonitorEventRequest or FilteredMonitorEventRequest extended operation to the server. The request is sent asynchronously; it does not wait for a response from the server. The functions return the constant LDAP_SUCCESS if the request was successfully sent, or another LDAP result code if not.

After a successful call to [ldap_monitor_events](#) or [ldap_monitor_events_filtered](#), server responses to the EventMonitorRequest are retrieved by calling the [ldap_result](#) (page 231) function. If the return value of [ldap_result](#) is equal to LDAP_RES_EXTENDED, it indicates that an error or exceptional situation occurred and events are not monitored. The result is parsed by calling the [ldap_parse_monitor_events_response](#) (page 356). If the return value of [ldap_result](#) is equal to LDAP_RES_INTERMEDIATE it indicates that an event has occurred. The result should be parsed by calling [ldap_parse_ds_event](#) (page 350).

Memory allocated by the [ldap_parse_monitor_events_response](#) and the [ldap_parse_ds_event](#) functions must be freed by the application by calling the [ldap_event_free](#).

For an example, see [monitorevents.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/extensions/monitorevents.c.html)

Filtered Event Monitoring

Filtered event monitoring enables you to limit the events returned to your application by the server, possibly reducing network traffic and processing in your application.

See [ldap_monitor_events_filtered](#) (page 348) for additional information.

1.7.2 LBURP

The LDAP Bulk Update/Replication Protocol (LBURP) is used to send asynchronous requests to an LDAP server. This guarantees that the requests are processed in the order specified by the protocol and not in an arbitrary order influenced by multiprocessor interactions or the operating system's scheduler. LBURP also lets the client send several update operations in a single request and receive the response for all of those update operations in a single response. This adds to the network efficiency of the protocol.

LBURP works as follows:

1. The client to an LDAP server.
2. The server sends a bind response to the client.
3. The client sends a start LBURP extended request to the server.
4. The server sends a start LBURP extended response to the client.
5. The client sends zero or more LBURP operation extended requests to the server.
These requests can be sent asynchronously. Each request contains a sequence number identifying the order of this request relative to other requests sent by the client over the same connection. Each request also contains at least one LDAP update operation.
6. The server processes each of the LBURP operation extended requests in the order specified by the sequence number and sends an LBURP operation extended response for each request.
7. After all of the updates have been sent to the server, the client sends an end LBURP extended request to the server.
8. The server sends an end LBURP extended response to the client.

The LBURP processor in eDirectory also commits update operations to the database in groups to gain further efficiency in processing the update operations. LBURP can greatly improve the efficiency of your LDIF imports over a traditional synchronous approach.

1.8 Character Conversions

This section contains reference information on character encoding and a description of UTF-8, the encoding used by LDAPv3.

1.8.1 A Brief History of Character Encoding

In the early days of computing, 7-bit ASCII was the standard. The need for more characters drove the creation of a number of 8bit Single Byte Character Sets (SBCS). ISO-8859 for example provided the 7-bit ASCII characters and many of the accented characters required for Wester Europe.

Asian languages required much more than 256 characters. Multi-byte character sets were developed using a variable number of bytes per character, such as Shift-JIS or EUC-JP.

Other encodings appeared that were stateful. They used Shift-In/Shift-Out characters, or escape sequences to switch between encoding schemes.

In an attempt to bring order to this confusion, two separate standards organizations started work on a Universal Character Set (UCS) which would encode all the characters of all the major languages in the world. The two organizations ultimately agreed to maintain a consistent encoding, and the ISO-10646/Unicode standard became widespread. ISO-10646 officially supports a 31-bit code space (0 - 0x7FFFFFFF), while Unicode supports the 21-bit space (0 - 0x10FFFF) of over a million characters. So far no characters have been assigned beyond the 16-bit Basic Multi-Lingual Plane (BMP). While the code point value assigned to each character are well defined, there are different ways that the value may be encoded.

UCS-2 refers to the encoding where each character is a fixed 16-bit length, allowing access to the BMP.

UCS-4 or UTF-32 refers to an encoding where each character is a fixed 32-bit value, allowing direct access to the entire UCS.

UTF-16 is an encoding where a character is one or two 16-bit values, allowing access to the full Unicode code space 0 - 0x10FFFF.

1.8.2 UTF-8 Encoding

There are a few problems with using these UCS-2/4 or UTF-16 encodings.

- Since most characters used today are still from the 7-bit ASCII set, it takes almost twice as much space to use Unicode.
- It is incompatible with many current APIs.
- Byte order (big endian/little endian) is an issue.
- If data is being sent across a byte stream, and a byte is dropped, all the rest of the 16 bit Unicode characters will be out of sync and there's no way to sync up.

To address these problems, a byte-encoded form of Unicode was developed called Unicode Transformation Format 8-bit Encoding (UTF-8). This is just a simple algorithmic encoding of each 16-bit Unicode character into 1, 2, or 3 bytes. 4 bytes cover the entire Unicode 21-bit space, or 6 bytes to get the full 31-bit address space.

The greatest advantage is that the encoding for all 7-bit ASCII characters is identical in UTF-8. This solves the wasted space problem nicely, and provides a degree of compatibility with older systems. Byte order is not an issue since it's a byte stream.

The encoding of UTF-8 also allows unambiguous determination of the start of a character. By examining only the first byte, one can determine the number of bytes in the UTF-8 character sequence. Continuation bytes are easily recognizable, allowing one to detect a missing byte in a stream.

RFC2279 describes the UTF-8 encoding format in detail. Many other resources on the Web, including the Unicode Consortium website contain more information.

1.8.3 UTF-8

In the LDAP version 2 specification, strings were limited to the T.61 character set, which is basically 7-bit US-ASCII minus several characters (such as tilde, caret, and curly braces). T.61 was a severe limitation to globalization and efforts to make LDAP a world-wide standard. In LDAP version 3, strings are to be encoded in UTF-8.

Because 7-bit ASCII characters are encoded identically in UTF-8, many applications continue to use local text strings with the LDAP C APIs. This works for ASCII characters, but will fail for extended 8-bit characters such as, (e accent) or multi-byte Asian characters.

The correct approach is to make sure all local strings are encoded into UTF-8 before using them in an LDAP API. Likewise strings returned from the APIs should be converted to local text if required, such as displaying them with printf.

Novell's LDAP C SDK provides routines for converting Unicode strings into UTF-8 strings. Both single character and string versions of these routines are provided. Several string processing routines are also provided, such as UTF-8 versions of strchr and strtok, next, and prev.

1.8.4 wchar_t Type

Novell's SDK conversion routines use the `wchar_t` type. This type is 2 bytes on some machines and 4 bytes on other machines, so care must be taken if `wchar_t` strings are transported to other systems. UTF-8 is the most portable way to transfer strings between systems.

`wchar_t` strings will either be UCS-2 or UCS-4 encoded, depending on the size of `wchar_t`. The advantage to using `wchar_t` strings is that all the standard wide character string processing routines may be used, such as `wcslen`, `wscat`, etc.

In summary, LDAP C applications which make the distinction between local and UTF-8 strings, and handle each properly, will be much easier to internationalize and move into the global marketplace.

1.9 Time Formats

Generalized Time Format. Generalized time represents the values of year, month, day, hour, minutes, seconds and fractions of a second in any of three forms:

- Local time "YYYYMMDDHHMMSS.fff", where `fff` is optional and is fractions of a second
- Greenwich Mean Time (UTC) "YYYYMMDDHHMMSS.fffZ", `Z` indicates Greenwich Mean Time
- Difference between local and UTC time, "YYYYMMDDHHMMSS.fff+HHMM", the +HHMM or -HHMM represents the time differential between the local and Greenwich Mean Times.

UTC Time Format. UTC format represents the values of year (2 digit), month, day, hour, minutes and optionally seconds.

- Local time "YYMMDDHHMMSS", where seconds (SS) is optional
- Greenwich Mean Time (UTC), "YYMMDDHHMMSSZ", seconds (SS) is optional and `Z` represents Greenwich Mean Time
- Difference between local and UTC time, "YYMMDDHHMMSS+HHMM", seconds (SS) is optional and +HHMM or -HHMM represents the time differential between local and Greenwich Mean Times.

1.10 Controls and Extensions

Controls and Extensions were added to version 3 of the LDAP protocol. In version 2, there was no standard mechanism to extend the protocol, requiring developers to extend the protocol non-standard ways. In version 3, extensions and controls were defined to provide consistent expansion of the protocol.

NOTE: The eDirectory and LDAP Integration guide contains a good introduction to LDAP controls and extensions, and contains information and limitations you need to be aware of when using these controls with eDirectory. It is recommended that you read the “**LDAP Controls**” and the “**LDAP Extensions**” chapters in the eDirectory and LDAP Integration guide.

1.10.1 Controls

The following table contains a list of controls supported in the LDAP Libraries for C. For examples using these controls, see the LDAP Libraries for C [“Sample Code” on page 18](#).

Table 1-10 *Supported Controls in the LDAP Libraries for C*

OID	Description
1.2.840.113556.1.4.473	Sever-side sort control request
1.2.840.113556.1.4.474	Server-side sort control response
2.16.840.1.113730.3.4.9	Virtual list view request
2.16.840.1.113730.3.4.10	Virtual list view response
2.16.840.1.113730.3.4.3	Persistent search
2.16.840.1.113730.3.4.7	Entry change notification

- **Server Side Sort.** Returns results from a search operation in sorted order. This can be used to off-load processing from the client, or if you cannot sort the results on the client.
- **Vertical List View.** Works in conjunction with the Server Side Sort control to provide a dynamic view of a scrolling list.
- **Persistent Search & Entry Change Notification.** Performs a continuous search notifying the application of changes as they occur.

1.10.2 Extensions

The following table contains a list of extensions supported in the LDAP Libraries for C. For examples using these extensions, see the LDAP Libraries for C [“Sample Code” on page 18](#).

Table 1-11 *Supported Extensions in the LDAP Libraries for C*

OID	Name
2.16.840.1.113719.1.27.100.1	ndsToLdapResponse
2.16.840.1.113719.1.27.100.2	ndsToLdapRequest
2.16.840.1.113719.1.27.100.3	splitPartitionRequest
2.16.840.1.113719.1.27.100.4	splitPartitionResponse
2.16.840.1.113719.1.27.100.5	mergePartitionRequest
2.16.840.1.113719.1.27.100.6	mergePartitionResponse
2.16.840.1.113719.1.27.100.7	addReplicaRequest
2.16.840.1.113719.1.27.100.8	addReplicaResponse
2.16.840.1.113719.1.27.100.9	refreshLDAPServerRequest
2.16.840.1.113719.1.27.100.10	refreshLDAPServerResponse

OID	Name
2.16.840.1.113719.1.27.100.11	removeReplicaRequest
2.16.840.1.113719.1.27.100.12	removeReplicaResponse
2.16.840.1.113719.1.27.100.13	partitionEntryCountRequest
2.16.840.1.113719.1.27.100.14	partitionEntryCountResponse
2.16.840.1.113719.1.27.100.15	changeReplicaTypeRequest
2.16.840.1.113719.1.27.100.16	changeReplicaTypeResponse
2.16.840.1.113719.1.27.100.17	getReplicaInfoRequest
2.16.840.1.113719.1.27.100.18	getReplicaInfoResponse
2.16.840.1.113719.1.27.100.19	listReplicaRequest
2.16.840.1.113719.1.27.100.20	listReplicaResponse
2.16.840.1.113719.1.27.100.21	receiveAllUpdatesRequest
2.16.840.1.113719.1.27.100.22	receiveAllUpdatesResponse
2.16.840.1.113719.1.27.100.23	sendAllUpdatesRequest
2.16.840.1.113719.1.27.100.24	sendAllUpdatesResponse
2.16.840.1.113719.1.27.100.25	requestPartitionSyncRequest
2.16.840.1.113719.1.27.100.26	requestPartitionSyncResponse
2.16.840.1.113719.1.27.100.27	requestSchemaSyncRequest
2.16.840.1.113719.1.27.100.28	requestSchemaSyncResponse
2.16.840.1.113719.1.27.100.29	abortPartitionOperationRequest
2.16.840.1.113719.1.27.100.30	abortPartitionOperationResponse
2.16.840.1.113719.1.27.100.31	getBindDNRequest
2.16.840.1.113719.1.27.100.32	getBindDNResponse
2.16.840.1.113719.1.27.100.33	getEffectivePrivilegesRequest
2.16.840.1.113719.1.27.100.34	getEffectivePrivilegesResponse
2.16.840.1.113719.1.27.100.35	setReplicationFilterRequest
2.16.840.1.113719.1.27.100.36	setReplicationFilterResponse
2.16.840.1.113719.1.27.100.37	getReplicationFilterRequest
2.16.840.1.113719.1.27.100.38	getReplicationFilterResponse
2.16.840.1.113719.1.27.100.39	splitOrphanPartitionRequest
2.16.840.1.113719.1.27.100.40	splitOrphanPartitionResponse
2.16.840.1.113719.1.27.100.41	removeOrphanPartitionRequest
2.16.840.1.113719.1.27.100.42	removeOrphanPartitionResponse

1.11 Runtime Version of the Library Files

The licenses governing this SDK grant permission to redistribute the LDAP Libraries for C with your application. You should review the enclosed licenses to ensure compliance.

These files are also shipped with eDirectory and the service packs. However, the NDK updates them more frequently, so you may have a newer version than the version shipping with eDirectory. In some instances, you may have older versions. If you select to redistribute the files, make sure your installation program does not overwrite newer versions.

The following sections provide some guidelines for the following platforms:

- “Windows (NT, 95, 98, 2000, XP)” on page 49
- “NetWare” on page 50
- “UNIX (Solaris, Linux, AIX, HP-UX)” on page 51

1.11.1 Windows (NT, 95, 98, 2000, XP)

On the Windows platforms, you can copy the LDAP Libraries for C files to the same directory in which you install your program or to a directory that is part of the system's path variable. Copy the non-debug version of the following library files to that directory:

```
ldapsdk.dll  
ldapssl.dll  
ldapx.dll  
nmas.dll
```

You also need the message file. Copy the nls directory and all its subdirectories and files to the same directory you copied the library files, keeping the `ldapsdkmsg.dll` file in the same relative directory structure.

Also include the following license and copyright files:

```
copyright.hspencer  
copyright.openldap  
license.openldap  
license.openssl
```

If your application uses any of the LDAP tools, these executables also need to be copied to the same directory as the library files. The ice utility requires the following files:

```
ice.cfg  
ice.exe  
ldaphdlr.dll  
ldif.dll  
sal.dll  
zone.dll
```

1.11.2 NetWare

Two versions of the LDAP libraries are provided for NetWare: A Clib version and a LibC version. The installation process extracts the files and creates two directories; one containing the libC version, and another containing the Clib version. The following tables list these directories and their contents:

Clib

[install location]\NetWare\Clib

Table 1-12 *Clib Version of the LDAP Libraries*

Folder	Description
bin	Libraries. The Clib NLMs are: <ul style="list-style-type: none">• ldapsdk.nlm• ldapssl.nlm• ldapx.nlm
imports	Import files for linking
inc	Include files
tools	Ldap tools (add, delete, modify, search)
samples	Sample programs

LibC

[install location]\NetWare\LibC

Table 1-13 *LibC Version of the LDAP Libraries*

Folder	Description
bin	Libraries. The LibC NLMs are: <ul style="list-style-type: none">• ldapsdk.nlm• ldapssl.nlm• ldapx.nlm
imports	Import files for linking
inc	Include files
tools	Ldap tools (add, delete, modify, search)
samples	Sample programs

Copy the non-debug version of either the Clib or LibC version of the library files to the sys:\system directory with your application:

```
ldapsdk.nlm  
ldapssl.nlm  
ldapx.nlm
```

You also need to copy the nls directory and its subdirectories the `sys:\system` directory, keeping the `ldapsdk.msg` file in the same relative directory structure.

If your application uses any of the LDAP tools, these nlms also need to be copied to the `sys:\system` directory. The ice utility requires the following files:

```
delim.nlm  
dirload.nlm  
ice.cfg  
ice.nlm  
ldaphdlr.nlm  
ldif.nlm  
sal.nlm  
zone.nlm
```

1.11.3 UNIX (Solaris, Linux, AIX, HP-UX)

The library files and the application's binaries must be copied to a directory where the user has all access permissions. In the following descriptions, this directory is labelled the *<install directory>*. Copy the non-debug version of the following libraries files to the *<install directory>/cldapsdk/lib* directory:

For Solaris, Linux, AIX:

```
libldapsdk.so  
libldapssl.so  
libldapx.so
```

For HP-UX:

```
libldapsdk.sl  
libldapssl.sl  
libldapx.sl
```

If your application uses any of the LDAP tools, these files also need to be copied to the *<install directory>/cldapsdk/tools* directory. The ice utility requires the following files:

For Solaris, Linux, AIX:

```
libldaphdlr.so  
libdelim.so  
libdirload.so  
libldif.so
```

For HP-UX:

```
libldaphdlr.sl  
libdelim.sl
```

```
libdirload.sl  
libldif.sl
```

Copy your application binaries to the `<install directory>/cldapsdk/bin` directory.

Copy the locale directory and its subdirectories to the `<install directory>/cldapsdk/lib` directory, keeping the `ldapsdk.mo` file in the same relative directory structure.

Export the following:

```
For Solaris and Linux: export LD_LIBRARY_PATH=<install directory>/  
    cldapsdk/lib  
For AIX: export LIBPATH=<install directory>/cldapsdk/lib  
For HP-UX: export SHLIB_PATH=<install directory>/cldapsdk/lib
```

1.12 Internationalization

The LDAP Libraries for C are enabled for internationalization. Message files are supplied for 12 major languages. These message files contain the text strings associated with each defined LDAP error code. When an application calls `ldap_err2string`, for example, the error message is returned translated into the appropriate language. If an appropriate language file is not present on the system, English strings are returned.

1.12.1 File Locations

Table 1-14 Location Details of the Message Files

Platform	Location
NetWare	<code>SYS:system\nls\<language>\ldapsdk.msg</code>
Windows	<code>nls\<language>\ldapsdkmsg.dll</code>
Unix	<code><install directory>/cldapsdk/lib/locale/<language>/ LC_MESSAGES/ldapsdk.po</code>

1.12.2 Language Directory Names

Table 1-15 Language Directory Names

Language	NetWare	Windows	Unix
Chinese Simplified	1	chineses	zh_CN
English	4	english	en
French	6	francais	fr
German	7	deutsch	de
Italian	8	italiano	it
Japanese	9	nihongo	ja

Language	NetWare	Windows	Unix
Korean	10	korean	ko
Portugese	12	portugue	pr
Russian	13	rusски	ru
Spanish	14	espanol	es
Chinese Traditional	16	chineset	zh_TW
Polish	17	polski	pl

This chapter provides step-by-step instructions for a few of the common tasks most LDAP applications perform. See [DeveloperNet University \(http://developer.novell.com/education/codeproject.html\)](http://developer.novell.com/education/codeproject.html) for C LDAP tasks that

- Create an authenticated connection
- Create an eDirectory entry
- Read attribute values
- Read and write stream attribute values
- Search for attribute values
- Write attribute values

2.1 Establishing an SSL Connection

To establish an SSL connection, both the client and the LDAP server must be set up to use SSL. For instructions, see [Section 1.3, “Authentication and Security,” on page 26](#).

To establish the SSL connection, call the following functions.

- 1 Initialize the SSL library by calling the `ldapsl_client_init` function.
- 2 Create an LDAP session handle (`ld`) by calling the `ldapsl_init` function.
- 3 Establish an authenticated SSL connection by calling the `ldap_simple_bind_s` function with a login distinguished name and password.
- 4 When you are finished with the connection, call the `ldap_unbind` function to free the memory associated with the `ld`.
- 5 To uninitialized the SSL library and free the associated memory, call the `ldapsl_client_deinit` function.

For sample code, see [sslbind.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

2.2 Reading the Root DSE

Reading the root DSE returns information about support of the following features of the LDAP server:

- LDAP versions (2 and 3)
- LDAP controls
- Schema name

With the schema name, you can then extend the schema or read its definitions. You must establish an LDAP v3 connection to read the DSE.

To read the DSE, call the following functions.

- 1** Set the LDAP version to LDAP v3 by calling the `ldap_set_option` function with the option parameter set to `LDAP_OPT_PROTOCOL_VERSION` and the invalue parameter set to `LDAP_VERSION3`.
- 2** Initialize a session and obtain an LDAP session handle (`ld`) by calling the `ldap_init` function.
- 3** Establish an authenticated connection by calling the `ldap_simple_bind_s` function.
- 4** Read the DSE by calling the `ldap_search_ext_s` function. Set the search base to `NULL`, the search filter to `(objectclass=*)`, and the scope to `LDAP_SCOPE_BASE`.
- 5** Obtain the DSE entry from the results by calling the `ldap_first_entry` function.
- 6** Obtain the first attribute by calling the `ldap_first_attribute` function.
- 7** Obtain the other attributes by calling the `ldap_next_attribute` function.
- 8** Obtain the values for the attributes by calling the `ldap_get_values` function.
- 9** Free the attributes and values by calling the `ldap_memfree` function.
- 10** Free the memory from the search results by calling the `ldap_msgfree` function.
- 11** When you are done with the session handle, call the `ldap_unbind` function to free the `ld` and the associated memory.

For sample code, see [getdse.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

2.3 Adding an Entry

To add an entry to the directory, the client must have create permissions to the container that will be the entry's parent container. Entries can be created programmatically or from an LDIF file. (For more information on using an LDIF file, see “[Adding Entries](#)”.)

To add an entry programmatically, complete the following steps.

- 1** Create an `LDAPMod` structure for each attribute that will be added with the entry.

You need a structure for each attribute. For example, an entry with a base class of `inetOrgPerson` requires `LDAPMod` structures for the following attributes: `cn`, `sn`, and `objectClass`. If you want the entry to log in to the directory, the entry also requires a structure for the `userPassword` attribute.
- 2** In each structure, set the modification operation to `LDAP_MOD_ADD` and the type to the name of the attribute. Add a `NULL`-terminated string of values for each attribute.
- 3** Add each structure to a `NULL`-terminated array of `LDAPMod` structures.
- 4** Set the `dn` for the entry.

The containers in the entry's `dn` must already exist in the directory.
- 5** Call the `ldap_add_ext_s` function to add the entry.

For sample code, see [addentry.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

2.4 Modifying an Entry

To modify an entry, the client must have write permissions to the attributes that are being modified.

- 1 Create an LDAPMod structure for each attribute that will be modified.
- 2 Set the modification operation, type, and value in each structure.
To add a value even when it may already exist, set the operation to LDAP_MOD_REPLACE.
To add a value and report an error if it already exists, set the operation to LDAP_MOD_ADD.
To delete an existing value, set the operation to LDAP_MOD_DELETE.
- 3 Add each structure to a NULL-terminated array of LDAPMod structures.
- 4 Call `ldap_modify_ext_s` to modify the specified entry.

For sample code, see [modattrs.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

2.5 Modifying an Entry's Password

eDirectory has a number of restricts that prevent password modification. The user can have insufficient rights for the following reasons:

- The user is not a supervisor of the entry.
- The flag that allows user to change the password is false.
- The password unique flag is true and the password supplied matches a previous password.
- A minimum length for the password has been set and the password is too short.
- The user did not supply the old password value with the new value in the same operation.

Passwords in eDirectory are stored as RSA public and private key pairs. The Novell LDAP server uses the `userPassword` attribute to generate these key pairs for an LDAP client.

- NDS 8.17 or higher is required for users to change their own passwords.
- NDS 7.xx is required for an administrator to change user passwords.

If the user has sufficient rights, the process is similar to modifying any attribute of an entry. For a user to change his or her own password, complete the following steps.

- 1 Create two LDAPMod structures for the `userPassword` attribute.
- 2 In the first LDAPMod structure, set the modification operation to LDAP_MOD_DELETE, the modification type to "userPassword", and the value to the current password.
- 3 In the second LDAPMod structure, set the modification operation to LDAP_MOD_ADD, the modification type to "userPassword", and the value to the new password.
- 4 Add the structures to a NULL-terminated array of LDAPMod structures.
- 5 Call `ldap_modify_ext_s` to modify the specified entry's password.

For sample code that allows a user to change his or her password, see [modpass.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm)

For sample code that allows an administrator to set a password, see [setpass.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

The user can also change the password in one LDAP modification. To change the password in a single operation:

```
dn: cn=test,o=org
changetype: modify
delete: userpassword
userpassword: pass
-
add: userpassword
userpassword: password
```

2.6 Extending the Schema

The eDirectory schema can be extended through LDAP programmatically using LDAP functions or using an LDIF file with a utility such as the “**Novell Import Convert Export Utility**” and “**ldapmodify**”.

The following steps give a simple example how to programmatically extend the schema by creating an auxiliary class that uses existing attributes.

- 1 Create a NULL-terminated string that defines the OID, the class name, description, super class, class type, must attributes, and may attributes. RFC 2252 defines the format of the string.

It should look similar to the following definition for the TestAuxClass.

```
char    *auxClassDefVals[] = { "( 1.1.1.1.1.1111
    NAME 'TestAuxClass'
    DESC 'Useless ObjectClass for testing'
    SUP 'top'
    AUXILIARY
    MUST ( operator $ server )
    MAY ( status ) )"
    , NULL };
```

NOTE: You need to use a valid OID when extending the schema. To register and obtain a unique OID for your group of attribute and class extensions, see the [Novell Developer Support Web site \(http://developer.novell.com/support\)](http://developer.novell.com/support)

- 2 Create an LDAPMod structure for the class.
 - Set the mod_op to LDAP_MOD_ADD
 - Set the mod_type to "objectclasses"
 - Set the mod_values to the string (in the example above, to auxClassDefVals)
- 3 Add each structure to a NULL-terminated array of LDAPMod structures.
- 4 To add the class, call ldap_modify_ext_s with the parameters set to the following values:
 - dn to "cn=schema" (the name of the schema is obtained by reading the root DSE; see [Section 2.2, “Reading the Root DSE,” on page 55](#))
 - mods to the NULL-terminated array of LDAPMod structures you have created
 - serverctrls to NULL
 - clientctrls to NULL

Standard LDAP Functions

3

This chapter documents the standard LDAP functions defined by RFCs and Internet drafts maintained by IETF. For information on the LDAP extensions Novell provides to perform eDirectory partition and replica operations, see [“LDAP Extension Functions” on page 315](#).

ber_alloc_t

Constructs and returns an empty BerElement.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

options

(IN) Specifies the options used to create a BerElement.

Return Values

Returns a newly created BerElement on success; otherwise, returns a NULL pointer on failure.

Remarks

The options parameter specifies a bitwise OR of options to be used when encoding a new BerElement. You should always supply the following option:

LBER_USE_DER 0x01	Specifies that lengths will always be encoded in the minimum number of octets. However, this option does not cause values of sets to be rearranged in tag and byte order or for default values to be removed, so these options are not sufficient for generating DER output as defined in the X.509 and X.680 specifications. If you order set values and remove default values correctly, you can produce output according to the defined specifications.
-------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Unrecognized option bits are ignored.

Calls to the ber_printf function append bytes to the end of the BerElement.

Each BerElement structure allocated by the ber_alloc_t function should be freed by a call to the function.

See Also

[ber_free \(page 66\)](#)

ber_bvdup

Returns a copy of a berval structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

bv

(IN) Points to a structure to return.

Return Values

Returns a pointer to a berval structure on success; otherwise, returns NULL on failure.

Remarks

The bv_val field in the returned berval structure points to a different area of memory than the original bv_val field of the bv parameter.

The berval structures created by the ber_bvdup function should be freed by a call to the ber_bvfree function.

See Also

[ber_bvfree \(page 63\)](#)

ber_bvecfree

Frees an array of returned berval structures.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

bv

(IN) Points to the array of berval structures that are to be freed.

Remarks

Each structure in the array is freed by calling the ber_bvfree function, then the array itself is freed.

If the bv parameter is NULL, the ber_bvfree function does nothing.

ber_bvfree

Frees a returned berval structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

bv

(IN) Points to the berval structure to be freed.

Remarks

Both the bv_val string in the berval structure and the structure itself are freed.

If the bv parameter is NULL, this function does nothing.

ber_first_element

Positions the state of a BerElement to its first element and returns the type of the first element.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
ber_tag_t ber_first_element (
    BerElement  *ber,
    ber_len_t   *lenPtr,
    char        **opaquePtr);
```

Parameters

ber

(IN) Points to the first element in the constructed type.

lenPtr

(OUT) Is used for internal use only. Use this value in the subsequent call to the ber_next_element function.

opaquePtr

(OUT) Is used for internal use only. Use this value in the subsequent call to the ber_next_element function

Return Values

On success, returns a tag indicating the type of the first element. Returns LBER_DEFAULT if there are no elements.

Remarks

Use the ber_scanf function to obtain the value of the first element.

See Also

[ber_scanf \(page 72\)](#)

ber_flatten

Allocates a berval structure whose contents are a BER encoding of the specified BerElement.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
int ber_flatten (
    BerElement      *ber,
    struct berval    **bvPtr);
```

Parameters

ber

(IN) Points to the encoded contents for a BerElement.

bvPtr

(OUT) Points to the returned berval structure.

Return Values

Returns zero on success; otherwise, returns -1 on failure.

Remarks

The berval structure should be freed by calling the ber_bvfree function.

The ber_flatten function returns -1 if all '{' and '}' format modifiers are not properly matched.

Ber_init and ber_flatten are opposite functions. Ber_init converts a berval to a BerElement, and ber_flatten converts a BerElement to a berval.

See Also

[ber_bvfree \(page 63\)](#), [ber_init \(page 67\)](#)

ber_free

Frees a BerElement structure allocated by the ber_alloc_t or the ber_init function.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

ber

(IN) Points to a BerElement to be freed.

fbuf

(IN) Flag indicating if the internal buffer associated with the BerElement should also be freed.

1 frees the internal buffer, 0 does not free it.

Remarks

BerElements allocated by the library and returned to the application should be freed.

Note that when ldap_first_attribute returns a BerElement, it should be freed with ber_free(ber, 0). The internal buffer should not be freed since it points to the original searchResults.

If the ber parameter is NULL, the ber_free function does nothing.

See Also

[ber_alloc_t \(page 60\)](#), [ber_init \(page 67\)](#)

ber_init

Allocates and initializes a new BerElement structure with a copy of the data in the given berval structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

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

Parameters

bv

(IN) Points to the berval structure with which to initialize the new BerElement.

Return Values

Returns a new BerElement with the specified data on success; otherwise, returns a NULL pointer on failure.

Remarks

BerElements allocated with the ber_init function should be freed by calling the ber_free function.

Ber_init and ber_flatten are opposite functions. Ber_init converts a berval to a BerElement, and ber_flatten converts a BerElement to a berval.

See Also

[ber_free \(page 66\)](#), [ber_flatten \(page 65\)](#)

ber_next_element

Positions the state of a BerElement to the next element and returns its type.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
ber_tag_t ber_next_element (  
    BerElement  *ber,  
    ber_len_t   *lenPtr,  
    char        *opaquePtr);
```

Parameters

ber

(IN) Points to a BerElement structure.

lenPtr

(OUT) Is used for internal use only. Points to the value returned by the ber_first_element function. On subsequent calls, points to the value returned by the ber_next_element function.

opaquePtr

(OUT) Is used for internal use only. Points to the value returned by the ber_first_element function. On subsequent calls, points to the value returned by the ber_next_element function.

Return Values

On success, returns a tag indicating the type of the next element. Returns LBER_DEFAULT if there are no further elements.

Remarks

Use the ber_scanf function to obtain the value of the element.

See Also

[ber_scanf \(page 72\)](#)

ber_peek_tag

Returns the tag and length of the next element in a BerElement.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
ber_tag_t ber_peek_tag (  
    BerElement  *ber,  
    ber_len_t   *lenPtr);
```

Parameters

ber

(IN) Points to the BerElement.

lenPtr

(OUT) Points to the length of next element to be parsed.

Return Values

Returns the tag of the next element to be parsed on success; returns LBER_DEFAULT if there is no further data to be read.

Remarks

The decoding position within the ber parameter is not changed and will not affect the future use of the ber parameter.

ber_printf

encodes data items into a BerElement.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
int ber_printf (
    BerElement *ber,
    const char *fmt,
    ...);
```

Parameters

ber

(IN) Points to a BerElement.

fmt

(IN) Points to a format string.

...

(IN) Specifies data values for each tag in the format string.

Return Values

Returns a nonnegative number on success; otherwise, returns -1 on failure.

Remarks

The `ber_printf` function encodes a BerElement in a similar manner as the `sprintf` function. However, the `ber_printf` function must keep state information in the `ber` parameter so that this function can be called subsequently to append information to the end of a BerElement.

Similar to the `sprintf` function, each character in the `fmt` parameter refers to an argument to the `ber_printf` function.

The `fmt` parameter can have the following characters.

b	Boolean	The next parameter is a <code>ber_int_t</code> , which contains either 0 for False or 0xff for True.
B	Bitstring	The next two parameters are a <code>char*</code> pointer to the start of the bitstring, followed by a <code>ber_len_t</code> , which contains the number of bits in the bitstring. A bitstring element is output in primitive form. If this character is not preceded by the 't' modifier, the 0x03U tag is used for the element.

e	Enumerated	The next parameter is a <code>ber_int_t</code> , which contains the enumerated value in the host's byte order. An enumerated element is output. If this character is not preceded by the 't' modifier, the 0x0AU tag is used for the element.
i	Integer	The next parameter is a <code>ber_int_t</code> , which contains the integer in the host's byte order. An integer element is output. If this character is not preceded by the 't' modifier, the 0x02U tag is used for the element.
n	NULL	No parameter is needed. An ASN.1 NULL element is output. If this character is not preceded by the 't' modifier, the 0x05U tag is used for the element.
o	Octet string	The next two parameters are a <code>char*</code> pointer, followed by a <code>ber_len_t</code> that contains the length of the string. The string can contain NULL bytes and do not have to be zero terminated. An octet string element is output in primitive form. If this character is not preceded by the 't' modifier, the 0x04U tag is used for the element.
O	Octet string	The next parameter is a pointer to a <code>berval</code> structure. If this character is not preceded by the 't' modifier, the 0x04U tag is used for the element.
s	Octet string	The next parameter is a <code>char*</code> pointer to a zero-terminated string. An octet string is output in primitive form and does not include the trailing '\0' (NULL) byte. If this character is not preceded by the 't' modifier, the 0x04U tag is used for the element.
t	Tag	The next parameter is a <code>ber_tag_t</code> , which specifies the tag to override the next element to be written to the <code>BerElement</code> .
v	Several octet strings	The next parameter is a <code>char**</code> , an array of <code>char*</code> pointers to zero-terminated strings. The last element in the array must be a NULL pointer. The octet strings do not include the trailing '\0' (NULL) byte. A construct similar to '{v}' is used to get an actual sequence of octet strings. The 't' modifier cannot be used with this character.
V	Several octet strings	A NULL-terminated array of <code>berval</code> structure pointers is supplied. Note that a construct similar to '{V}' is used to get an actual sequence of octet strings. The 't' modifier cannot be used with this character.
{	Begin sequence	No parameter is needed. If this character is not preceded by the 't' modifier, the 0x30U tag is used for the element.
}	End sequence	No parameter is needed. The 't' modifier cannot be used with this character.
[Begin set	No parameter is needed. If this character is not preceded by the 't' modifier, the 0x31U tag is used for the element.
]	End set	No parameter is needed. The 't' modifier cannot be used with this character.

Each use of a '{' character should be matched with a '}' character, either later in the format string or in the format string of a subsequent call to `ber_printf` for that specific `BerElement`. The same rules applies to the '[' and ']' characters.

ber_scanf

Decodes a BerElement, similar to the sscanf function.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
ber_tag_t ber_scanf (  
    BerElement *ber,  
    const char *fmt,  
    ...);
```

Parameters

ber

(IN) Points to a BerElement returned by the ber_init function.

fmt

(IN) Points to the format modifiers to use when interpreting the BerElement bytes.

...

(OUT) Returns pointers to data values returned by the function.

Return Values

Returns a non-LBER_ERROR value on success; otherwise, returns LBER_ERROR on failure.

Remarks

The ber_scanf function keeps some of the state information with the ber parameter so that the ber_scanf function can be called iteratively to sequentially read from the BerElement.

The results of successfully calling the ber_scanf function are stored in additional parameters.

The fmt parameter can have the following values.

a	Octet string	A char* pointer must be supplied. Memory is allocated and filled with the contents of the octet string (zero-terminated). The pointer to the string is stored in the parameter. The returned value should be freed by calling the ldap_memfree function. The element tag must indicate the primitive form (constructed strings are not supported) but is otherwise ignored and discarded during the decoding. This character cannot be used with octet string that contain NULL bytes.
---	--------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

b	Boolean	A pointer to <code>ber_int_t</code> must be supplied. The stored value will be zero for FALSE or nonzero for TRUE. The element tag must indicate the primitive form but is otherwise ignored during the decoding.
B	Bitstring	A <code>char**</code> parameter must be supplied that will point to the allocated bits. This is followed by a <code>ber_len_t*</code> parameter that will point to the length (in bits) of the returned bitstring. The <code>ldap_memfree</code> function should be called to free the bitstring. The element tag must indicate the primitive form (constructed bitstrings are not supported) but is otherwise ignored during the decoding.
e	Enumerated	A pointer to <code>ber_int_t</code> must be supplied. The stored value will be in host byte order. The element tag must indicate the primitive form but is otherwise ignored during the decoding. The <code>ber_scanf</code> function returns an error if the enumerated value cannot be stored in a <code>ber_int_t</code> .
i	Integer	A pointer to <code>ber_int_t</code> must be supplied. The stored value will be in host byte order. The element tag must indicate the primitive form but is otherwise ignored during the decoding. The <code>ber_scanf</code> function returns an error if the integer cannot be stored in a <code>ber_int_t</code> .
L	Length	A pointer to a <code>ber_len_t</code> must be supplied. The length of the next element in bytes is returned.
n	NULL	No parameter is needed. The element is verified to have a zero-length value and is skipped. The tag is ignored.
o	Octet string	A <code>berval *</code> parameter must be supplied, pointing to an existing empty <code>berval</code> structure. The buffer inside the <code>berval</code> is allocated as required and should be freed with the <code>ldap_memfree</code> function when done.
O	Octet string	A <code>berval **</code> parameter must be supplied, which will point to an allocated <code>berval</code> structure that contains the octet string and its length upon return. The <code>ber_bvfree</code> function should be called to free the allocated memory. The element tag must indicate the primitive form (constructed strings are not supported) but is otherwise ignored during the decoding.
s	Octet string	A <code>char *</code> buffer must be supplied, point to an existing buffer. It must be followed by a <code>ber_len_t *</code> parameter. The object of this pointer contains the size of the buffer on input and is replaced with the size of the data written to the buffer on output.
t	Tag	A pointer to <code>ber_tag_t</code> must be supplied. The stored value will be the tag of the next element in the <code>BerElement</code> <code>ber</code> parameter and represented so it can be written using the 't' modifier of the <code>ber_printf</code> function. The decoding position within the <code>ber</code> parameter is not changed and can be used in the future.
v	Several octet strings	A <code>char***</code> parameter must be supplied, which points to an allocated, NULL-terminated array of <code>char*</code> pointers that contain the octet strings upon return. NULL is stored if the sequence is empty. The <code>ldap_memfree</code> function should be called to free each element of the array and the array itself. The sequence tag and the octet string tags are ignored.
V	Several octet strings	A <code>berval***</code> structure pointer must be supplied, which points to an allocated, NULL-terminated array of <code>berval*</code> structure pointers that contain the octet strings and their lengths upon return. NULL is stored if the sequence is empty. The <code>ber_bvecfree</code> function can be called to free the allocated memory. The sequence tag and the octet string tags are ignored.
x	Skip element	The next element is skipped. No parameter is needed.
{	Begin sequence	No parameter is needed. The initial sequence tag and length are skipped.
}	End sequence	No parameter is needed.

[Begin set	No parameter is needed. The initial set tag and length are skipped.
]	End set	No parameter is needed.

ber_skip_tag

Skips the next element of a BerElement, returning its length and tag.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

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

```
ber_tag_t ber_skip_tag (  
    BerElement *ber,  
    ber_len_t *lenPtr);
```

Parameters

ber

(IN) Points to the BerElement.

lenPtr

(OUT) Points to the length of the skipped element.

Return Values

Returns the tag of the element that was skipped on success; otherwise, returns LBER_DEFAULT if there is no further data to be read.

ldap_abandon

Abandons an asynchronous LDAP operation already in progress. This function has been deprecated; LDAP v3 clients should use [ldap_abandon_ext](#) (page 78).

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_abandon (
    LDAP      *ld,
    int       msgid);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgid

(IN) Specifies the message ID of the asynchronous LDAP operation to abandon.

Return Values

0	Success
-1	Failure

Remarks

The msgid parameter must specify a message ID returned by an outstanding asynchronous LDAP operation, such as ldap_search or ldap_modify.

The ldap_abandon function checks to see if the results of the operation has already come in.

- If not, it sends an LDAP abandon operation to the LDAP server.
- If the results have already come in, the LDAP operation cannot be abandoned.

If the ldap_abandon function returns -1, use the ldap_get_option function with the option parameter set to LDAP_OPT_RESULT_CODE to retrieve the error code from the LDAP session handle.

See Also

[ldap_abandon_ext](#) (page 78)

ldap_abandon_ext

Abandons an asynchronous LDAP operation already in progress using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_abandon_ext (
    LDAP      *ld,
    int        msgid,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgid

(IN) Specifies the message ID of the asynchronous LDAP operation to abandon.

serverctrls

(IN) Points to a list of LDAP server controls to use with the abandon operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the abandon operation. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x51	LDAP_SERVER_DOWN
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR

Remarks

The msgid parameter must specify a message ID returned by an outstanding asynchronous LDAP operation, such as `ldap_search` or `ldap_modify`.

The `ldap_abandon` function checks to see if the results of the operation has already come in.

- If not, it sends an LDAP abandon operation to the LDAP server.
- If the results have already come in, the LDAP operation cannot be abandoned.

eDirectory does not currently support any controls to use with an abandon operation.

See Also

[ldap_abandon \(page 76\)](#)

ldap_add

Asynchronously adds an entry to the directory. This function has been deprecated; LDAP v3 clients should use [ldap_add_ext](#) (page 82).

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_add (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **attrs);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to add, for example: "o=novell", "ou=provo", "cn=kim"

All components of the dn must exist except for the leaf component. The leaf component name must be unique within the container.

attrs

(IN) Points to a NULL terminated array of LDAPMod structures that contain the attributes and value to add with the entry. All mandatory attributes must have values or the operation fails.

Return Values

>0	Message ID of request
-1	Failure

Remarks

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

For a list of mandatory attributes for an entry see the LDAP server's schema. For eDirectory, see *Developer Kit*.

See Also

[ldap_add_s](#) (page 87), [ldap_add_ext](#) (page 82), [ldap_add_ext_s](#) (page 84), [ldap_modify](#) (page 179)

ldap_add_ext

Asynchronously adds an entry to the directory using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_add_ext (
    LDAP          *ld,
    const char     *dn,
    LDAPMod        **attrs,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    int            *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to add, for example: "o=novell", "ou=provo", "cn=kim"

All components of the dn must exist except for the leaf component. The leaf component name must be unique within the container.

attrs

(IN) Points to a NULL-terminated array of LDAPMod structures that contain the attributes and values to add with the entry. All mandatory attributes must have values or the operation fails.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the add. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the add. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the add request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

For a list of mandatory attributes for an entry see the LDAP server's schema. For eDirectory, see *[Developer Kit](#)*.

If you are adding an entry that logs in to the directory, you need to set a value for the `userPassword` attribute. The `userPassword` attribute is not a mandatory attribute. However, if you create an entry without a `userPassword` attribute, the entry cannot log in.

eDirectory does not currently support any server-side controls to use with adding entries.

See Also

[ldap_add](#) (page 80), [ldap_add_ext_s](#) (page 84), [ldap_add_s](#) (page 87)

ldap_add_ext_s

Synchronously adds an entry to the directory using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_add_ext_s (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **attrs,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to add, for example: "o=novell", "ou=provo", "cn=kim"

All components of the dn must exist except for the leaf component. The leaf component name must be unique within the container.

attrs

(IN) Points to a NULL-terminated array of LDAPMod structures that contain the attributes and values to add with the entry. All mandatory attributes must have values or the operation fails.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the add. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the add. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the add request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x01	LDAP_OPERATIONS_ERROR
0x02	LDAP_PROTOCOL_ERROR
0x08	LDAP_STRONG_AUTH_REQUIRED
0x11	LDAP_UNDEFINED_TYPE
0x13	LDAP_CONSTRAINT_VIOLATION
0x14	LDAP_TYPE_OR_VALUE_EXISTS
0x15	LDAP_INVALID_SYNTAX
0x0A	LDAP_REFERRAL
0x0C	LDAP_UNAVAILABLE_CRITICAL_EXTENSION
0x0D	LDAP_CONFIDENTIALITY_REQUIRED
0x20	LDAP_NO_SUCH_OBJECT
0x22	LDAP_INVALID_DN_SYNTAX
0x32	LDAP_INSUFFICIENT_ACCESS
0x33	LDAP_BUSY
0x35	LDAP_UNWILLING_TO_PERFORM
0x36	LDAP_LOOP_DETECT
0x40	LDAP_NAMING_VIOLATION
0x41	LDAP_OBJECT_CLASS_VIOLATION
0x44	LDAP_ALREADY_EXISTS
0x50	LDAP_OTHER
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

For a list of mandatory attributes for an entry see the LDAP server's schema. For eDirectory, see *Developer Kit*.

If you are adding an entry that logs in to the directory, you need to set a value for the userPassword attribute. The userPassword attribute is not a mandatory attribute. However, if you create an entry without a userPassword attribute, the entry cannot log in.

eDirectory does not currently support any server-side controls to use with adding entries.

For sample code, see [addentry.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) and [addentry1.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_add](#) (page 80), [ldap_add_s](#) (page 87), [ldap_add_ext](#) (page 82),

ldap_add_s

Synchronously adds an entry to the directory.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_add_s (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **attrs);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to add, for example: "o=novell", "ou=provo", "cn=kim"

All components of the dn must exist except for the leaf component. The leaf component name must be unique within the container.

attrs

(IN) Points to an array of LDAPMod structures that contain the attributes and values to add with the entry. All mandatory attributes must have values or the operation fails.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The ldap_add_s is an older function. LDAP v3 clients should use the ldap_add_ext_s function.

For a list of mandatory attributes for an entry see the LDAP server's schema. For eDirectory, see *Developer Kit*.

See Also

[ldap_add](#) (page 80), [ldap_add_ext](#) (page 82), [ldap_add_ext_s](#) (page 84), [ldap_modify](#) (page 179)

ldap_bind

Asynchronously authenticates a specified entry to the directory. This function has been deprecated; use the `ldap_simple_bind` function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

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

Parameters

ld

(IN) Points to the handle for the LDAP session which is returned by either the `ldap_open` or `ldap_init` function.

dn

(IN) Points to the distinguished name of the entry to use for authentication, for example: "o=novell", "ou=provo", "cn=kim"

cred

(IN) Points to the credentials to use for authentication

method

(IN) Specifies the authentication method. eDirectory supports the following methods:

- `LDAP_AUTH_NONE` (0x00)—no authentication
- `LDAP_AUTH_SIMPLE` (0x80)—context specific + primitive

Return Values

>0	Message ID of operation
-1	Failure

See Also

[ldap_simple_bind](#) (page 270), [ldap_unbind](#), [ldap_unbind_s](#) (page 278), [ldap_unbind_ext](#), [ldap_unbind_ext_s](#) (page 279)

ldap_bind_digest_md5_start

Begins the DIGEST-MD5 SASL bind process.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_bind_digest_md5_start (
    LDAP          *ld,
    LDAP_DIGEST_MD5_CONTEXT *digestMD5ctx
);
```

Parameters

ld

(IN) the handle for the LDAP session.

digestMD5ctx

(IN) A pointer to an LDAP_DIGEST_MD5_CONTEXT variable that will be initialized with a new DIGEST-MD5 login context. This context must be used in a subsequent call to [ldap_bind_digest_md5_finish](#) (page 93).

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.

Remarks

The LDAP_OPT_NETWORK_TIMEOUT option (set by calling [ldap_set_option](#) (page 266)) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by spaces in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing NULL for the ld parameter of [ldap_set_option](#) sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init](#) (page 174) or [ldapsl_init](#) (page 297). To clear the timeout pass NULL for the inval parameter of [ldap_set_option](#).

A connection timeout will cause an LDAP_SERVER_DOWN error (81) "Can't contact LDAP server".

See Also

[ldap_get_digest_md5_realms \(page 160\)](#), [ldap_bind_digest_md5_finish \(page 93\)](#)

ldap_bind_digest_md5_finish

Finishes a DIGEST-MD5 bind started by a call to [ldap_bind_digest_md5_start](#) (page 91). It must also be called if the application must abort the bind sequence after calling `ldap_bind_digest_md5_start`.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_bind_digest_md5_finish (
    LDAP_DIGEST_MD5_CONTEXT *digestMD5ctx,
    char *authID,
    char *password,
    int passwordLen,
    int realmIndex,
    int abortFlag
);
```

Parameters

digestMD5ctx

(IN) The DIGEST-MD5 context created by a call to `ldap_bind_digest_md5_begin_s`. The function will set the context pointer to NULL.

authID

(IN) A NULL-terminated UTF-8 encode string containing the properly formatted authorization identity for the user to be authenticated.

password

(IN) A NULL-terminated UTF-8 encode string containing the user's password.

passwordLen

(IN) The length in bytes of the password. This is required to allow passwords that have embedded NULL bytes. If the password is known to be a NULL-terminated string, the `passwordLen` value can be set to minus one (-1) or the length of the string.

realmIndex

(IN) This is the index of the realm selected by the client application.

abortFlag

(IN) Must be equal to `DIGEST_MD5_ABORT` or `DIGEST_MD5_FINISH`.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.

Remarks

If abortFlag is equal to DIGEST_MD5_FINISH, the function attempts to complete the bind sequence with the server and then frees any memory allocated during the bind process. If abortFlag is equal to DIGEST_MD5_ABORT, the function sends a SASL bind request to the server with a zero length string for the mechanism and no credentials. This signals the server that the bind sequence was aborted by the client. Any allocated memory is also freed.

See Also

[ldap_bind_digest_md5_start](#) (page 91)

ldap_bind_nmas_s

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_nmas_bind_s (
    LDAP          *ld,
    LDAP_CONST    char *dn,
    LDAP_CONST    char *password,
    LDAP_CONST    char *reqSequence,
    LDAP_CONST    char *reqClearance
);
```

Parameters

ld

(IN) the handle for the LDAP session.

dn

(IN) The dn of the user to be authenticated.

password

(IN) The users password if the requested sequence allows for a password.

reqSequence

(IN) The NMAS login sequence to be used. May be NULL.

reqClearance

(IN) The clearance requested by the client. May be NULL.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.

Remarks

If `abortFlag` is equal to `DIGEST_MD5_FINISH`, the function attempts to complete the bind sequence with the server and then frees any memory allocated during the bind process. If `abortFlag` is equal to `DIGEST_MD5_ABORT`, the function sends a SASL bind request to the server with a zero length string for the mechanism and no credentials. This signals the server that the bind sequence was aborted by the client. Any allocated memory is also freed.

The `LDAP_OPT_NETWORK_TIMEOUT` option (by calling [ldap_set_option \(page 266\)](#) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by commas in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing `NULL` for the `ld` parameter of `ldap_set_option` sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init \(page 174\)](#) or [ldapssl_init \(page 297\)](#). To clear the timeout pass `NULL` for the `invalue` parameter of `ldap_set_option`.

A connection timeout will cause an `LDAP_SERVER_DOWN` error (81) "Can't contact LDAP server".

ldap_bind_s

Synchronously authenticates a specified entry to the directory. This function has been deprecated; use the [ldap_simple_bind_s](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

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

Parameters

ld

(IN) Points to the handle for the LDAP session which is returned by either the `ldap_open` or `ldap_init` function.

dn

(IN) Points to the distinguished name of the entry to use for authentication, for example: "o=novell", "ou=provo", "cn=kim"

cred

(IN) Points to the credentials to use for authentication

method

(IN) Specifies the authentication method. eDirectory supports the following methods:

- `LDAP_AUTH_NONE` (0x00)—no authentication
- `LDAP_AUTH_SIMPLE` (0x80)—context specific + primitive

Return Values

0x00	<code>LDAP_SUCCESS</code>
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	<code>LDAP_DECODING_ERROR</code>
0x56	<code>LDAP_AUTH_UNKNOWN</code>

0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

See Also

[ldap_simple_bind](#) (page 270), [ldap_unbind](#), [ldap_unbind_s](#) (page 278), [ldap_unbind_ext](#), [ldap_unbind_ext_s](#) (page 279)

ldap_cancel_ext

Cancels an asynchronous LDAP operation already in progress using LDAP client or server controls. The LDAP Cancel operation should be used instead of the LDAP abandon operation when the client needs to know the results.

LDAP Version: v3

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>int ldap_cancel_ext (  
LDAP *ld,  
int msgid,  
LDAPControl **serverctrls,  
LDAPControl **clientctrls,  
Int          *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgid

(IN) Specifies the message ID of the asynchronous LDAP operation to cancel.

serverctrls

(IN) Points to a list of LDAP server controls to use with the abandon operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the abandon operation. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request if the search request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x51	LDAP_SERVER_DOWN
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR

0x5A	LDAP_NO_MEMORY
0x76	LDAP_CANCELED
0x77	LDAP_CANCEL_NO_SUCH_OPERATION
0x78	LDAP_CANCEL_TOO_LATE
0x79	LDAP_CANCEL_CANNOT_CANCEL

Remarks

The `msgid` parameter must specify a message ID returned by an outstanding asynchronous LDAP operation, such as `ldap_search` or `ldap_modify`.

The `ldap_cancel_ext` function checks to see if the results of the operation have already come in.

- If not, it sends an LDAP cancel operation to the LDAP server.
- If the results have already come in, the LDAP operation cannot be cancelled.

eDirectory currently does not support any controls to use with a cancel operation.

See Also

[ldap_add_ext_s \(page 84\)](#)

ldap_cancel_ext_s

Synchronously Cancels an asynchronous LDAP operation already in progress using LDAP client or server controls. The LDAP Cancel operation should be used instead of the LDAP abandon operation when the client needs to know the result.

LDAP Version: v3

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>int ldap_cancel_ext_s (  
LDAP *ld,  
int msgid,  
LDAPControl **serverctrls,  
LDAPControl **clientctrls,  
);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgid

(IN) Specifies the message ID of the asynchronous LDAP operation to cancel.

serverctrls

(IN) Points to a list of LDAP server controls to use with the abandon operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the abandon operation. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x51	LDAP_SERVER_DOWN
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0X76	LDAP_CANCELED

0X77	LDAP_CANCEL_NO_SUCH_OPERATION
0X78	LDAP_CANCEL_TOO_LATE
0X79	LDAP_CANCEL_CANNOT_CANCEL

Remarks

The msgid parameter must specify a message ID returned by an outstanding asynchronous LDAP operation, such as `ldap_search` or `ldap_modify`.

The `ldap_cancel_ext_s` function checks to see if the results of the operation have already come in.

- If not, it sends an LDAP cancel operation to the LDAP server.
- If the results have already come in, the LDAP operation cannot be cancelled.

eDirectory currently does not support any controls to use with a cancel operation.

See Also

[ldap_cancel_ext \(page 99\)](#)

ldap_compare

Asynchronously determines whether a specified entry contains a specified attribute value.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_compare (
    LDAP      *ld,
    const char *dn,
    const char *attr,
    const char *value);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry whose attribute is being compared.

attr

(IN) Points to the name of the attribute to compare.

value

(IN) Points to a string value of the attribute to compare.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

This function compares the specified value with the values in the entry's attribute. The results specify whether a match is found.

The ldap_compare function is an older function. LDAP v3 clients should use the ldap_compare_ext function.

The `ldap_compare` function can compare only attributes with string values. Use `ldap_compare_ext` to compare binary values.

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

Compare operations are faster than search operations. Whenever possible in your application, use a compare rather than a search operation.

See Also

[ldap_compare_ext \(page 105\)](#)

ldap_compare_ext

Asynchronously determines whether a specified entry contains a specified attribute value. LDAP client or server controls can be used with the compare.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_compare_ext (
    LDAP          *ld,
    const char     *dn,
    const char     *attr,
    const struct berval *bvalue,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    int            *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry whose attribute is being compared.

attr

(IN) Points to the name of the attribute to compare.

bvalue

(IN) Points to a berval structure that contains the attribute's value to compare with the entry's attribute value.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the search. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the search. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the search request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

This function compares the specified value with the values in the entry's attribute. The results specify whether a match is found.

The `ldap_compare_ext` function can be used to compare any type of data. For string data, you can use the `ldap_compare` function.

The data returned in `msgidp` is opaque to the caller. To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

Compare operations are faster than search operations. Whenever possible in your application, use a compare rather than a search operation.

eDirectory does not currently support any server controls to use with compare operations.

See Also

[ldap_compare](#) (page 103)

ldap_compare_ext_s

Synchronously determines whether a specified entry contains a specified attribute value. LDAP client or server controls can be used with the compare.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_compare_ext_s (
    LDAP          *ld,
    const char     *dn,
    const char     *attr,
    const struct berval *bvalue,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry whose attribute is being compared.

attr

(IN) Points to the name of the attribute to compare.

bvalue

(IN) Points to berval structure that contains the attribute's value to compare with the entry's attribute value.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the search. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the search. Use NULL to specify no client controls.

Return Values

0x05	LDAP_COMPARE_FALSE: the entry does not contain the attribute value.
0x06	LDAP_COMPARE_TRUE: the entry contains the attribute value
Non-zero value other than 0x05 or 0x06	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

This function compares the specified value with the values in the entry's attribute and returns whether a match is found.

The `ldap_compare_ext_s` function can be used to compare any type of data. For string data, you can use `ldap_compare_s`.

Compare operations are faster than search operations. Whenever possible in your application, use a compare rather than a search operation.

eDirectory does not currently support any server controls to use with compare operations.

For sample code, see `cpatrs.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_compare_s](#) (page 109)

ldap_compare_s

Synchronously determines whether a specified entry contains a specified attribute value.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_compare_s (
    LDAP      *ld,
    const char *dn,
    const char *attr,
    const char *value);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry whose attribute is being compared.

attr

(IN) Points to the name of the attribute to compare.

value

(IN) Points to a string value of the attribute to compare.

Return Values

0x05	LDAP_COMPARE_FALSE: the entry does not contain the attribute value.
0x06	LDAP_COMPARE_TRUE: the entry contains the attribute value
Non-zero value other than 0x05 or 0x06	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The `ldap_compare_s` function takes the attribute and its value and compares them to those found in the specified entry (dn).

The `ldap_compare_s` function is an older function. LDAP v3 clients should use the `ldap_compare_ext_s` function.

This function can compare only attributes with string values. Use `ldap_compare_ext_s` to compare binary values.

Compare operations are faster than search operations. Whenever possible in your application, use a compare rather than a search operation.

See Also

[ldap_compare_ext \(page 105\)](#)

ldap_control_free

Frees an LDAPControl structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_control_free (
    LDAPControl *ctrl);
```

Parameters

ctrl

(IN) Points to the control structure to free.

Remarks

If you have created a control, you should call this function to free the structure when you are finished with the control.

See Also

[ldap_controls_free \(page 112\)](#)

ldap_controls_free

Frees an array of LDAPControl structures.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
void ldap_controls_free (  
    LDAPControl **ctrls);
```

Parameters

ctrls

(IN) Points to an array of control structures.

Remarks

You should call this function to free any arrays of controls that you create or that are returned to you by other functions such as `ldap_parse_result`.

See Also

[ldap_control_free](#) (page 111)

ldap_count_entries

Returns the number of LDAPMessage structures that are of the type LDAP_RES_SEARCH_ENTRY.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_count_entries (
    LDAP      *ld,
    LDAPMessage *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result message chain returned by the ldap_result function or a synchronous search function.

Return Values

>0	Number of entries
0	No more entries
-1	Failure

Remarks

The ldap_count_entries function can be used to count the number of message structures that remain in a chain. Messages are removed from the chain by calling one of the following functions:

- ldap_first_message
- ldap_next_message
- ldap_first_entry
- ldap_next_entry

This function counts from the current position of the pointer to the end of the chain.

- If you pass a pointer that points to the first message structure in the chain, it counts all the entries in the chain.
- If you pass a pointer that points to a structure in the middle of the chain, it counts the entries from that point to the end of the chain.

See Also

[ldap_first_entry](#) (page 153), [ldap_next_entry](#) (page 201), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_count_messages

Returns the number of LDAPMessage structures of any type in an LDAP message chain.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_count_messages (
    LDAP      *ld,
    LDAPMessage *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result message chain returned by the ldap_result function or a synchronous search function.

Return Values

>0	Number of messages in the chain
0	No more messages
-1	Failure

Remarks

The ldap_count_messages function can be used to count the number of message structures that remain in a chain. The following functions are used to iterate through the chain:

- ldap_first_message
- ldap_next_message
- ldap_first_reference
- ldap_next_reference

This function counts from the current position of the pointer to the end of the chain.

- If you pass a pointer that points to the first message structure in the chain, it counts all the messages in the chain.
- If you pass a pointer that points to a structure in the middle of the chain, it counts the messages from that point to the end of the chain.

See Also

[ldap_first_message](#) (page 155), [ldap_next_message](#) (page 203), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_count_references

Returns the number of LDAPMessage structures in an LDAP result message chain that are of type LDAP_RES_SEARCH_REFERENCE.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_count_references (
    LDAP      *ld,
    LDAPMessage *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result chain returned by the ldap_result function or a synchronous search function.

Return Values

>0	Number of references
0	No more references
-1	Failure

Remarks

This function counts from the current position of the pointer to the end of the chain.

- If you pass a pointer that points to the first message structure in the chain, it counts all the references in the chain.
- If you pass a pointer that points to a structure in the middle of the chain, it counts the references from that point to the end of the chain.

See Also

[ldap_first_reference](#) (page 156), [ldap_next_reference](#) (page 204), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_count_values

Returns the number of strings in the array.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_count_values (
    char **vals);
```

Parameters

vals

(IN) Points to the array of values returned by the `ldap_get_values` or `ldap_get_values_len` function.

Return Values

>0	Number of values
-1	Failure

Remarks

The `ldap_count_values` function can be used for attributes that have character string values. If the array contains berval structures (binary data), use the `ldap_count_values_len` function.

See Also

[ldap_get_values_len](#) (page 169), [ldap_get_values](#) (page 167), [ldap_count_values_len](#) (page 120), [ldap_value_free](#) (page 292), [ldap_value_free_len](#) (page 293)

ldap_count_values_len

Returns the number of berval structures in the array.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_count_values_len (
    struct berval **vals);
```

Parameters

vals

(IN) Points to the array of values returned by the ldap_get_values or ldap_get_values_len function.

Return Values

>0	Number of values
-1	Failure

Remarks

The ldap_count_values_len function can be used to count the number of values for attributes that have binary data. Use ldap_count_values to count string attribute values.

The memory for the vals parameter is dynamically allocated. When you are done with the array, free the memory by calling the ldap_value_free_len function.

See Also

[ldap_get_values_len](#) (page 169), [ldap_get_values](#) (page 167), [ldap_count_values](#) (page 119), [ldap_value_free](#) (page 292), [ldap_value_free_len](#) (page 293)

ldap_create_geteffective_control

Creates and encodes a get effective privilege control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>ldap_create_geteffective_control (
    LDAP          *ld,
    LDAPGetprvInfo **getprvinfo,
    int            efPrvvalue,
    int            isCritical,
    LDAPControl **ctrlp );
```

Parameter

ld

(IN) Points to the handle for the LDAP session obtained from a call to ldap_init().

Getprvinfo

(IN) Points to a null-terminated array of pointers to LDAPGetprvInfo structures, containing a description of each of the EffectivePrivilege value selection type.

efPrvvalue

(IN) Specifies a bool value indicating includeAllLegalAttributes value is selected. -1 indicates that the includeAllLegalAttributes value is not selected.

isCritical

(IN) Indicates the criticality of the control to the operation. 0 indicates that the control is not critical to the operation and a non-zero values indicates that the control is critical to the operation.

ctrlp

(OUT) Returns a pointer to the created LDAPControl. This control is free from calling the ldap_control_free() after returning the pointer.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”

Remarks

The `ldap_create_geteffective_control` creates a sort control, that can be used as the server control parameter in the `ldap_search_ext` and the `ldap_search_ext_s` functions.

ldap_create_persistentsearch_control

Creates and encodes a persistent search control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.5 or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_create_persistentsearch_control (
    LDAP          *ld,
    int            changeTypes,
    int            changesOnly,
    int            returnEchgCtls,
    char           isCritical,
    LDAPControl    **ctrlp);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

changeTypes

(IN) an integer whose value is the bit-wise OR of the flag values corresponding to the changes types for which a the client wishes to be notified. Valid flags are as follows:

LDAP_CHANGETYPE_ADD	specifies that you want to be notified when entries are added to the directory
LDAP_CHANGETYPE_DELETE	specifies that you want to be notified when entries are deleted from the directory
LDAP_CHANGETYPE_MODIFY	specifies that you want to be notified when entries are modified.
LDAP_CHANGETYPE_MODDN	specifies that you want to be notified when entries are renamed.
LDAP_CHANGETYPE_ANY	specifies that you want to be notified when any of the above changes are made.

changesOnly

(IN) If non-zero, the initial search is only used to establish a result set on the server. No results are returned from this initial search. As changes are subsequently made to entries in the result set, the server returns the changed entries to the client. If zero, both the results of the initial search and entries that are subsequently changed are returned.

returnEntryChangeCtrl

(IN) If non-zero, an entry change notification control is included with each entry. If 0, entry change notification controls are not included with the entries returned from the server.

isCritical

(IN) Specifies whether or not the persistent search control is critical to the search operation. If non-zero, the control is critical to the search operation. If the server does not support persistent searches, the server will return the error LDAP_UNAVAILABLE_CRITICAL_EXTENSION.

If 0, the control is not critical to the search operation. Even if the server does not support persistent searches, the search operation is still performed.]

ctrlp

(OUT) Points to a pointer for the LDAPControl structure which this function creates and which can be used in the search operation. When you are done with this control, its memory should be freed by calling the ldap_control_free function.

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This API creates an LDAP persistent search control using the supplied parameters. The control can then be used in a call to ldap_search_ext to request that the server perform a persistent search. A persistent search allows the client to be notified when changes are made to entries that satisfy the specified search filter. When a persistent search is performed, the connection to the server remains open until the client abandons the search or unbinds from the server. The timeout parameters to the search are ignored.

For example code, see [searchPersist.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_search_ext](#) (page 253), [ldap_parse_entrychange_control](#) (page 206)

ldap_create_reference_control

Creates and encodes a reference control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>ldap_create_reference_control (
    LDAP      *ld,
    int        isCritical,
    LDAPControl **ctrlp )
```

Parameters

ld

(IN) Points to the handle for the LDAP session obtained from a call to ldap_init().

isCritical

(IN) Indicates the criticality of the control to the operation. 0 indicates that the control is not critical to the operation and a non-zero values indicates that the control is critical to the operation.

ctrlp

(OUT) Returns a pointer to the created LDAPControl. This control is free from calling the ldap_control_free() after returning the pointer.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See " LDAP Return Codes "

Remarks

ldap_create_reference_control creates a continuity reference control, that can be used as the server control parameter in the ldap_search_ext and the ldap_search_ext_s functions.

ldap_create_sort_control

Creates and encodes a server-side sort control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_create_sort_control (
    LDAP      *ld,
    LDAPSortKey **keyList,
    int        isCritical,
    LDAPControl **ctrlp);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

keyList

(IN) Points to a NULL-terminated array of pointers to LDAPSortKey structures which contain the attributes to match and the rules to use for matching.

isCritical

(IN) Specifies whether the control is required for the search operation:

- Non-zero specifies that the control is required.
- Zero specifies that the search operation can be performed without the control.

ctrlp

(OUT) Points to a pointer for the LDAPControl structure which this function creates and which can be used in the search operation. When you are done with this control, its memory should be freed by calling the ldap_control_free function.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”.

Remarks

The `ldap_create_sort_control` function creates a sort control that you can use as the server control parameter in the `ldap_search_ext` and the `ldap_search_ext_s` functions.

For example code, see `sortcntl.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

`ldap_parse_sort_control` (page 221), `ldap_control_free` (page 111), `ldap_controls_free` (page 112)

ldap_create_sort_keylist

Creates an array of pointers to LDAPSortKey structures.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_create_sort_keylist (
    LDAPSortKey ***sortKeyList,
    char          *keyString);
```

Parameters

sortKeyList

(OUT) Points to a NULL-terminated array of pointers to LDAPSortKey structures which contain the attributes to sort on and the rules to use for ordering.

keyString

(IN) Points to a string representation of one or more sort keys, separated by spaces.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”.

Remarks

A key string uses the following format:

```
[ - ]attribute[:ordering rule]
```

The optional - indicates reverse sort order.

The attribute specifies an attribute in the LDAP server's schema.

The optional ordering rule is an OID (dotted string format) specifying the matching rule to use for sorting.

IMPORTANT: eDirectory currently supports only a single sort key, no ordering rules, and only forward sorting.

If the attribute corresponds to an existing index on the eDirectory server, performance is extremely good even with very large result sets. NDS 8 and NDS eDirectory have indexes for the following attributes:

sn (NDS name: Surname)
givenName (NDS name: Given Name)
cn (NDS name: CN)
uid (NDS name: uniqueID)

If you create a sort key for an attribute that does not have a defined index, one of the following happens:

- If the control is specified as critical, the function returns “No such attribute”.
- If the control is not marked critical, the control is ignored and the results are returned unsorted.

The `ldap_create_sort_keylist` function allocates memory for the `sortKeyList` array and this memory should be freed by calling the `ldap_free_sort_keylist` function.

For example code, see `sortcntl.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_create_sort_control](#) (page 126), [ldap_free_sort_keylist](#) (page 157)

ldap_create_sstatus_control

Creates and encodes a search status control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>ldap_create_sstatus_control (
    LDAP          *ld,
    LDAPSstatCtrl*sstatctrl,
    int            isCritical,
    LDAPControl    **ctrlp )
```

Parameters

ld

(IN) Points to the handle for the LDAP session obtained from a call to ldap_init().

sstatctrl

(IN) The address of an structure whose contents are used to construct the value of the control that is created.

isCritical

(IN) Indicates the criticality of the control to the operation. 0 indicates that the control is not critical to the operation and a non-zero values indicates that the control is critical to the operation.

ctrlp

(OUT) Returns a pointer to the created LDAPControl. This control is free from calling the ldap_control_free() after returning the pointer.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”

Remarks

The ldap_create_geteffective_control creates a search status control, that can be used as the server control parameter in the ldap_search_ext and the ldap_search_ext_s functions.

ldap_create_vlv_control

Creates and encodes a server-side virtual list view control to use with a search operation.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_create_vlv_control (
    LDAP          *ld,
    LDAPVLVInfo   *vlvinfo,
    LDAPControl    **ctrlp);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

vlvinfo

(IN) Points to an **LDAPVLVInfo** structure that contains the information required to create a virtual list view control.

ctrlp

(OUT) Points to the address of the LDAPControl structure that contains the virtual list view control created by this function. When this control is no longer in use, the memory should be freed by calling the ldap_control_free function.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See " LDAP Return Codes ".

Remarks

The virtual list view control must be used with the server-side sort control. The virtual list view control has been assigned the following OID:

2.16.840.1.113730.3.4.9

For example code, see [vlvctl.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_parse_vlv_control](#) (page 224), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256),
[ldap_create_sort_control](#) (page 126), [ldap_control_free](#) (page 111), [ldap_controls_free](#) (page 112)

[LDAPVLVInfo](#) (page 474), [LDAPControl](#) (page 460)

ldap_delete

Asynchronously deletes the specified entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_delete (
    LDAP      *ld,
    const char *dn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to delete.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

The entry specified for the delete must be a leaf entry. If the entry has children, the delete will fail. LDAP does not support the deletion of a subtree in a single operation.

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

If `ldap_delete` returns -1, check the `LDAP_OPT_RESULT_CODE` option in the LDAP handle for the error code.

See Also

[ldap_delete_s](#) (page 138), [ldap_delete_ext](#) (page 134), [ldap_delete_ext_s](#) (page 136)

ldap_delete_ext

Asynchronously deletes the specified entry using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_delete_ext (
    LDAP      *ld,
    const char *dn,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    int        *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to delete.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with this delete. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with this delete. Use NULL to specify no client controls.

msgidp

(OUT) Points to the integer value to set as the message ID of the request. When the delete request succeeds, use ldap_result with this value to retrieve the response.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x53	LDAP_ENCODING_ERROR

Remarks

The entry specified for the delete must be a leaf entry. If the entry has children, the delete will fail. LDAP does not support the deletion of a subtree in a single operation.

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

eDirectory does not currently support any server-side controls for delete operations.

See Also

[ldap_delete \(page 133\)](#), [ldap_delete_ext_s \(page 136\)](#), [ldap_delete_s \(page 138\)](#)

ldap_delete_ext_s

Synchronously deletes the specified entry using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_delete_ext_s (
    LDAP      *ld,
    const char *dn,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to delete.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with this delete. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with this delete. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The entry specified for the delete must be a leaf entry. If the entry has children, the delete will fail. LDAP does not support the deletion of a subtree in a single operation.

eDirectory does not currently support any server-side controls for delete operations.

For sample code, see [delentry.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_delete](#) (page 133), [ldap_delete_s](#) (page 138), [ldap_delete_ext](#) (page 134)

ldap_delete_s

Synchronously deletes the specified entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_delete_s (
    LDAP      *ld,
    const char *dn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to delete.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The entry specified for the delete must be a leaf entry. If the entry has children, the delete will fail. LDAP does not support the deletion of a subtree in a single operation.

See Also

[ldap_delete](#) (page 133), [ldap_delete_ext_s](#) (page 136), [ldap_delete_ext](#) (page 134)

ldap_destroy

Destroys the session handle.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_destroy (
    LDAP *ld
)
```

Parameters

ld

(IN) Points to the LDAP session handle.

Return Values

0x00	LDAP_SUCCESS
0x59	LDAP_PARAM_ERROR

Remarks

This function destroys the duplicated session handle and should be used in conjunction with [ldap_dup](#).

ldap_dn2ufn

Converts a distinguished name into the user friendly format.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char *ldap_dn2ufn (
    const char *dn);
```

Parameters

dn

(IN) Points to the distinguished name to be converted.

Return Values

>0	Pointer to the converted name
NULL	Failure

Remarks

The user friendly format is defined in RFC 1781. The format strips off the types and places a comma between the components of the name. Components which have commas in their names are placed in quotation marks.

The memory for the user friendly format is newly allocated and should be freed with a call to the ldap_memfree function.

See Also

[ldap_get_dn \(page 159\)](#), [ldap_explode_dn \(page 144\)](#), [ldap_explode_rdn \(page 146\)](#)

ldap_dup

Returns a duplicate of a session handle.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
LDAP * ldap_dup (  
    LDAP *ld  
)
```

Parameters

ld

(IN) Points to the LDAP session handle.

Return Values

Address of the duplicate session handle	Success
-----------------------------------------	---------

Null	Failure
------	---------

ldap_err2string

Converts a numeric LDAP error code into a character string.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char *ldap_err2string (
    int    err);
```

Parameters

err

(IN) Specifies an LDAP error code returned by an LDAP function.

Return Values

>0	Pointer to a zero-terminated character string.
----	------------------------------------------------

Remarks

The ldap_err2string function converts LDAP error codes returned by the following functions:

- ldap_parse_result
- ldap_parse_sasl_bind_result
- ldap_parse_extended_result
- synchronous LDAP operation functions

The LDAP error code is converted to a zero-terminated character string which describes the error.

The return value points to a string contained in static data. Be aware of the following:

- It should be used or copied before another call to ldap_err2string is made.
- The pointer should not be used to modify the original string.
- The string should not be freed by the application program.
- The returned string is UTF-8 encoded if the API succeeds.

If the API succeeds, errno is set to 0. Else, the returned string will be in local codepage.

If the returned string is UTF-8 encoded then it has to be converted into the local codepage before you can print it. Otherwise, the returned pointer can be used directly in a printf statement as displayed in the following example:

```
err=ldap_search(...); if (err) { char *s; s=
ldap_err2string(err); if (errno==0) // returned string is utf8
encoded { //convert to local codepage and print it }
else // returned string is not utf8 encoded, it is in local codepage
printf("Search error: %s\n",s); }
```

For information on converting utf8 to local code page, refer to the utf8bind.c sample code.

NOTE: If the locale is a single byte charset (for example, English), you do not need to convert from UTF-8 to local charset, since UTF-8 charset is the same as the local charset for a single byte charset.

See Also

[ldap_parse_result](#) (page 216), [ldap_parse_extended_result](#) (page 208), [ldap_parse_sasl_bind_result](#) (page 219)

ldap_explode_dn

Breaks a distinguished name into its components.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char **ldap_explode_dn (
    const char    *dn,
    int           notypes);
```

Parameters

dn

(IN) Points to the distinguished name to explode.

notypes

(IN) Specifies whether the name should include type information:

- If zero, type information is included, for example "cn=Kim".
- If non-zero, type information is stripped, for example "cn=Kim" becomes "Kim".

Return Values

>0	Pointer to a character array of components
NULL	Failure

Remarks

The ldap_explode_dn function takes a dn returned by ldap_get_dn and returns a NULL-terminated character array of the components in the name. The components are returned in the order they appear in the dn and are with or without types as indicated by the notypes parameter.

For example, if the dn is "cn=kim,ou=sales,o=myorg", the function returns the following array: {"cn=kim", "ou=sales", "o=myorg", NULL}.

When the array is no longer in use, free the memory by calling the ldap_value_free function.

See Also

[ldap_get_dn](#) (page 159), [ldap_explode_rdn](#) (page 146), [ldap_dn2ufn](#) (page 140)

ldap_explode_rdn

Breaks a relative distinguished name into its components.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char **ldap_explode_rdn (
    const char    *rdn,
    int           notypes);
```

Parameters

rdn

(IN) Points to the relative distinguished name of the entry.

notypes

(IN) Specifies whether the name should include type information:

- If zero, type information is included, for example "cn=Kim".
- If non-zero, type information is stripped, for example "cn=Kim" becomes "Kim".

Return Values

>0	Pointer to a character array of components
NULL	Failure

Remarks

The ldap_explode_rdn returns a NULL-terminated character array with or without types as indicated by the notypes parameter. The components are returned in the order they appear in the rdn.

For example, if the rdn is "ou=sales+cn=kim", the function returns the following array: { "ou=sales", "cn=kim", NULL }.

When the array is no longer in use, free the memory by calling the ldap_value_free function.

See Also

[ldap_get_dn \(page 159\)](#), [ldap_explode_dn \(page 144\)](#), [ldap_dn2ufn \(page 140\)](#)

ldap_extended_operation

Asynchronously passes extended LDAP operations to the LDAP server.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_extended_operation (
    LDAP          *ld,
    const char     *requestoid,
    const struct berval *requestdata,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    int            *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

requestoid

(IN) Points to the dotted-OID text string identifying the extended operation to perform.

requestdata

(IN) Points to the data required for the operation. If NULL, no data is sent to the server.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with this extended operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with this extended operation. Use NULL to specify no client controls.

msgidp

(OUT) Points to the integer value to set as the message ID of the request. When the extended operation succeeds, the results are identified by this value.

Return Values

0	Success
Non-zero	Failure

Remarks

The data returned in the `msgidp` parameter is opaque to the caller. You must use the `ldap_result` and `ldap_parse_extended_result` functions to obtain the result, the OID, and the data.

The LDAP server must support the operation; otherwise an `LDAP_NOT_SUPPORTED` error is returned.

See Also

[ldap_extended_operation_s](#) (page 149)

ldap_extended_operation_s

Synchronously passes extended LDAP operations to the LDAP server.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_extended_operation_s (
    LDAP          *ld,
    const char     *requestoid,
    const struct berval *requestdata,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    char           **retoidp,
    struct berval  **retdatap);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

requestoid

(IN) Points to the dotted-OID text string identifying the operation to perform.

requestdata

(IN) Points to the data required for the operation. If NULL, no data is sent to the server.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with this extended operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with this extended operation. Use NULL to specify no client controls.

retoidp

(OUT) Points to a dotted-OID text string returned by the LDAP server. A NULL value means an OID is not returned. The memory used by the string should be freed with the ldap_memfree function.

retdatap

(OUT) Points to a pointer to a berval structure that contains the returned data. If no data is returned, the server set this to NULL. The memory used by this structure should be freed with the `ber_bvfree` function.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

See Also

[ldap_extended_operation](#) (page 147)

ldap_first_attribute

Returns the name of the first attribute in an entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char *ldap_first_attribute (
    LDAP      *ld,
    LDAPMessage *entry,
    BerElement **ptr);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry whose attributes are being read.

ptr

(OUT) Returns a pointer to a BerElement allocated by the library. It is used internally to track the current position in the entry. This returned value is passed in subsequent calls to the ldap_next_attribute function. It should be freed by the application with a call to the ber_free(ptr, 0) function.

Return Values

NULL	No more attributes or failure
>0	Pointer to the name of the first attribute in an entry

Remarks

The ldap_first_attribute function returns a pointer to the first attribute of an entry returned by either the ldap_first_entry or the ldap_next_entry function.

If NULL is returned and the ptr parameter is not NULL, check the LDAP_OPT_RESULT_CODE option in the LDAP handle for the error code.

If NULL is returned and the ptr parameter is not NULL, all attributes have been retrieved.

The pointer to the name of the first attribute should be passed to the `ldap_get_values` function (or others of its type) to retrieve the attribute's values. When you are done with the name pointer, you must free it by calling the `ldap_memfree` function.

The `ptr` parameter should be used in subsequent calls to the `ldap_next_attribute` function to retrieve other attributes of the entry. When you are done with the `BerElement` structure and its value is non-NULL, you must free it by calling the `ber_free` function with the second parameter set to 0. If the `ptr` parameter is set to NULL, then the `ldap_first_attribute` function frees the memory.

See Also

[ldap_next_attribute \(page 199\)](#), [ldap_get_values \(page 167\)](#)

ldap_first_entry

Returns a pointer to the first entry of message type, LDAP_RES_SEARCH_ENTRY, from a search result chain.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_first_entry (
    LDAP      *ld,
    LDAPMessage *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result chain returned by the ldap_result function or a synchronous search function.

Return Values

NULL	No more entries in the chain or failure
>0	Pointer to the next entry in the chain

Remarks

The ldap_first_entry function parses the results received from the ldap_result, the ldap_search_s, the ldap_search_ext_s, or the ldap_search_st functions.

If the ldap_first_entry function encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

Use the ldap_get_dn, ldap_first_attribute, ldap_get_values functions to retrieve information about the entry.

Use the value returned by the ldap_first_entry function as the entry parameter for the ldap_next_entry function to retrieve the next entry.

See Also

[ldap_next_entry](#) (page 201), [ldap_count_entries](#) (page 113), [ldap_search](#) (page 251),
[ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259),
[ldap_search_st](#) (page 261)

ldap_first_message

Returns a pointer to the first message type, LDAP_RES_SEARCH_ENTRY, LDAP_RES_SEARCH_RESULT, or LDAP_RES_SEARCH_REFERENCE in a result chain.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_first_message (
    LDAP          *ld,
    LDAPMessage    *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result chain returned by the ldap_result function or a synchronous search function.

Return Values

NULL	No more messages or failure
>0	Pointer to a message

Remarks

If ldap_first_message encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

Use the ldap_count_messages function to determine the number of messages in the chain. Use the ldap_next_message function to retrieve subsequent messages.

See Also

[ldap_next_message](#) (page 203), [ldap_count_messages](#) (page 115), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_first_reference

Returns a pointer to the first reference of message type, LDAP_RES_SEARCH_REFERENCE, in a search result chain.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_first_reference (
    LDAP      *ld,
    LDAPMessage *res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to the result chain returned by the ldap_result function or a synchronous search function.

Return Values

NULL	No more references in the chain or failure
>0	Pointer to the next reference in the chain.

Remarks

If the ldap_first_reference function encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

See Also

[ldap_next_reference](#) (page 204), [ldap_count_references](#) (page 117), [ldap_parse_reference](#) (page 212), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_free_sort_keylist

Frees the memory allocated by the ldap_create_sort_keylist function.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_free_sort_keylist (
    LDAPSortKey    **sortKeyList);
```

Parameters

sortKeyList

(IN) Points to an array of pointers to LDAPSortKey structures that you want to free.

Remarks

The ldap_free_sort_keylist function frees the memory used by the LDAPSortKey structures, the memory referenced by the structures, and the array of pointers to the structures. The ldap_free_sort_keylist function should be called only if the memory was allocated by the ldap_create_sort_keylist function.

See Also

[ldap_create_sort_keylist \(page 128\)](#)

ldap_free_urldesc

Frees the memory allocated by the ldap_url_parse function.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_free_urldesc (
    const char      *ludp);
```

Parameters

ludp

(IN) Points to the LDAPURLDesc structure that you want to free.

See Also

[ldap_url_parse \(page 282\)](#)

ldap_get_dn

Returns the distinguished name of an entry from a search result chain.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char *ldap_get_dn (
    LDAP      *ld,
    LDAPMessage *entry);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry returned by the `ldap_first_entry` or the `ldap_next_entry` function.

Return Values

>0	Pointer to the distinguished name of the entry
NULL	Failure to parse the name

Remarks

The `ldap_get_dn` function takes an entry returned by either the `ldap_first_entry` or `ldap_next_entry` function and returns a copy of the entry's dn. It returns a pointer to this newly allocated memory. When you are finished with the name, free the memory with a call to the `ldap_memfree` function.

The distinguished name is returned in the UTF-8 string format as described in RFC 2253.

See Also

[ldap_explode_dn](#) (page 144), [ldap_explode_rdn](#) (page 146), [ldap_dn2ufn](#) (page 140)

ldap_get_digest_md5_realms

Allows the application to retrieve the realm values returned by the server in the digest-challenge from the DIGEST-MD5 context created by a call to [ldap_bind_digest_md5_start](#) (page 91).

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_get_digest_md5_realms (
    LDAP_DIGEST_MD5_CONTEXT    *digestMD5ctx,
    char***                    *realms
);
```

Parameters

digestMD5ctx

(IN) The DIGEST-MD5 context created by a call to [ldap_bind_digest_md5_begin_s](#).

realms

(IN) A pointer to an array of char pointers. This argument will be set to point to an array of char pointers that point to the realm values. The end of the array is indicated by a NULL element value.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see " LDAP Return Codes ".

Remarks

This function allocates memory for the realms array. This memory is freed by calling [ldap_bind_digest_md5_finish](#). The application should NOT attempt to free this memory directly. Multiple calls to the [ldap_get_digest_md5_realms](#) function using the same digest-md5 context will return a pointer to the same array allocated by the first call. This function must not be called after a call to [ldap_bind_digest_md5_finish](#) for the same digest-md5 context.

See Also

[ldap_bind_digest_md5_start](#) (page 91), [ldap_bind_digest_md5_finish](#) (page 93)

ldapssl_install_routines

Enables an existing, but new, LDAP session handle for SSL.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_install_routines (
    LDAP      *ld);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

Return Values

0	Success
-1	Failure

Remarks

To use this function, you must call the following LDAP function in the specified order:

- `ldapssl_client_init` which initializes the SSL library
- `ldap_init` which creates the session handle
- `ldapssl_install_routines` which enables the session handle for SSL

Behavior is unpredictable when other LDAP functions are called between the `ldap_init` function and the `ldapssl_install_routines` function.

The preferred method is to use the `ldapssl_init` function.

See Also

[ldapssl_client_init](#) (page 294), [ldapssl_init](#) (page 297), [ldap_init](#) (page 174)

ldap_get_entry_controls

Retrieves LDAP controls from an entry.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_get_entry_controls (
    LDAP          *ld,
    LDAPMessage    *entry,
    LDAPControl ***serverctrlsp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry, returned by the ldap_first_entry or the ldap_next_entry function, from which to extract controls.

serverctrlsp

(OUT) Points to an array of LDAPControl structures copied out of the entry. If this parameter is set to NULL, no controls are returned.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

Each LDAPControl structure represents one LDAP v3 server control. When the array of LDAPControl structures is no longer in use, free the memory by calling the ldap_controls_free function.

See Also

[ldap_control_free](#) (page 111), [ldap_controls_free](#) (page 112)

ldap_get_lderrno

Returns error information about the last LDAP operation. This function has been deprecated; use the [ldap_get_option](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_get_lderrno (
    LDAP      *ld,
    char      **matchedDN,
    char      **errmsg);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

matchedDN

(OUT) Points to the name of the lowest entry in the directory that was matched on the operation.

errmsg

(OUT) Points to a text string that is optionally returned by the server. This string includes additional details about the error and is not the standard string associated with the error code. Applications should not depend on format of this parameter or assume that it contains data.

Return Values

Returns the LDAP error code from the last operation. Use the `ldap_err2string` function to get the text string associated with this error code.

Remarks

The pointers returned in the function point directly into the LDAP structure.

NOTE: The application should not free these pointers. The pointers must not be used after another LDAP operation has been called. The pointers should not be used to modify the data.

The application should examine or copy the strings before calling another LDAP function.

The pointers are set after every LDAP operation which returns or parses an LDAP result message.

If information is not needed for either the matchedDN or the errmsg parameter, the parameter can be set to NULL.

This is not a standard IETF function. It has been added for compatibility with other LDAP vendors' libraries and should not be used in new applications. Use the `ldap_get_option` function with `LDAP_OPT_ERR_NUMBER`, `LDAP_OPT_MATCHED_DN`, and `LDAP_OPT_ERROR_STRING`.

See Also

[ldap_set_lderrno](#) (page 264), [ldap_get_option](#) (page 166)

ldap_get_option

Returns information about session preferences.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int    ldap_get_option (
    LDAP    *ld,
    int      option,
    void     *outvalue);
```

Parameters

ld

(IN) Contains the session handle. If this is NULL, the function returns information about global defaults.

option

(IN) Contains the name of the option for which information is returned (see [Section 6.10, “Session Preference Options,” on page 413](#)).

outvalue

(OUT) Returns a pointer to a buffer that contains the information about the specified option.

Return Values

0x00	LDAP_SUCCESS
-1	Failure

Remarks

The type of buffer pointed to by the outvalue parameter depends on the option requested. For details, see [Section 6.10, “Session Preference Options,” on page 413](#).

See Also

[ldap_set_option \(page 266\)](#)

ldap_get_values

Returns string values of a specified attribute from an entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char **ldap_get_values (
    LDAP      *ld,
    LDAPMessage *entry,
    const char *attr);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry returned by the ldap_first_entry or the ldap_next_entry function.

attr

(IN) Points to the attribute returned by the ldap_first_attribute function, the ldap_next_attribute function, or the name of an attribute in string format.

Return Values

>0	An array of attribute values
NULL	Failure or no values were found for the attribute

Remarks

The ldap_get_values function takes an entry and attribute and returns a NULL-terminated array of attribute string values. The memory for the array is dynamically allocated. When you are done with the array, free the memory by calling the ldap_value_free function.

The ldap_get_values function can be used to return only character string values. For binary data, use the ldap_get_values_len function.

See Also

[ldap_get_values_len](#) (page 169), [ldap_count_values](#) (page 119), [ldap_count_values_len](#) (page 120), [ldap_value_free](#) (page 292), [ldap_value_free_len](#) (page 293)

ldap_get_values_len

Returns binary values of a specified attribute from an entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

struct berval **ldap_get_values_len (
    LDAP      *ld,
    LDAPMessage *entry,
    const char *attr);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry returned by the ldap_first_entry or the ldap_next_entry function.

attr

(IN) Points to the attribute returned by the ldap_first_attribute function, the ldap_next_attribute function, or the name of an attribute in string format.

Return Values

>0	An array of values
Null	Failure or no values were found for the attribute

Remarks

The ldap_get_values_len function takes an entry and attribute and returns the attribute values in a NULL-terminated array of pointers to berval structures. The memory for the array is dynamically allocated. When you are done with the array, free the memory by calling the ldap_value_free_len function.

See Also

[ldap_get_values](#) (page 167), [ldap_count_values](#) (page 119), [ldap_count_values_len](#) (page 120), [ldap_value_free](#) (page 292), [ldap_value_free_len](#) (page 293)

ldap_gssbind

Authenticates the specified client to the LDAP server using the SASL-GSSAPI mechanism.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.8 or higher

Platform: Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_gss.h>

int ldap_gssbind (
    LDAP      *ld,
    const char *host,
    char       *mechanism,
    const char *dn,
    const char *passwd,
    gss_err_code *err_code);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

host

(IN) Contains the names of the available hosts, each separated by a space, or a list of IP addresses (in dot format) of the hosts, each separated by a space. If a port number is included with the name or the address, it is separated from them with a colon (:).

mechanism

(IN) Supported mechanism. Set this parameter to GSSAPI (for Kerberos V5.)

dn

(IN) Points to the distinguished name of the entry that is authenticating. Set this parameter to NULL if GSSAPI (Kerberos V5) is used as an input mechanism.

passwd

(IN) Points to the client's password. Set this parameter to NULL if GSSAPI (Kerberos V5) is used as input mechanism.

err_code

(OUT) Points to the requested attribute names and values.

Return Values

LDAP_GSS_ERROR 0x62

LDAP_GSS_SECURITY_ERROR 0x63

LDAP_GSS_IMPORT_ERROR 0x64

See Also

[ldap_gss_error \(page 173\)](#)

ldap_gss_error

Converts GSSAPI error into a character string.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.8 or higher

Platform: Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_gss.h>

char * ldap_gss_error (
    gss_err_code      *err);
```

Parameters

err

(IN) Points to the GSS error code structure.

Return Values

>0	Pointer to a zero-terminated character string.
----	------------------------------------------------

See Also

[ldap_gssbind \(page 171\)](#)

ldap_init

Initializes an LDAP session associated with an LDAP server and returns a pointer to a session handle.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAP *ldap_init (
    const char    *host,
    int           port);
```

Parameters

host

(IN) Contains the names of the available hosts, each separated by a space, or a list of IP addresses (in dot format) of the hosts, each separated by a space. If a port number is included with the name or the address, it is separated from them with a colon (:).

port

(IN) Contains the TCP port number to connect to. If a port number is included with the host parameter, this parameter is ignored.

Return Values

>0	Success; session handle
NULL	Unsuccessful

Remarks

If you connect to an LDAP v2 server, you must call an LDAP bind operation before performing any operations. If you connect to an LDAP v3 server, some operations can be performed before calling a bind operation.

The ldap_init function does not actually communicate with the LDAP server. Communication begins when the application binds or does some other operation.

The LDAP libraries first contact the first server listed in the host parameter. If they are unable to communicate with that server, they try the next server and then the next.

The session handle returned contains opaque data identifying the session. To get or set handle information, use `ldap_set_option` and `ldap_get_option`. For a list of the handle options, see [Section 6.10, “Session Preference Options,” on page 413](#).

IMPORTANT: The `ldap_init` function allocates memory for the LDAP structure. This memory must be freed by calling `ldap_unbind` or `ldap_unbind_s` even when an LDAP bind function is not called or the LDAP bind function fails.

See Also

[ldap_get_option \(page 166\)](#), [ldap_set_option \(page 266\)](#), [ldap_open \(page 205\)](#)

ldap_is_ldap_url

Determines whether the URL is an LDAP URL.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_is_ldap_url (
    const char    *url);
```

Parameters

url

(IN) Points to the URL that you want to check.

Return Values

1	URL is an LDAP URL
0	URL is not an LDAP URL

Remarks

An LDAP URL has the protocol set to ldap:// for simple authentication.

See Also

[ldap_url_parse](#) (page 282), [ldap_is_ldaps_url](#) (page 177)

ldap_is_ldaps_url

Determines whether the URL is an LDAPS URL.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_is_ldaps_url (
    const char    *url);
```

Parameters

url

(IN) Points to the URL that you want to check.

Return Values

1	URL is an LDAPS URL
0	URL is not an LDAPS URL

Remarks

An LDAPS URL has the protocol set to ldaps:// for an encrypted SSL connection.

See Also

[ldap_url_parse](#) (page 282), [ldap_is_ldap_url](#) (page 176)

ldap_memfree

Frees memory allocated by a call to the LDAP libraries.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_memfree (
    char *mem);
```

Parameters

mem

(IN) Points to the memory to free. If this argument is NULL, the function does nothing.

Remarks

The ldap_memfree function is used to free memory the LDAP libraries allocated for names on calls to the ldap_first_attribute, ldap_next_attribute, and ldap_get_dn functions.

See Also

[ldap_first_attribute \(page 151\)](#), [ldap_next_attribute \(page 199\)](#), [ldap_get_dn](#)

ldap_modify

Asynchronously modifies the specified entry on the LDAP server.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modify (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **mods);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

mods

(IN) Points to a NULL-terminated array of pointers to the modifications to make to the entry. Each LDAPMod structure contains the modifications for one attribute.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

To obtain the results of the operation, call the ldap_result function with the returned message ID.

If the ldap_modify function returns -1, check the LDAP_OPT_RESULT_CODE option in the LDAP handle for the error code.

To free the memory used by the LDAPMod structures, call the ldap_mods_free function.

Use the ldap_rename or ldap_rename_s function to modify the entry's name.

See Also

[ldap_modify_s](#) (page 185), [ldap_modify_ext](#) (page 181), [ldap_modify_ext_s](#) (page 183),
[ldap_rename](#) (page 227), [ldap_rename_s](#) (page 229)

ldap_modify_ext

Asynchronously modifies specified attributes of an entry on an LDAP server, using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modify_ext (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **mods,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    int        *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

mods

(IN) Points to a NULL-terminated array of pointers to the modifications to make to the entry. Each LDAPMod structure contains the modifications for one attribute.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the modify operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the modify operation. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the search request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

If successful, the message ID of the operation is placed in the msgidp parameter. To obtain the results of the operation, call the ldap_result function using the message ID returned in the msgidp parameter.

Use the ldap_rename or ldap_rename_s function to modify the entry's name.

eDirectory does not currently support any server-side controls to use with modify operations.

See Also

[ldap_modify](#) (page 179), [ldap_modify_s](#) (page 185), [ldap_modify_ext_s](#) (page 183)

ldap_modify_ext_s

Synchronously modifies specified attributes of an entry on an LDAP server, using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modify_ext_s (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **mods,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

mods

(IN) Points to a NULL-terminated array of pointers to the modifications to make to the entry. Each LDAPMod structure contains the modifications for one attribute.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the modify operation. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the modify operation. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.

0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

Use the `ldap_rename` or `ldap_rename_s` function to modify the entry's name.

eDirectory does not currently support any server-side controls to use with modify operations.

For sample code, see `modattrs.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_modify](#) (page 179), [ldap_modify_s](#) (page 185), [ldap_modify_ext](#) (page 181)

ldap_modify_s

Synchronously modifies the specified entry on an LDAP server.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modify_s (
    LDAP      *ld,
    const char *dn,
    LDAPMod   **mods);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

mods

(IN) Points to a NULL-terminated array of pointers to the modifications to make to the entry. Each LDAPMod structure contains the modifications for one attribute.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

Use the ldap_rename or ldap_rename_s function to modify the entry's name.

See Also

[ldap_modify](#) (page 179), [ldap_modify_ext](#) (page 181), [ldap_modify_ext_s](#) (page 183)

ldap_modrdn

Asynchronously modifies the relative distinguished name of a specified entry. This function has been deprecated; use the [ldap_rename](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modrdn (
    LDAP      *ld,
    const char *dn,
    const char *newrdn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

newrdn

(IN) Points to the new relative distinguished name for the entry. The entry's parent must remain the same.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

The `ldap_modrdn` function replaces the old rdn with the value of the new rdn.

The `ldap_modrdn` function has been replaced by the `ldap_rename` function. Unless you need this older function for backwards compatibility, use the newer `ldap_rename` function.

See Also

[ldap_rename](#) (page 227), [ldap_rename_s](#) (page 229)

ldap_modrdn_s

Synchronously modifies the relative distinguished name of a specified entry. This function has been deprecated; use the [ldap_rename_s](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modrdn_s (
    LDAP      *ld,
    const char *dn,
    const char *newrdn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

newrdn

(IN) Points to the new relative distinguished name for the entry. The entry's parent must remain the same.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see " LDAP Return Codes ".

Remarks

The ldap_modrdn_s function replaces the old rdn with the value of the new rdn.

The ldap_modrdn_s function has been replaced by the ldap_rename function. Unless you need this older function for backwards compatibility, use the newer ldap_rename_s function.

See Also

[ldap_rename](#) (page 227), [ldap_rename_s](#) (page 229)

ldap_modrdn2

Asynchronously modifies the relative distinguished name of the specified entry. This function has been deprecated; use the `ldap_rename` function.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modrdn2 (
    LDAP      *ld,
    const char *dn,
    const char *newrdn,
    int        deleteoldrdn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

newrdn

(IN) Points to the new relative distinguished name for the entry. The entry's parent must remain the same.

deleteoldrdn

(IN) Specifies whether the old RDN should be retained or deleted.

- Zero indicates that the old RDN should be retained. If you choose this option, the attribute will contain both names (the old and the new).
- Non-zero indicates that the old RDN should be deleted.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

The `ldap_modrdn2` function has been replaced by the `ldap_rename` function. Unless you need this older function for backwards compatibility, use the newer `ldap_rename` function.

See Also

[ldap_rename](#) (page 227), [ldap_rename_s](#) (page 229)

ldap_modrdn2_s

Synchronously modifies the relative distinguished name of the specified entry. This function has been deprecated; use the [ldap_rename_s](#) function.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_modrdn2_s (
    LDAP      *ld,
    const char *dn,
    const char *newrdn,
    int        deleteoldrdn);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry to modify.

newrdn

(IN) Points to the new relative distinguished name for the entry. The entry's parent must remain the same.

deleteoldrdn

(IN) Specifies whether the old RDN should be retained or deleted.

- Zero indicates that the old RDN should be retained. If you choose this option, the attribute will contain both names (the old and the new).
- Non-zero indicates that the old RDN should be deleted.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see " LDAP Return Codes ".

Remarks

The `ldap_modrdn2_s` function has been replaced by the `ldap_rename` function. Unless you need this older function for backwards compatibility, use the newer `ldap_rename_s` function.

See Also

[ldap_rename](#) (page 227), [ldap_rename_s](#) (page 229)

ldap_msgfree

Frees each message in the result chain pointed to by the res parameter and returns the type of the last message in the chain.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_msgfree (
    LDAPMessage *res);
```

Parameters

res

(IN) Points to the message chain to free. If res is set to NULL, nothing is done.

Return Values

>0x60	Success
0x00	Nothing was done.

Remarks

The ldap_msgfree function is used to free the memory allocated by the ldap_result, ldap_search_s, ldap_search_st, and ldap_search_ext_s functions and returns the type of the last message in the chain.

For a list of possible message types returned by this function, see [Section 6.9, “Result Message Types,”](#) on page 412.

See Also

[ldap_msgid](#) (page 195), [ldap_msgtype](#) (page 196)

ldap_msgid

Returns the message ID associated with the res parameter.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_msgid (
    LDAPMessage *res);
```

Parameters

res

(IN) Points to a message chain returned by the ldap_result, ldap_search_s, ldap_search_st, or ldap_search_ext_s function.

Return Values

>0	The message ID
-1	Failure

See Also

[ldap_msgfree \(page 194\)](#), [ldap_msgtype \(page 196\)](#)

ldap_msgtype

Returns the type of message associated with the res parameter.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_msgtype (
    LDAPMessage *res);
```

Parameters

res

(IN) Points to a message chain returned by the ldap_result, ldap_search_s, ldap_search_st, or ldap_search_ext_s function.

Return Values

>0x60	Type of message
-1	Failure

Remarks

For a list of possible types, see [Section 6.9, “Result Message Types,”](#) on page 412.

See Also

[ldap_msgfree](#) (page 194), [ldap_msgid](#) (page 195)

ldap_multisort_entries

Sorts a chain of entries, returned by an LDAP search operation, using either the entries' DN or a specified array of attributes.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_multisort_entries (
    LDAPMessage    *ld
    LDAPMessage    **res
    char            **attrs
    int (*cmp)      (const void *, const void *));
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

res

(IN) Points to a message chain returned by the `ldap_result`, `ldap_search_s`, `ldap_search_st`, or `ldap_search_ext_s` function.

attrs

(IN) Points to the array of attributes to use for sorting. Pass in NULL to sort by distinguished name.

cmp

(IN) Points to a function to use for sorting. This function returns an int and has two void pointers for parameters.

Return Values

0	Success
-1	Failure

Remarks

If the function returns failure, use [ldap_get_option \(page 166\)](#) to check the LDAP_OPT_RESULT_CODE option in the LDAP handle for the error code.

The sorting order is not well defined when attributes have multiple values. The number of values and the order in which they are received affect the sorting order. For consistent results, use this function with attributes containing single values.

See Also

[ldap_result](#) (page 231), [ldap_sort_entries](#) (page 274), [ldap_search_s](#) (page 259), [ldap_sort_strcasecmp](#) (page 276).

ldap_next_attribute

Returns the name of the next attribute in an entry.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char *ldap_next_attribute (
    LDAP      *ld,
    LDAPMessage *entry,
    BerElement *ptr);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN) Points to the entry whose attributes are being read.

ptr

(IN/OUT) Points to a value used internally to track the current position in the entry. For the first call, use the value returned by the `ldap_first_attribute` function. In subsequent calls, use the value returned by the previous `ldap_next_attribute` call. When the application is done with the `ptr`, it should free the `BerElement` by calling the `ber_free (ptr, 0)` function.

Return Values

NULL	No more attributes or failure
>0	Pointer to the name of the next attribute

Remarks

The `ldap_next_attribute` function returns a pointer to the next attribute of an entry returned by either the `ldap_first_entry` or the `ldap_next_entry` function.

If NULL is returned and the `ptr` parameter is not NULL, check the `LDAP_OPT_RESULT_CODE` option in the LDAP handle for the error code.

If NULL is returned and the `ptr` parameter is NULL, all attributes have been retrieved.

The pointer to the name of the attribute should be passed to the `ldap_get_values` function (or others of its type) to retrieve the attribute's values. When you are done with this pointer, you must free it by calling the `ldap_memfree` function.

The `ptr` parameter should be used in subsequent calls to the `ldap_next_attribute` function to retrieve other attributes of the entry. When you are done with the `BerElement` structure and its value is non-NULL, you must free it by calling the `ber_free` function with the second parameter set to 0. If the `ptr` parameter is set to NULL, then the `ldap_next_attribute` function frees the memory.

See Also

[ldap_first_attribute](#) (page 151), [ldap_get_values](#) (page 167)

ldap_next_entry

Returns a pointer to the next entry of message type, LDAP_RES_SEARCH_ENTRY, in chain of LDAPMessage structures.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_next_entry (
    LDAP          *ld,
    LDAPMessage    *entry);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

entry

(IN/OUT) Points to the next LDAPMessage structure in the chain. On the first call, this is the value returned by the ldap_first_entry function. On subsequent calls, it is the value returned by the ldap_next_entry function.

Return Values

NULL	No more entries in the chain or failure
>0	Pointer to the next entry in the chain

Remarks

If the ldap_next_entry function encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

Use the ldap_get_dn, ldap_first_attribute, ldap_get_values functions to retrieve information about the entry.

See Also

[ldap_first_entry](#) (page 153), [ldap_count_entries](#) (page 113), [ldap_search](#) (page 251),
[ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259),
[ldap_search_st](#) (page 261)

ldap_next_message

Returns a pointer to the next message of message type, LDAP_RES_SEARCH_ENTRY, LDAP_RES_SEARCH_RESULT, or LDAP_RES_SEARCH_REFERENCE, in chain of LDAPMessage structures.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_next_message (
    LDAP      *ld,
    LDAPMessage *msg);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msg

(IN/OUT) Points to the LDAPMessage structure returned by a previous call. On the first call, this is the value returned by the ldap_first_message function. On subsequent calls, it is the value returned by the ldap_next_message function.

Return Values

NULL	No more messages or failure.
>0	Pointer to the next message in the chain

Remarks

If the ldap_next_message function encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

See Also

[ldap_first_message](#) (page 155), [ldap_count_messages](#) (page 115), [ldap_msgid](#) (page 195), [ldap_msgtype](#) (page 196), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_next_reference

Returns a pointer to the next reference of message type, LDAP_RES_SEARCH_REFERENCE, in chain of LDAPMessage structures.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAPMessage *ldap_next_reference (
    LDAP      *ld,
    LDAPMessage *ref);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

ref

(IN/OUT) Points to the next LDAPMessage structure in the search result chain. On the first call, this is the value returned by the ldap_first_reference function. On subsequent calls, it is the value returned by the ldap_next_reference function.

Return Values

NULL	No more references in the chain or failure
>0	Pointer to the next reference in the chain.

Remarks

If the ldap_next_reference function encounters an error, the function returns NULL and sets the LDAP_OPT_RESULT_CODE option in the LDAP session handle.

See Also

[ldap_first_reference](#) (page 156), [ldap_count_references](#) (page 117), [ldap_parse_reference](#) (page 212), [ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261)

ldap_open

Initializes the LDAP library and opens a connection to the LDAP server. This function has been deprecated; use the [ldap_init](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

LDAP *ldap_open (
    const char *hostname,
    int portno);
```

Parameters

hostname

(IN) Contains the names of the available hosts, each separated by a space, or a list of IP addresses (in dot format) of the hosts, each separated by a space. If a port number is included with the name or the address, it is separated from them with a colon (:), for example hostname:port.

portno

(IN) Contains the TCP port number to connect to. If a port number is included with the hostname parameter, this parameter is ignored.

Return Values

>0	Pointer to a session handle
NULL	Failure to establish a session

Remarks

The port number assigned to LDAP is 389.

If the connection is established to an LDAP v2 server, an LDAP bind function must be called before any other operations can be performed.

See Also

[ldap_init](#) (page 174)

ldap_parse_entrychange_control

Decodes the information returned from a search operation that used a persistent search control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_entrychange_control (
    LDAP          *ld,
    LDAPControl   **ctrls,
    int           *changeType,
    char          **prevDN,
    int           *hasChangeNum,
    long          *changeNum) ;
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

ctrls

(IN) A pointer to an array of pointers to controls returned by the server. The controls are obtained by calling the ldap_get_entry_controls() function on an entry returned by the server.

changeType

(OUT) A pointer to an integer specifying the type of change made to the entry. Valid flags are as follows:

LDAP_CHANGETYPE_ADD	Specifies that the entry was added to the directory.
LDAP_CHANGETYPE_DELETE	Specifies that the entry was deleted from the directory.
LDAP_CHANGETYPE_MODIFY	Specifies that the entry was modified.
LDAP_CHANGETYPE_MODDN	Specifies that the DN or RDN of the entry was changed (a modify RDN or modify DN operation was performed).

prevDN

(OUT) A pointer to the previous DN of the entry, if the changetypes argument is LDAP_CHANGETYPE_MODDN. (If the changetypes argument has a different value, this argument is set to NULL.)

When done, you should free this by calling the ldap_memfree function. This parameter is optional and can be set to NULL.

hasChangeNum

(OUT) A pointer to an integer specifying whether or not the change number is included in the control. A non-zero value indicates that the change number is included and is available as the changeNum argument. Zero indicates that the change number is not included. This parameter and the changeNum parameter must either both be NULL or both be non-NULL.

changeNum

(OUT) A pointer to the change number identifying the change made to the entry. This parameter and the hasChangeNum parameter must either both be NULL or both be non-NULL. Change numbers are typically only returned by servers that support a change log.

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_DECODING_ERROR
0x5A	LDAP_NO_MEMORY
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

The ldap_parse_entrychange_control function examines the controls returned with an entry as a result of a persistent search operation. If an entry change control is present, the control is parsed and its elements' values are retrieved. This function should be called after an entry is returned to the client as a result of a persistent search operation. An entry's controls are retrieved by calling the ldap_get_entry_controls function.

For example code, see [searchPersist.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_search_ext](#) (page 253), [ldap_get_entry_controls](#) (page 162),
[ldap_create_persistentsearch_control](#) (page 123)

ldap_parse_extended_result

Retrieves data from an LDAPMessage that contains data from an extended operation.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_extended_result (
    LDAP          *ld,
    LDAPMessage    *res,
    char           **retoidp,
    struct berval  **retdatap,
    int            freeit);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to an LDAPMessage containing the results of an LDAP extended operation.

retoidp

(OUT) Points to the dotted-OID text string that represents the name of the extended operation. Pass in NULL to ignore this field. When you are finished, you must free this string by calling the ldap_memfree function.

retdatap

(OUT) Points to a berval structure that contains data from the extended operation response.

freeit

(IN) Specifies whether the resources allocated by the res parameter are freed.

- Zero indicates that the resources used by the res parameter are not freed automatically. When you are done with the res parameter, you need to call the ldap_msgfree function to free the memory.
- Non-zero indicates that memory is freed after the function extracts the information.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

After calling the `ldap_extended_operation` and `ldap_result` functions, use `ldap_parse_extended_result` to parse the extended information returned by the LDAP server.

See Also

[ldap_err2string](#) (page 142), [ldap_parse_result](#) (page 216), [ldap_parse_sasl_bind_result](#) (page 219)

ldap_parse_intermediate

Retrieves intermediate data from an LDAPMessage that contains data from an extended operation.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_intermediate (
    LDAP          *ld,
    LDAPMessage    *res,
    char          **retoidp,
    struct berval  **retdatap,
    int            freeit);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to an LDAPMessage containing the results of an LDAP extended operation.

retoidp

(OUT) Points to the dotted-OID text string that represents the name of the extended operation. Pass in NULL to ignore this field. When you are finished, you must free this string by calling the ldap_memfree function.

retdatap

(OUT) Points to a berval structure that contains data from the extended operation response.

freeit

(IN) Specifies whether the resources allocated by the res parameter are freed.

- Zero indicates that the resources used by the res parameter are not freed automatically. When you are done with the res parameter, you need to call the ldap_msgfree function to free the memory.
- Non-zero indicates that memory is freed after the function extracts the information.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

After calling the `ldap_extended_operation` and `ldap_result` functions, use `ldap_parse_intermediate` to parse intermediate extended information results returned by the LDAP server.

See Also

[ldap_err2string](#) (page 142), [ldap_parse_result](#) (page 216), [ldap_parse_sasl_bind_result](#) (page 219), [ldap_parse_extended_result](#) (page 208)

ldap_parse_reference

Extracts URLs and controls from an LDAPMessage structure of type LDAP_RES_SEARCH_REFERENCE.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_reference (
    LDAP          *ld,
    LDAPMessage    *ref,
    char          ***referralsp,
    LDAPControl    ***serverctrlsp,
    int            freeit);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

ref

(IN) Points to the reference to parse which was returned by the ldap_result, ldap_first_reference, ldap_next_reference, ldap_first_message, or ldap_next_message function.

referralsp

(OUT) Points to a NULL-terminated array of strings which contains zero or more alternate LDAP server URLs where the request can be sent. Pass in NULL to ignore this parameter. When you are finished, free the referrals array by calling the ldap_value_free function.

serverctrlsp

(OUT) Points to a NULL-terminated array of LDAPControl structures which are returned by the LDAP server and which list the controls the LDAP server supports. When you are finished, free the control array by calling the ldap_controls_free function. Pass in NULL to ignore this parameter.

freeit

(IN) Specifies whether the resources specified by the ref parameter are freed.

- Zero indicates that the resources specified by the res parameter are not freed automatically. When you are done with the LDAPMessage structure, you must call the ldap_msgfree function to free the memory.

- Non-zero indicates that memory is freed by the `ldap_parse_reference` function after it extracts the information.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR

See Also

[ldap_first_reference](#) (page 156), [ldap_next_reference](#) (page 204)

ldap_parse_reference_control

Decodes the information returned from a search operation that used a server-side sort control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>ldap_parse_reference_control (
    LDAP          *ld,
    LDAPControl   **ctrls,
    char          *locRef,
    int           refType,
    char          *remainingName,
    int           scope,
    char          **searchedSubtrees,
    char          *failedName)
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

ctrls

(IN) Points to the address of a null-terminated array of LDAPControl structures obtained by a call to the ldap_parse_result function.

locRef

(OUT) Names the DSE found to hold distributed knowledge information.

refType

(OUT) Indicates the DSE type of ContinuationReference.

remainingName

(OUT) Indicates the new target object if localReference do not completely name the DSE.

searchScope

(OUT) Indicates the search scope of the search operation.

searchedSubtrees

(OUT) Indicates that the search operation has already searched the subtree.

failedName

(OUT) Specifies the non-local names.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”.

ldap_parse_result

Extracts error, referral, and server control information from an LDAPMessage structure.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_result (
    LDAP          *ld,
    LDAPMessage    *res,
    int            *errcodep,
    char           **matcheddn,
    char           **errmsgp,
    char           ***referralsp,
    LDAPControl    ***serverctrlsp,
    int            freeit);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to an LDAPMessage containing the results of an LDAP operation.

errcodep

(OUT) Points to the LDAP error code that specifies the results of the last LDAP operation.

matcheddn

(OUT) Points to a string that specifies how much of the name in the request was recognized. Pass in NULL to ignore this parameter. When you are finished, you must free the matched DN string by calling the ldap_memfree function.

errmsgp

(OUT) Points to the error message string that is associated with the error code. Pass in NULL to ignore this parameter. When you are finished, you must free the error message string by calling the ldap_memfree function.

referralsp

(OUT) Points to a NULL-terminated array of strings which contains zero or more alternate LDAP server URLs where the request can be sent. Pass in NULL to ignore this parameter.

When you are finished, you must free the referrals array by calling the `ldap_value_free` function.

serverctrlsp

(OUT) Points to a NULL-terminated array of LDAPControl structures which are returned by the LDAP server and which list the controls the LDAP server supports. When you are finished, you must free the control array by calling the `ldap_controls_free` function.

freemit

(IN) Specifies whether the resources specified by the `res` parameter are freed.

- Zero indicates that the resources specified by the `res` parameter are not freed automatically. When you are done with the LDAPMessage structure, you must call the `ldap_msgfree` function to free the memory.
- Non-zero indicates that memory is freed by the `ldap_parse_result` function after it extracts the information.

Return Values

0x00	LDAP_SUCCESS
0x5E	LDAP_NO_RESULTS_RETURNED
0x5F	LDAP_MORE_RESULTS_TO_RETURN
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR

Remarks

Upon successful completion, the `ldap_result` function returns the type of the first result returned in the `res` parameter. The type will be one of the following constants:

LDAP_RES_BIND
LDAP_RES_SEARCH_ENTRY
LDAP_RES_SEARCH_REFERENCE
LDAP_RES_SEARCH_RESULT
LDAP_RES_MODIFY
LDAP_RES_ADD
LDAP_RES_DELETE
LDAP_RES_MODDN
LDAP_RES_COMPARE
LDAP_RES_EXTENDED

The `ldap_parse_result` function cannot be used to parse LDAP_RES_SEARCH_ENTRY or LDAP_RES_SEARCH_REFERENCE messages. Use `ldap_first_entry` to parse entries. Use `ldap_parse_reference` to parse references.

If a chain of messages is passed to this function, the function operates only on the first message in the result chain that is not of type `LDAP_RES_SEARCH_ENTRY` or `LDAP_RES_SEARCH_REFERENCE`. Use the `ldap_first_message` and `ldap_next_message` functions to step through a chain of messages.

If the result message contains data from an extended operation, use the `ldap_parse_extended_result` function to retrieve additional information.

See Also

[ldap_err2string](#) (page 142), [ldap_parse_extended_result](#) (page 208), [ldap_parse_sasl_bind_result](#) (page 219)

ldap_parse_sasl_bind_result

Extracts SASL bind information from an LDAPMessage structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

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

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to an LDAPMessage containing the results of an LDAP operation.

servercredp

(OUT) Points to the credentials passed back by the LDAP server to use for mutual authentication. When done with the structure, free the memory by calling the ber_bvfree function.

freeit

(IN) Specifies whether the resources allocated by the res parameter are freed.

- Zero indicates that the resources used by the res parameter are not freed automatically. When you are done with the res parameter, you need to call the ldap_msgfree function to free the memory.
- Non-zero indicates that memory is freed after the function extracts the information.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR

0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

After calling the `ldap_sasl_bind` and the `ldap_result` functions, use the `ldap_parse_sasl_bind_result` to obtain the SASL bind information.

See Also

[ldap_err2string](#) (page 142), [ldap_parse_extended_result](#) (page 208), [ldap_parse_result](#) (page 216)

ldap_parse_sort_control

Decodes the information returned from a search operation that used a server-side sort control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_sort_control (
    LDAP          *ld,
    LDAPControl    **ctrls,
    unsigned long  *returnCode,
    char           **attribute);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

ctrls

(IN) Points to the address of a NULL-terminated array of LDAPControl structures, usually obtained by a call to the ldap_parse_result function.

returnCode

(OUT) Points to the sort control result code. This parameter must not be NULL. See Remarks for a list of possible return codes.

attribute

(OUT) If the sort operation fails, the server may return a string that indicates the first attribute in the sortKey list that caused the failure. If this parameter is NULL, no string is returned. If a string is returned, the memory should be freed by calling the ldap_memfree function.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”.

Remarks

The returnCode parameter returns one of the following values.

Return Value	Description
success (0)	Server returned sorted results.
operationsError (1)	Server had an internal failure.
timeLimitExceeded (3)	Server reached the time limit before the sorting was completed.
strongAuthRequired (8)	Server refused to return sorted results over an insecure protocol.
adminLimitExceeded (11)	The results contain too many matching entries for the server to sort.
noSuchAttribute (16)	Server does not recognized an attribute type in the sort key.
inappropriateMatching (18)	Server does not recognized the matching rule in the sort key, or the matching rule is inappropriate for the attribute type.
insufficientAccessRights (50)	Server refused to return sorted results to this client.
busy (51)	Server is too busy to process.
unwillingToPerform (53)	Server is unable to sort the results.
other (80)	An error occurred.

For example code, see [sortcntl.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) and [vlvcntl.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_create_sort_control](#) (page 126)

ldap_parse_sstatus_control

Decodes the information returned from a search status control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>ldap_parse_sstatus_control (
    LDAP          *ld,
    LDAPControl    **ctrls,
    int            *numEax,
    int            *numPass,
    int            *evaDone,
    int            *numAva )
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

ctrls

(IN) Points to the address of a null-terminated array of LDAPControl structures, obtained by a call to the ldap_parse_result function.

numEax

(OUT) Indicates the number of examined records.

numPass

(OUT) Indicates the number of examined records that matchs the search criteria.

evaDone

(OUT) Indicates the evaluation done.

numAva

(OUT) Indicates the number of records available

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”.

ldap_parse_vlv_control

Decodes the information returned from a search operation that used a VLV (virtual list view) control.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 8.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_vlv_control (
    LDAP          *ld
    LDAPControl    **ctrlp,
    unsigned long  *target_posp,
    unsigned long  *list_countp,
    struct berval  **contextp,
    int            *errcodep);
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

ctrlp

(IN) Points to a NULL-terminated array of LDAPControl structures, typically obtained by calling the ldap_parse_result function.

target_posp

(OUT) Points to the list index of the target entry. If this parameter is NULL, the target position is not returned. The index returned is an approximation of the position of the target entry. It is not guaranteed to be exact.

list_countp

(OUT) Points to the server's estimate of the size of the list. If this parameter is NULL, the size is not returned.

contextp

(OUT) Points to the address of a berval structure that contains a server-generated context identifier if server returns one. If server does not return a context identifier, the server returns a NULL in this parameter. If this parameter is set to NULL, the context identifier is not returned.

You should use this returned context in the next call to create a VLV control.

When the berval structure is no longer needed, you should free the memory by calling the ber_bvfree function.

errcodep

(OUT) Points to the result code returned by the server. If this parameter is NULL, the result code is not returned. See Remarks for a list of possible return codes.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”.
0x5D	LDAP_CONTROL_NOT_FOUND

Remarks

The errcodep parameter returns one of the following values.

Return Value	Description
success (0)	Server returned VLV results.
operationsError (1)	Server had an internal failure.
timeLimitExceeded (3)	Server reached the time limit before the virtual list view was completed.
adminLimitExceeded (11)	The results contain too many matching entries for the server to place in a virtual list view.
insufficientAccessRights (50)	Server refused to return sorted results to this client.
busy (51)	Server is too busy to process.
unwillingToPerform (53)	Server is unable to sort the results.
sortControlMissing (60)	The sort control for the virtual list view is missing.
offsetRangeError (61)	The offset is set to less than zero.
other (80)	An error occurred.

For example code, see [vlvctl.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[berval](#) (page 429), [LDAPControl](#) (page 460)

ldap_perror

Prints a specified message and the current LDAP error message to standard error. This function has been deprecated; use the [ldap_err2string](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_perror (
    LDAP      *ld,
    const char *msg);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msg

(IN) Points to the message that is displayed before the LDAP error message.

See Also

[ldap_err2string](#) (page 142), [ldap_parse_result](#) (page 216)

ldap_rename

Asynchronously renames the specified entry.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_rename (
    LDAP          *ld,
    const char    *dn,
    const char    *newrdn,
    const char    *newparent,
    int           deleteoldrdn,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    int           *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry that is being renamed.

newrdn

(IN) Points to the new relative distinguished name to give the entry.

newparent

(IN) Points to the distinguished name of the entry's new parent. If this parameter is NULL, only the RDN is changed. The root DN is specified by passing a zero length string, "".

This function can be used with LDAP v2 servers if the newparent parameter is NULL. LDAP v2 does not allow the entry to be moved to a new parent.

deleteoldrdn

(IN) Specifies whether the old RDN should be retained or deleted.

- Zero indicates that the old RDN should be retained. If you choose this option, the attribute will contain both names (the old and the new).
- Non-zero indicates that the old RDN should be deleted.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the rename. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the rename. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the rename request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

The ldap_rename function changes the leaf component of an entry's distinguished name and optionally moves the entry to a new parent container.

To obtain the results of the operation, call the ldap_result function using the message ID in the msgidp parameter.

eDirectory does not currently support any server-side controls to use with renaming an entry.

See Also

[ldap_rename_s \(page 229\)](#)

ldap_rename_s

Synchronously renames the specified entry.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_rename_s (
    LDAP      *ld,
    const char *dn,
    const char *newrdn,
    const char *newparent,
    int        deleteoldrdn,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry that is being renamed.

newrdn

(IN) Points to the new relative distinguished name to give the entry.

newparent

(IN) Points to the distinguished name of the entry's new parent. If this parameter is NULL, only the RDN is changed. The root DN is specified by passing a zero length string, "".

This function can be used with LDAP v2 servers if the newparent parameter is NULL. LDAP v2 does not allow the entry to be moved to a new parent.

deleteoldrdn

(IN) Specifies whether the old RDN should be retained or deleted.

- Zero indicates that the old RDN should be retained. If you choose this option, the attribute will contain both names (the old and the new).
- Non-zero indicates that the old RDN should be deleted.

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the rename. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the rename. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

eDirectory does not currently support any server-side controls to use with renaming an entry.

For sample code, see [renamerdn.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_rename](#) (page 227)

ldap_result

Obtains results from a previous asynchronously initiated operation.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_result (
    LDAP          *ld,
    int           msgid,
    int           all,
    struct timeval *timeout,
    LDAPMessage   **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgid

(IN) Specifies the message ID returned in the LDAP operation's msgidp parameter. Accepts the following special flags:

- LDAP_RES_UNSOLICITED (0) indicates that an unsolicited result is requested.
- LDAP_RES_ANY (-1) indicates that any result is to be returned.

all

(IN) Specifies how many messages are retrieved in a single call to the ldap_result function. Uses one of the following flags:

- LDAP_MSG_ONE (0x00) indicates that messages are retrieved one at a time.
- LDAP_MSG_ALL (0x01) indicates that all results of the search must be received before returning with all the messages in a single chain.
- LDAP_MSG_RECEIVED (0x02) indicates that all messages received so far must be returned in a result chain.

timeout

(IN) Points to a timeval structure that specifies how long to wait for the results to be returned.

- To block until the results are available, pass a NULL value.
- To cause continuous polling, set the tv_sec field in the timeval structure to zero seconds.

res

(OUT) Points to the results of the search. If no results are returned, this parameter is set to NULL.

Return Values

0x00	Timeout expired
-1	Failure

Remarks

Only asynchronous search operations can contain more than one message.

Upon successful completion, the `ldap_result` function returns the type of message. For a list of possible types, see [Section 6.9, “Result Message Types,” on page 412](#).

If the `ldap_result` function returns a -1, use the `ldap_get_option` function with the option parameter set to `LDAP_OPT_RESULT_CODE` to retrieve the error code from the LDAP session handle.

The `ldap_result` function allocates memory for the `res` parameter. When you are done with it, free the memory by calling the `ldap_msgfree` function.

See Also

[ldap_msgfree \(page 194\)](#), [ldap_msgid \(page 195\)](#), [ldap_msgtype \(page 196\)](#)

ldap_result2error

Converts the result message into a numeric LDAP error code. This function has been deprecated. Use the [ldap_parse_result](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_result2error (
    LDAP          *ld,
    LDAPMessage    *res,
    int            freeit);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

res

(IN) Points to an LDAPMessage containing the results of an `ldap_result` or `ldap_search_s` operation.

freeit

(IN) Specifies whether the resources allocated by the `res` parameter are freed.

- Zero indicates that the resources used by the `res` parameter are not freed automatically. When you are done with the `res` parameter, you need to call the `ldap_msgfree` function to free the memory.
- Non-zero indicates that memory is freed after the function extracts the information

Return Values

>0	LDAP error code. See “ LDAP Return Codes ”.
----	-------------------------------------------------------------

Remarks

The `ldap_result2error` function does the following:

- Converts the result message into a numeric LDAP error code.

- Parses the result message and puts the matched distinguished name in the LDAP_OPT_MATCHED_DN option of the LDAP session handle.
- Parses the result message and puts the error code in the LDAP_OPT_ERROR_STRING option of the LDAP session handle.

All synchronous operation routines call the `ldap_result2error` function before returning, ensuring that the options are set correctly.

See Also

[ldap_parse_result](#) (page 216), [ldap_parse_extended_result](#) (page 208), [ldap_perror](#) (page 226)

ldap_sasl_bind

Asynchronously authenticates the specified client to the LDAP server using a Simple Authentication Security Layer (SASL).

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_sasl_bind (
    LDAP          *ld,
    const char    *dn,
    const char    *mechanism,
    const struct berval *cred,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    int           *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry who is authenticating.

mechanism

(IN) Points to the method to use for authentication, either

- LDAP_SASL_SIMPLE (NULL) for simple authentication
- A dotted-string representation of the OID identifying the SASL method

cred

(IN) Points to the credentials with which to authenticate.

serverctrls

(IN) Points to a list of server controls. Use NULL to specify no server controls.

clientctrls

(IN) Points to a list of client controls. Use NULL to specify no client controls.

msgidp

(OUT) Points to the message ID of the request when the bind request succeeds.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

The `ldap_sasl_bind` function is an asynchronous function and does not return the results directly. To obtain the results, call the `ldap_parse_result` function using the message ID in the `msgidp` parameter.

If you want the function to return the results directly, use the `ldap_sasl_bind_s` function.

If the `ldap_sasl_bind` function returns -1, check the `LDAP_OPT_RESULT_CODE` option in the LDAP handle for the error code.

The `LDAP_OPT_NETWORK_TIMEOUT` option (set by calling [ldap_set_option \(page 266\)](#)) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by spaces in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing NULL for the `ld` parameter of `ldap_set_option` sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init \(page 174\)](#) or [ldapssl_init \(page 297\)](#). To clear the timeout pass NULL for the `invalue` parameter of `ldap_set_option`.

A connection timeout will cause an `LDAP_SERVER_DOWN` error (81) "Can't contact LDAP server".

See Also

[ldap_sasl_bind_s \(page 237\)](#), [ldap_parse_sasl_bind_result \(page 219\)](#)

ldap_sasl_bind_s

Synchronously authenticates the specified client to the LDAP server using a Simple Authentication Security Layer (SASL).

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_sasl_bind_s (
    LDAP          *ld,
    const char     *dn,
    const char     *mechanism,
    const struct berval *cred,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    struct berval  **servercredp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry who is authenticating.

mechanism

(IN) Points to the method to use for authentication, either

- LDAP_SASL_SIMPLE (NULL) for simple authentication
- A dotted-string representation of the OID identifying the SASL method

cred

(IN) Points to the credentials with which to authenticate.

serverctrls

(IN) Points to a list of server controls. Use NULL to specify no server controls.

clientctrls

(IN) Points to a list of client controls. Use NULL to specify no client controls.

servercredp

(OUT) Points to the credentials passed back by the server for mutual authentication. The `berval` structure must be freed by calling the `ber_bvfree` function. To ignore this parameter, set it to `NULL`.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

The `LDAP_OPT_NETWORK_TIMEOUT` option (set by calling [ldap_set_option \(page 266\)](#)) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by spaces in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing `NULL` for the `ld` parameter of `ldap_set_option` sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init \(page 174\)](#) or [ldapssl_init \(page 297\)](#). To clear the timeout pass `NULL` for the `invalue` parameter of `ldap_set_option`.

A connection timeout will cause an `LDAP_SERVER_DOWN` error (81) "Can't contact LDAP server".

See Also

[ldap_sasl_bind \(page 235\)](#)

ldap_schema_fetch

Connects to a directory and returns the schema to an LDAPSchema struct.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_fetch (
    LDAP      ld,
    LDAPSchema **schema,
    const char *subschemaSubentryDN);
```

Parameters

ld

(IN) LDAP session handle.

schema

(OUT) Address of a handle to LDAPSchema, contains a local copy of the entire directory schema.

subschemaSubentryDN

(IN) Distinguished name of the entry from which to return schema.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

A call to ldap_schema_fetch will connect to a directory and locate the SubSchemaSubEntry. It allocates an LDAPSchema structure and populates it with all available schema definitions.

The schema will be read from the subschemaSubentry passed in. If subschemaSubentryDN is NULL then the first subschemaSubentry listed in the root DSE will be used.

NOTE: Setting the SubSchemaSubentryDN to NULL requires version 3 and eDirectory 8.xx.

See Also

[ldap_schema_free](#) (page 240)

ldap_schema_free

Frees the memory allocated to an LDAPSchema handle.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_free (
    LDAPSchema *schema);
```

Parameters

schema

(IN) Handle to a local copy of directory schema.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

For every handle created by `ldap_schema_fetch`, `ldap_schema_free` must be called to free the memory.

See Also

[ldap_schema_fetch](#) (page 239)

ldap_schema_get_by_name

Retrieves a handle to a schema element, identified by its type and either a name or oid.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_get_by_name (
    LDAPSchema          *schema,
    char*               nameOrOid,
    int                 elementType,
    LDAPSchemaElement **element);
```

Parameters

schema

(IN) A handle to the schema of an LDAP directory.

nameOrOid

(IN) Name or oid of the schema element requested.

elementType

(IN) Type of element requested. Use the definitions listed in [Section 6.11, “Schema Element Types,” on page 418](#).

element

(OUT) Address to a handle of the schema element requested. The user must not modify this memory.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

The returned handle to an LDAPSchemaElement structure, 'element', is a pointer to memory within the LDAPSchema structure, 'schema'. Therefore if 'schema' changes or is freed, 'element' may also change, or become invalid. Likewise, if the user frees or tampers with 'element', 'schema' may become corrupted.

ldap_schema_get_count

Returns the count of schema elements of the type specified.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_get_count (
    LDAPSchema      *schema,
    int              elementType);
```

Parameters

schema

(IN) A handle to the schema of an LDAP directory.

elementType

(IN) Type of element requested. Use the definitions listed in [Section 6.11, “Schema Element Types,” on page 418](#).

Return Values

Return value is -1 if the LDAPSchema structure is invalid or the elementType is invalid. Otherwise the return value is the count of schema elements.

Remarks

Ldap_schema_get_count is used to get valid values for the index parameter of [ldap_schema_get_by_index \(page 243\)](#).

See Also

[ldap_schema_get_by_index \(page 243\)](#)

ldap_schema_get_by_index

Allows you to iterate through schema elements of a specific type.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_get_by_index (
    LDAPSchema          *schema,
    int                 index,
    int                 elementType,
    LDAPSchemaElement **element);
```

Parameters

schema

(IN) A handle to the schema of an LDAP directory.

index

(IN) index of the desired schema element. (Uses array numbering; starts at zero.

[ldap_schema_get_count \(page 242\)](#) is used to determine valid indices for this parameter.

elementType

(IN) Type of element requested. Use the definitions listed in [Section 6.11, “Schema Element Types,”](#) on page 418.

element

(OUT) Address to a handle of the schema element requested. The user must not modify this memory.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

The index is zero based and goes through `ldap_schema_get_count - 1`. The returned handle to an `LDAPSchemaElement` structure, 'element', is a pointer to memory within the `LDAPSchema` structure, 'schema'. Therefore if 'schema' changes or is freed, 'element' may also change, or become invalid. Likewise, if the user frees or tampers with 'element', 'schema' may become corrupted.

Remarks

See Also

[ldap_schema_get_count](#) (page 242)

ldap_schema_get_field_names

Retrieves a list of field names in a null-terminated array.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_get_field_names (
    LDAPSchemaElement *element,
    char               *(* fieldnames[]));
```

Parameters

element

(IN) Handle to a Schema element.

fieldNames

(OUT) Address of a null-terminated array of string pointers that contain all field names defined for this schema element. Free this memory with [ldap_value_free \(page 292\)](#).

Return Values

See the “[LDAP Return Codes](#)” for return values.

ldap_schema_get_field_values

Retrieves a list of field names in a null-terminated array.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_get_field_values (
    LDAPSchemaElement *element,
    char *fieldName,
    char * (*values[]));
```

Parameters

element

(IN) Handle to a Schema element.

fieldName

(IN) Name of the field for which values are requested. See [Section 6.11, “Schema Element Types,” on page 418](#).

values

(OUT) Null-terminated array of string pointers containing the values for a field. Free this memory with [ldap_value_free \(page 292\)](#).

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

Valid field names are listed in [Section 6.11, “Schema Element Types,” on page 418](#). Some fields, although valid, may not have values (For example, LDAP_SCHEMA_OBSOLETE.) In this case values will be NULL and the return value will be LDAP_SUCCESS. If the field name does not exist values will be NULL and LDAP_NO_SUCH_ATTRIBUTE is returned.

ldap_schema_add

Adds a schema element definition to the local copy of schema in an LDAPSchema structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_add (
    LDAPSchema      *schema,
    int             type,
    LDAPSchemaMod   *fields[]);
```

Parameters

schema

(IN) A handle to the schema of a directory.

type

(IN) Type of element requested. Use the definitions listed in [Schema Element Types](#).

fields

(IN) An array of pointers to LDAPSchemaMod structures. Each structure represents a field in an attribute definition. ldap_schema_add ignores the 'op' field in this structure.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

ldap_schema_add will construct a new schema element definition from the schema mod structures passed in and add the definition to the LDAPSchema structure passed in. Additions are only made to the LDAPSchema structure. To commit this addition to the directory, call [ldap_schema_save](#) (page 250).

ldap_schema_modify

Modifies an existing schema element definition.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_modify (
    LDAPSchema      *schema,
    char            *nameOrOid,
    int              type,
    LDAPSchemaMod    *fieldsToChange[]);
```

Parameters

schema

(IN) A handle to the schema of a directory.

nameOrOid

(IN) A name or OID that identifies the schema definition to modify.

type

(IN) Type of element to modify. Use the definitions listed in [Section 6.11, “Schema Element Types,” on page 418](#).

fieldsToChange

(IN) An null-terminated array of pointers of LDAPSchemaMod structures. Each structure represents a field in an attribute definition.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

ldap_schema_modify modifies an existing schema element definition. Using an existing definition in schema, this constructs a new definition according the list of fields passed in. Modifications are only made to the LDAPSchema structure. To commit this modification to the directory, call [ldap_schema_save \(page 250\)](#).

A field with an operation code of LDAP_MOD_ADD will add values to a field, creating new fields if one does not already exist. A field with an operation code of LDAP_MOD_REPLACE will replace the existing field values, or creating new values if the field does not exist. A field with an operation of LDAP_MOD_DELETE will remove the field values listed, if they exist.

ldap_schema_delete

Removes a schema element definition from the directory and from the local copy of schema in an LDAPSchema structure.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_delete (
    LDAPSchema      *schema,
    char             *nameOrOid,
    int              type);
```

Parameters

schema

(IN) A handle to the schema of a directory.

nameOrOid

(IN) A name or OID that identifies the schema definition to modify.

type

(IN) Type of element to modify. Use the definitions listed in [Section 6.11, “Schema Element Types,” on page 418](#).

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

ldap_schema_delete removes a schema element definition from the from the local copy of schema in an LDAPSchema structure, 'schema'. Deletions are only made to the LDAPSchema structure. To commit deletions to the directory, call [ldap_schema_save \(page 250\)](#).

ldap_schema_save

Commits any changed made in the LDAPSchema structure since the schema was fetched from a directory.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_schema_save (
    LDAP          *ld,
    LDAPSchema    *schema,
    const char    *subschemaSubentryDN);
```

Parameters

ld

(IN) LDAP session handle.

schema

(IN) A handle to the schema of a directory..

subschemaSubentryDN

(IN) Distinguished name of the entry from which to return schema.

Return Values

See the “[LDAP Return Codes](#)” for return values.

Remarks

The schema changes will be saved to the subschemaSubentry passed in. If subschemaSubentryDN is NULL then the first subschemaSubentry listed in the root DSE will be used.

NOTE: Setting the SubSchemaSubentryDN to NULL requires version 3 and eDirectory 8.xx.

All changes made to the LDAPSchema structure using ldap_schema_add, ldap_schema_modify, and ldap_schema_delete, are sent to the directory as a single transactional request.

ldap_search

Asynchronously searches the directory.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_search (
    LDAP      *ld,
    const char *base,
    int       scope,
    const char *filter,
    char      **attrs,
    int       attrsonly);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

base

(IN) Points to the distinguished name of the entry from which to start the search.

scope

(IN) Specifies the scope of the search and uses one of the following flags:

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

filter

(IN) Points to a search filter.

If NULL is passed, a default filter ("objectclass=*") is used, a filter which matches all entries in the directory. Using a NULL filter is not recommended for subtree searches on trees that potentially have hundreds of thousands of entries.

Simple filters take the form of strings: attribute name=attribute value. For more complex filters, see [“Using Search Filters” on page 37](#).

attrs

(IN) Points to a NULL-terminated array of strings indicating which attributes to return with each matching entry. To return only entry names (and no attributes), set the first, and only string in the array, to LDAP_NO_ATTRS. To return all attributes, set this parameter to NULL.

For example, to return the cn, surname, and givenName attributes, declare attrs as: `char* attrs[]={ "cn", "surname", "givenName", NULL};`

attrsonly

(IN) Specifies whether to return just attributes or attributes and values.

- Zero—return both attributes and values
- Non-zero—return only attributes

Return Values

>0	Message ID of operation
-1	Failure

Remarks

The `ldap_search` function is an older function which does not allow you to specify LDAP controls.

The LDAP_OPT_DEREF option in the LDAP session handle affects how aliases are handled during the search.

- The LDAP_DEREF_FINDING value means aliases are dereferenced when locating the base object but not during the search.
- The LDAP_DEREF_SEARCHING value means aliases are dereferenced during the search but not when locating the base object of the search.

To obtain the results of the operation, call the `ldap_result` function using the message ID returned to the `ldap_search` function.

If the function returns a -1, use the `ldap_get_option` function with the option parameter set to LDAP_OPT_RESULT_CODE to retrieve the error code from the LDAP session handle.

Server timeouts and size limits for this function are set using the LDAP_OPT_TIMELIMIT and LDAP_OPT_SIZELIMIT options on the LDAP handle. This function has no client time or size limits.

See Also

[ldap_search_ext](#) (page 253), [ldap_search_st](#) (page 261)

ldap_search_ext

Asynchronously searches the directory using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_search_ext (
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attrsonly,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls,
    struct timeval *timeout,
    int           sizelimit,
    int           *msgidp);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

base

(IN) Points to the distinguished name of the entry from which to start the search.

scope

(IN) Specifies the scope of the search and uses one of the following flags:

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

filter

(IN) Points to a search filter.

If NULL is passed, a default filter ("objectclass=*") is used, a filter which matches all entries in the directory. Using a NULL filter is not recommended for subtree searches on trees that potentially have hundreds of thousands of entries.

Simple filters take the form of strings: attribute name=attribute value. For more complex filters, see [“Using Search Filters” on page 37](#).

attrs

(IN) Points to a NULL-terminated array of strings indicating which attributes to return with each matching entry. To return only entry names (and no attributes), set the first, and only string in the array, to LDAP_NO_ATTRS. To return all attributes, set this parameter to NULL.

For example, to return the cn, surname, and givenName attributes, declare attrs as: `char* attrs[]={ "cn", "surname", "givenName", NULL};`

attrsonly

(IN) Specifies whether to return just attributes or attributes and values.

- Zero—return both attributes and values
- Non-zero—return only attributes

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the search. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the search. Use NULL to specify no client controls.

timeout

(IN) Points to a timeval structure that specifies the maximum time to wait for the results of a search to complete. It specifies both the time the server waits for the operation to complete as well as the time the local function waits for the server to respond. If the timeout parameter is set to NULL, the client timeout is infinite and the server uses the timeout value stored in the session handle option, LDAP_OPT_TIMELIMIT (whose default value is no timeout). For more information about possible values, see [timeval \(page 476\)](#).

sizelimit

(IN) Specifies the maximum number of entries to return.

- To specify no limit, pass LDAP_NO_LIMIT (0).
- To use the current value in the LDAP session handle (the LDAP_OPT_SIZELIMIT option), pass LDAP_DEFAULT_SIZELIMIT (-1).

msgidp

(OUT) Points to the message ID of the request if the search request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes” .
0x53	LDAP_ENCODING_ERROR
0x55	LDAP_TIMEOUT

Remarks

The LDAP_OPT_DEREF option in the LDAP session handle affects how aliases are handled during the search.

- The LDAP_DEREF_FINDING value means aliases are dereferenced when locating the base object but not during the search.
- The LDAP_DEREF_SEARCHING value means aliases are dereferenced during the search but not when locating the base object of the search.

eDirectory supports two server controls:

- Server-side sorting—1.2.840.113556.1.4.473
- Virtual list views—2.16.840.1.113730.3.4.9

To obtain the results of the operation, call the ldap_result function using the message ID returned to the ldap_search_ext function.

Server timeouts and size limits for this function are set using the LDAP_OPT_TIMELIMIT and LDAP_OPT_SIZELIMIT options on the LDAP handle. Client timeouts and size limits are set using the timeout and sizelimit parameters.

For sample code, see [searchas.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_search](#) (page 251), [ldap_search_st](#) (page 261)

ldap_search_ext_s

Synchronously searches the directory using LDAP client or server controls.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>
```

```
int ldap_search_ext_s (
    LDAP          *ld,
    const char     *base,
    int            scope,
    const char     *filter,
    char           **attrs,
    int            attrsonly,
    LDAPControl    **serverctrls,
    LDAPControl    **clientctrls,
    struct timeval *timeout,
    int            sizelimit,
    LDAPMessage    **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

base

(IN) Points to the distinguished name of the entry from which to start the search.

scope

(IN) Specifies the scope of the search and uses one of the following flags:

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

filter

(IN) Points to a search filter.

If NULL is passed, a default filter ("objectclass=*") is used, a filter which matches all entries in the directory. Using a NULL filter is not recommended for subtree searches on trees that potentially have hundreds of thousands of entries.

Simple filters take the form of strings: attribute name=attribute value. For more complex filters, see [“Using Search Filters” on page 37](#).

attrs

(IN) Points to a NULL-terminated array of strings indicating which attributes to return with each matching entry. To return only entry names (and no attributes), set the first, and only string in the array, to LDAP_NO_ATTRS. To return all attributes, set this parameter to NULL.

For example, to return the cn, surname, and givenName attributes, declare attrs as: `char* attrs[]={ "cn", "surname", "givenName", NULL};`

attrsonly

(IN) Specifies whether to return just attributes or attributes and values.

- Zero—return both attributes and values
- Non-zero—return only attributes

serverctrls

(IN) Points to an array of LDAPControl structures that list the server controls to use with the search. Use NULL to specify no server controls.

clientctrls

(IN) Points to an array of LDAPControl structures that list the client controls to use with the search. Use NULL to specify no client controls.

timeout

(IN) Points to a timeval structure that specifies the maximum time to wait for the results of a search to complete. It specifies both the time the server waits for the operation to complete as well as the time the local function waits for the server to respond. If the timeout parameter is set to NULL, the client timeout is infinite and the server uses the timeout value stored in the session handle option, LDAP_OPT_TIMELIMIT (whose default value is no timeout). For more information about possible values, see [timeval \(page 476\)](#).

sizelimit

(IN) Specifies the maximum number of entries to return.

- To specify no limit, pass LDAP_NO_LIMIT (0).
- To use the current value in the LDAP session handle (the LDAP_OPT_SIZELIMIT option), pass LDAP_DEFAULT_SIZELIMIT (-1).

res

(OUT) Returns a pointer to an array of result messages if the search succeeds or NULL if no results are returned.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes” .
0x53	LDAP_ENCODING_ERROR

0x55	LDAP_TIMEOUT
0x57	LDAP_FILTER_ERROR

Remarks

The LDAP_OPT_DEREF option in the LDAP session handle affects how aliases are handled during the search.

- The LDAP_DEREF_FINDING value means aliases are dereferenced when locating the base object but not during the search.
- The LDAP_DEREF_SEARCHING value means aliases are dereferenced during the search but not when locating the base object of the search.

eDirectory supports two server controls:

- Server-side sorting—1.2.840.113556.1.4.473
- Virtual list views—2.16.840.1.113730.3.4.9

You must use the ldap_result and the ldap_parse_result functions to retrieve the results of the search.

Server timeouts and size limits for this function are set using the LDAP_OPT_TIMELIMIT and LDAP_OPT_SIZELIMIT options on the LDAP handle. Client timeouts and size limits are set using the timeout and sizelimit parameters.

For sample code, see [search.c](#) and [searchmsg.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_search](#) (page 251), [ldap_search_s](#) (page 259), [ldap_search_st](#) (page 261), [ldap_search_ext](#) (page 253)

ldap_search_s

Synchronously searches the directory.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_search_s (
    LDAP          *ld,
    const char    *base,
    int           scope,
    const char    *filter,
    char          **attrs,
    int           attrsonly,
    LDAPMessage   **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

base

(IN) Points to the distinguished name of the entry from which to start the search.

scope

(IN) Specifies the scope of the search and uses one of the following flags:

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

filter

(IN) Points to a search filter.

If NULL is passed, a default filter ("objectclass=*") is used, a filter which matches all entries in the directory. Using a NULL filter is not recommended for subtree searches on trees that potentially have hundreds of thousands of entries.

Simple filters take the form of strings: attribute name=attribute value. For more complex filters, see [“Using Search Filters” on page 37](#).

attrs

(IN) Points to a NULL-terminated array of strings indicating which attributes to return with each matching entry. To return only entry names (and no attributes), set the first, and only string in the array, to LDAP_NO_ATTRS. To return all attributes, set this parameter to NULL.

For example, to return the cn, surname, and givenName attributes, declare attrs as: `char* attrs[]={ "cn", "surname", "givenName", NULL};`

attrsonly

(IN) Specifies whether to return just attributes or attributes and values.

- Zero—return both attributes and values
- Non-zero—return only attributes

res

(OUT) Returns a pointer to an array of result messages if the search succeeds or NULL if no results are returned.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0X57	LDAP_FILTER_ERROR

Remarks

The `ldap_search` function is an older function which does not allow you to specify LDAP controls.

The LDAP_OPT_DEREF option in the LDAP session handle affects how aliases are handled during the search.

- The LDAP_DEREF_FINDING value means aliases are dereferenced when locating the base object but not during the search.
- The LDAP_DEREF_SEARCHING value means aliases are dereferenced during the search but not when locating the base object of the search.

Server timeouts and size limits for this function are set using the LDAP_OPT_TIMELIMIT and LDAP_OPT_SIZELIMIT options on the LDAP handle. This function has no client time or size limits.

See Also

[ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_st](#) (page 261)

ldap_search_st

Synchronously searches the directory within a specified time limit.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_search_st (
    LDAP          *ld,
    const char     *base,
    int            scope,
    const char     *filter,
    char          **attrs,
    int            attrsonly,
    struct timeval *timeout,
    LDAPMessage    **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

base

(IN) Points to the distinguished name of the entry from which to start the search.

scope

(IN) Specifies the scope of the search and uses one of the following flags:

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the immediate subordinates of the entry specified by the base parameter.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

filter

(IN) Points to a search filter.

If NULL is passed, a default filter ("objectclass=*") is used, a filter which matches all entries in the directory. Using a NULL filter is not recommended for subtree searches on trees that potentially have hundreds of thousands of entries.

Simple filters take the form of strings: attribute name=attribute value. For more complex filters, see [“Using Search Filters” on page 37](#).

attrs

(IN) Points to a NULL-terminated array of strings indicating which attributes to return with each matching entry. To return only entry names (and no attributes), set the first, and only string in the array, to LDAP_NO_ATTRS. To return all attributes, set this parameter to NULL.

For example, to return the cn, surname, and givenName attributes, declare attrs as: `char* attrs[]={ "cn", "surname", "givenName", NULL};`

attrsonly

(IN) Specifies whether to return just attributes or attributes and values.

- Zero—return both attributes and values
- Non-zero—return only attributes

timeout

(IN) Points to a timeval structure that specifies the maximum time to wait for the results of a search to complete. The structure specifies both the time the server waits for the operation to complete as well as the time the local function waits for the server to respond. If the timeout parameter is set to NULL, the client timeout is infinite and the server uses the timeout value stored in the session handle option, LDAP_OPT_TIMELIMIT (whose default value is no timeout). For more information about possible values, see [timeval \(page 476\)](#).

res

(OUT) Returns a pointer to an array of result messages if the search succeeds or NULL if no results are returned.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x55	LDAP_TIMEOUT
0x57	LDAP_FILTER_ERROR

Remarks

The LDAP_OPT_DEREF option in the LDAP session handle affects how aliases are handled during the search.

- The LDAP_DEREF_FINDING value means aliases are dereferenced when locating the base object but not during the search.
- The LDAP_DEREF_SEARCHING value means aliases are dereferenced during the search but not when locating the base object of the search.

The results in the res parameter are opaque to the caller. You must call ldap_parse_result to read the results.

Remarks

To check the results of the operation, use the `ldap_result` or the `ldap_result2error` function.

Server timeouts and size limits for this function are set using the `LDAP_OPT_TIMELIMIT` and `LDAP_OPT_SIZELIMIT` options on the LDAP handle. Client timeouts are set using the `timeout` parameter. This function has no client size limit.

See Also

[ldap_search](#) (page 251), [ldap_search_ext](#) (page 253), [ldap_search_ext_s](#) (page 256), [ldap_search_s](#) (page 259)

ldap_set_lderrno

Sets error information in an LDAP structure. This function has been deprecated; use the [ldap_set_option](#) function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int    ldap_set_lderrno (
    LDAP    *ld,
    int      errnum,
    char     *matchedDN,
    char     *errmsg);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

errnum

(IN) Specifies the LDAP error number to set.

matchedDN

(IN) Points to the name of the lowest entry in the directory that was matched on the search operation. May be NULL.

errmsg

(IN) Points to a text string that contains information from the LDAP server about this error. May be NULL.

Return Values

Always returns LDAP_SUCCESS.

Remarks

The `ldap_set_lderrno` function can be used to add or modify information about an error in an LDAP handle. This information can be retrieved in a subsequent call to the `ldap_get_lderrno` function.

The LDAP libraries make a copy of the string before storing it in the LDAP handle, so you do not need to preserve the original string after the call.

NOTE: This is not a standard IETF function. It has been added for compatibility with other LDAP vendors' libraries and should not be used in new applications. Use the `ldap_set_option` function with `LDAP_OPT_ERR_NUMBER`, `LDAP_OPT_MATCHED_DN`, and `LDAP_OPT_ERROR_STRING`.

See Also

[ldap_get_lderrno \(page 164\)](#), [ldap_set_option \(page 266\)](#)

ldap_set_option

Sets the value of session-wide parameters.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_set_option (
    LDAP          *ld,
    int           option,
    LDAP_CONST void *invalue);
```

Parameters

ld

(IN) Points to the session handle. If this is NULL, the function accesses the global defaults.

option

(IN) Specifies the name of the option which is being set (see [Section 6.10, “Session Preference Options,”](#) on page 413).

invalue

(IN) Points to the value to which the specified option is set.

Return Values

0x00	LDAP_SUCCESS
-1	Failure

Remarks

The ldap_init function returns the value for the ld parameter. If you use the ldap_set_option function before calling ldap_init and use NULL for the ld parameter, the values are set globally and copied to all LDAP session handles you create afterwards. If the ldap_set_option function is called after the ldap_init function, one of the following occurs:

- If the ld parameter is NULL, the values are set globally but do not affect the values in currently created LDAP session handles.
- If the ld parameter is set to the value returned by the ldap_init function, the values are set for only that LDAP session handle.

The following examples illustrate how to globally set two of the options.

```
/* Don't chase referrals */
rc = ldap_set_option( NULL, LDAP_OPT_REFERRALS, LDAP_OPT_OFF );

/* Set LDAP version 3 */
int version = LDAP_VERSION3;
rc = ldap_set_option( NULL, LDAP_OPT_PROTOCOL_VERSION, &version );
```

See Also

[ldap_get_option](#) (page 166)

ldap_set_rebind_proc

Sets the process that is used to bind when following referrals.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_set_rebind_proc (
    LDAP      *ld,
    LDAP_REBIND_PROC  *ldap_proc);
```

Parameters

ld

(IN) Points to the session handle. If this is NULL, the function sets the rebind process globally.

ldap_proc

(IN) Specifies the rebind function.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.

Remarks

The ldap_set_rebind_proc function sets the process to use for binding when an operation returns a referral. 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 has the following syntax.

```
int LIBCALL rebind_function (
    LDAP      *ld,
    const char *url,
    int        request,
    ber_int_t  msgid)
{
    /* the body must perform a synchronous bind */
}
```

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` function to parse the string into its components.

The `request` parameter specifies the request operation that generated the referral. For possible values, see [Section 6.8, “Request Message Types,” on page 412](#).

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

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.

Your 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.

- If an anonymous bind is sufficient for your application, then you do not need to provide a `rebind` process. The LDAP libraries with the `LDAP_OPT_REFERRALS` option set to `ON` (default value) will automatically follow referrals using an anonymous bind.
- If your 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.

For sample code, see [rebind.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_set_option \(page 266\)](#), [ldap_url_parse \(page 282\)](#)

ldap_simple_bind

Asynchronously authenticates an entry to the directory.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_simple_bind (
    LDAP      *ld,
    char      *dn,
    char      *passwd);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the name of the entry to use for authentication. For an anonymous authentication, set this parameter to NULL.

passwd

(IN) Points to the entry's password which will be compared to the entry's userPassword attribute. For an anonymous authentication, set this parameter to NULL.

Return Values

>0	Message ID of operation
-1	Failure

Remarks

To obtain the results of the operation, call the ldap_result function using the message ID returned by the ldap_simple_bind function.

If the function returns a -1, use the ldap_get_option function with the option parameter set to LDAP_OPT_RESULT_CODE to retrieve the error code from the LDAP session handle.

By default, eDirectory does not accept clear text passwords. Make sure that the parameter for encrypted passwords is set to allow unencrypted passwords.

An anonymous bind to an eDirectory directory allows clients to access whatever the [Public] user has been granted access to. By default, this is just enough to allow the user to find an eDirectory server, match a distinguished name, and authenticate.

The LDAP_OPT_NETWORK_TIMEOUT option (set by calling [ldap_set_option \(page 266\)](#)) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by spaces in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing NULL for the ld parameter of ldap_set_option sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init \(page 174\)](#) or [ldapssl_init \(page 297\)](#). To clear the timeout pass NULL for the invalue parameter of ldap_set_option.

A connection timeout will cause an LDAP_SERVER_DOWN error (81) "Can't contact LDAP server".

See Also

[ldap_bind \(page 89\)](#), [ldap_simple_bind_s \(page 272\)](#), [ldap_unbind](#), [ldap_unbind_s \(page 278\)](#), [ldap_unbind_ext](#), [ldap_unbind_ext_s \(page 279\)](#)

ldap_simple_bind_s

Synchronously authenticates the specified client to the LDAP server using a distinguished name and password.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

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

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the entry who is authenticating. For an anonymous authentication, set this parameter to NULL.

passwd

(IN) Points to the client's password. For an anonymous authentication, set this parameter to NULL.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x54	LDAP_DECODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Remarks

By default, eDirectory does not accept clear text passwords. Make sure that the parameter for encrypted passwords is set to allow unencrypted passwords.

An anonymous bind to an eDirectory directory allows clients to access whatever the [Public] user has been granted access to. By default, this is just enough to allow the user to find an eDirectory server, match a distinguished name, and authenticate.

The LDAP_OPT_NETWORK_TIMEOUT option (set by calling [ldap_set_option \(page 266\)](#)) enables you to set a timeout for the initial connection to a server. If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.

Using the connection timeout, you can also specify multiple hosts separated by spaces in a bind call, then use a timeout to determine how long your application will wait for an initial response before attempting a connection to the next host in the list.

Passing NULL for the ld parameter of ldap_set_option sets this timeout as the default connection timeout for subsequent session handles created with [ldap_init \(page 174\)](#) or [ldapssl_init \(page 297\)](#). To clear the timeout pass NULL for the invalue parameter of ldap_set_option.

A connection timeout will cause an LDAP_SERVER_DOWN error (81) "Can't contact LDAP server".

For sample code, see [bind.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_simple_bind \(page 270\)](#), [ldap_unbind](#), [ldap_unbind_s \(page 278\)](#), [ldap_bind \(page 89\)](#), [ldap_bind_s \(page 97\)](#)

ldap_sort_entries

Sorts a chain of entries, returned by an LDAP search operation, using either the entries' DN or a specified attribute.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_sort_entries (
    LDAP          *ld,
    LDAPMessage   **res,
    char          *attr,
    int           (*cmp) (const void *, const void *));
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

res

(IN) Points to an LDAPMessage containing the results returned by the ldap_result or ldap_search_s function.

attr

(IN) Points to name of the attribute to use for sorting. Pass in NULL to sort by distinguished name.

cmp

(IN) Points to a function to use for sorting. This function returns an int and has two void pointers for parameters.

Return Values

0	LDAP_SUCCESS
-1	Failure.

Remarks

If the function returns failure, use [ldap_get_option \(page 166\)](#) to check the LDAP_OPT_RESULT_CODE option in the LDAP handle for the error code.

See Also

[ldap_result \(page 231\)](#), [ldap_search_s \(page 259\)](#), [ldap_sort_strcasecmp \(page 276\)](#)

ldap_sort_strcasecmp

Compares two strings, ignoring any differences in upper and lower case characters between the strings.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_sort_strcasecmp (
    const void    *a,
    const void    *b);
```

Parameters

a

(IN) Points to the address of the pointer for first string to use in the compare.

b

(IN) Points to the address of the pointer for second string to use in the compare.

Return Values

0	String a is equal to string b.
>0	String a is greater than string b.
<0	String a is less than string b.

See Also

[ldap_sort_values \(page 277\)](#), [ldap_sort_entries \(page 274\)](#)

ldap_sort_values

Sorts an array of values retrieved from an ldap_get_values function.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_sort_values (
    LDAP      *ld,
    char      **vals,
    int        (*cmp) (const void *, const void *));
```

Parameters

ld

(IN) Points to the handle of the LDAP session.

vals

(IN) Points to the array of values to sort.

cmp

(IN) Points to the function to use for sorting. This function returns an int and has two void pointers for parameters. The ldap_sort_strcasecmp function can be used for this parameter to compare ASCII strings.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”.

See Also

[ldap_get_values](#) (page 167), [ldap_sort_strcasecmp](#) (page 276)

ldap_unbind, ldap_unbind_s

Unbinds from the directory, closes the connection, and frees resources associated with the session. Functionally, there are no differences between ldap_unbind and ldap_unbind_s.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_unbind[_s] (
    LDAP *ld);
```

Parameters

ld

(IN) Points to the handle of the LDAP session that is to be unbound.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”.

Remarks

After the call to ldap_unbind[_s], the session handle (ld) is invalid.

Note that there are no functional differences between the four unbind functions.

See Also

[ldap_unbind_ext, ldap_unbind_ext_s \(page 279\)](#)

ldap_unbind_ext, ldap_unbind_ext_s

Unbinds from the directory, closes the connection, and frees resources associated with the session. Functionally, there are no differences between ldap_unbind_ext and ldap_unbind_ext_s.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_unbind_ext[_s] (
    LDAP          *ld,
    LDAPControl   **serverctrls,
    LDAPControl   **clientctrls);
```

Parameters

- ld**
(IN) Points to the handle of the LDAP session that is to be unbound.
- serverctrls**
(IN) Points to a list of server controls. Use NULL to specify no server controls.
- clientctrls**
(IN) Points to a list of client controls. Use NULL to specify no client controls.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “LDAP Return Codes”.

Remarks

After the call to ldap_unbind_ext[_s], the session handle (ld) is invalid.

Ldap_unbind_ext allows controls to be specified with the operation. eDirectory does not currently support any server-side controls to use with an unbind operation.

Note that there are no functional differences between the four unbind functions.

See Also

[ldap_unbind](#), [ldap_unbind_s](#) (page 278)

ldap_url_desc2str

Converts from an LDAPURLDesc structure to a URL string.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

char* ldap_url_parse (
    LDAPURLDesc *ludp);
```

Parameters

ludpp

(IN) Points to an LDAPURLDesc structure that contains the components of the URL.

Return Values

This function returns an LDAP URL in string format. NULL is returned if an allocation error occurs.

Remarks

Since this function does not return a standard LDAP error code, you should not call `ldap_err2string` to parse the return code.

An LDAP URL has the following format:

```
ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?
<filter>?<extensions>
```

If you plan to convert an LDAPURLDesc structure to an LDAP URL string then back again, use `ldap_url_parse_ext` (page 284), as it is better retains the original format of the structure.

The string returned by this function should be freed with `ldap_memfree`.

See Also

`ldap_memfree` (page 178), `ldap_url_parse_ext` (page 284)

ldap_url_parse

Parses the specified URL into its components.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_url_parse (
    const char    *url,
    LDAPURLDesc  **ludpp);
```

Parameters

url

(IN) Points to the URL that you want to parse.

ludpp

(OUT) Points to an LDAPURLDesc structure that contains the components of the URL.

Return Values

This function does not return a standard LDAP error code. It returns one of the following:

0x00	LDAP_URL_SUCCESS
Non-zero	Failure
0x01	LDAP_URL_ERR_MEM—can't allocate memory space
0x02	LDAP_URL_ERR_PARAM—invalid parameter
0x03	LDAP_URL_ERR_NOTLDAP—URL doesn't begin with "ldap[s]://"
0x04	LDAP_URL_ERR_BADENCLOSURE—URL is missing trailing ">"
0x05	LDAP_URL_ERR_BADURL—invalid URL
0x06	LDAP_URL_ERR_BADHOST—host port is invalid
0x07	LDAP_URL_ERR_BADATTRS—invalid or missing attributes
0x08	LDAP_URL_ERR_BADSCOPE—invalid or missing scope string
0x09	LDAP_URL_ERR_BADFILTER—invalid or missing filter
0x0A	LDAP_URL_ERR_BADEXTS—invalid or missing extensions

Remarks

Since this function does not return a standard LDAP error code, you should not call `ldap_err2string` to parse the return code.

[ldap_url_parse_ext \(page 284\)](#) performs a similar function, but handles default values differently. `ldap_url_parse_ext` is better suited for situations where you must convert an `LDAPURLDesc` structure back to a URL string, retaining the original form of the string.

The following lists describes how each field in the `LDAPURLDesc` structure is determined from the LDAP URL:

`lud_scheme`: Contains the URL scheme (either `ldap` or `ldaps`).

`lud_host`: Points to the name of the host as a dotted IP address or DNS format. Set to an empty string if missing from URL.

`lud_port`: Contains the port from the URL. Set to 389 or 636 if missing, depending on the scheme.

`lud_dn`: Points to the distinguished name of the base entry from the URL. Set to an empty string if missing.

`lud_attrs`: Points to a NULL-terminated list of attributes specified in the URL. NULL if no attributes specified.

`lud_scope`: Contains the scope in the URL and uses one of the following flags.

`LDAP_SCOPE_BASE` (0x00)-searches the entry specified by the base parameter.

`LDAP_SCOPE_ONELEVEL` (0x01)-searches the entry specified by the base parameter and one level beneath that entry.

`LDAP_SCOPE_SUBTREE` (0x02)-searches the entire subtree starting with the entry specified by the base parameter.

Default is `LDAP_SCOPE_BASE`.

`lud_filter`: Points to the search filter specified in the URL. If NULL is passed, a default filter ("`objectclass=*`") is used.

`lud_exts`: Points to a NULL-terminated list of the extensions specified in the URL. NULL if no extensions are specified.

`lud_crit_exts`: Specifies whether or not any critical extensions are included. Set to 1 if any critical extension are included, otherwise set to 0.

See Also

[ldap_free_urldesc \(page 158\)](#), [ldap_url_parse_ext \(page 284\)](#)

ldap_url_parse_ext

Parses the specified URL into its components.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_url_parse_ext (
    const char    *url,
    LDAPURLDesc  **ludpp);
```

Parameters

url

(IN) Points to the URL that you want to parse.

ludpp

(OUT) Points to an LDAPURLDesc structure that contains the components of the URL.

Return Values

This function does not return a standard LDAP error code. It returns one of the following:

0x00	LDAP_URL_SUCCESS
Non-zero	Failure
0x01	LDAP_URL_ERR_MEM—can't allocate memory space
0x02	LDAP_URL_ERR_PARAM—invalid parameter
0x03	LDAP_URL_ERR_NOTLDAP—URL doesn't begin with "ldap[s]://"
0x04	LDAP_URL_ERR_BADENCLOSURE—URL is missing trailing ">"
0x05	LDAP_URL_ERR_BADURL—invalid URL
0x06	LDAP_URL_ERR_BADHOST—host port is invalid
0x07	LDAP_URL_ERR_BADATTRS—invalid or missing attributes
0x08	LDAP_URL_ERR_BADSCOPE—invalid or missing scope string
0x09	LDAP_URL_ERR_BADFILTER—invalid or missing filter
0x0A	LDAP_URL_ERR_BADEXTS—invalid or missing extensions

Remarks

Since this function does not return a standard LDAP error code, you should not call `ldap_err2string` to parse the return code.

`ldap_url_parse` performs a similar function but handles default values differently.

`ldap_url_parse_ext` is better suited for situations where you must convert an `LDAPURLDesc` structure back to a URL string, retaining the original form of the string.

The following lists describes how each field in the `LDAPURLDesc` structure is determined from the LDAP URL:

`lud_scheme`: Contains the URL scheme (either `ldap` or `ldaps`).

`lud_host`: Points to the name of the host as a dotted IP address or DNS format Set to an empty string if missing from URL.

`lud_port`: Contains the port from the URL. Set to 0 if missing.

`lud_dn`: Points to the distinguished name of the base entry from the URL. Set to an empty string if missing.

`lud_attrs`: Points to a NULL-terminated list of attributes specified in the URL. NULL if no attributes specified.

`lud_scope`: Contains the scope in the URL and uses one of the following flags.

`LDAP_SCOPE_BASE` (0)-searches the entry specified by the base parameter.

`LDAP_SCOPE_ONELEVEL` (1)-searches the entry specified by the base parameter and one level beneath that entry.

`LDAP_SCOPE_SUBTREE` (2)-searches the entire subtree starting with the entry specified by the base parameter.

Set to `LDAP_SCOPE_DEFAULT` (-1) if missing.

`lud_filter`: Points to the search filter specified in the URL. If NULL is passed, a default filter ("`objectclass=*`") is used.

`lud_exts`: Points to a NULL-terminated list of the extensions specified in the URL. NULL if no extensions are specified.

`lud_crit_exts`: Specifies whether or not any critical extensions are included. Set to 1 if any critical extension are included, otherwise set to 0.

See Also

[ldap_free_urldesc \(page 158\)](#), [ldap_url_parse \(page 282\)](#)

ldap_url_search

Uses the specified URL to perform an asynchronous search operation.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_url_search (
    LDAP      *ld,
    const char *url,
    int        attrsonly);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

url

(IN) Points to the URL to use in the search operation.

attrsonly

(IN) Specifies whether attribute values are returned with the specified attributes.

- 0—return attributes and values
- 1—return only attributes

Return Values

Returns the message ID of the search operation.

Remarks

To check the results of the operation, use the `ldap_result` or the `ldap_result2error` function.

Server timeouts and size limits for this function are set using the `LDAP_OPT_TIMELIMIT` and `LDAP_OPT_SIZELIMIT` options on the LDAP handle. This function has no client time or size limits.

An LDAP URL has the following format:

```
ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?<filter>?<extensions>
```

<code>ldap://</code>	Specifies a clear-text connection.
<code>ldaps://</code>	Specifies an SSL connection.
<code><hostname></code>	Specifies the LDAP server.
<code><port></code>	Specifies the port number. Defaults to zero if unspecified. Port 0 causes the appropriate port (389 for clear-text and 636 for SSL) to be selected when the connection is made.
<code><base_dn></code>	Specifies the distinguished name of an entry in the directory where the search begins. Defaults to an empty string which starts the search at the top level of the directory.
<code><attributes></code>	Specifies a comma-separated list of attributes to return. If missing, all attributes are returned.
<code><scope></code>	Specifies the scope of the search: base—search just base entry one—search the immediate subordinates of the base entry sub—search the entire subtree of the base entry Defaults to base.
<code><filter></code>	Specifies a search filter. If empty, defaults to <code>(objectclass=*)</code> .
<code><extensions></code>	Specifies a comma-separated list of extension in one of the following formats: <code>[!]type=value</code> <code>[!]type</code> Extensions prefixed with "!" are considered critical extensions.

The following examples illustrate this URL format:

- `ldap://acme.com/ou=sales,o=acme?sn,telephoneNumber?sub?(objectclass=inetOrgPerson)?ext1=value1,ext2=value2`
- `ldaps://1.2.3.4:636/o=novell??one`

See Also

[ldap_free_urldesc](#) (page 158), [ldap_url_search_s](#) (page 288), [ldap_url_search_st](#) (page 290)

ldap_url_search_s

Uses the specified URL to perform a synchronous search operation.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_url_search_s (
    LDAP      *ld,
    const char *url,
    int        attronly,
    LDAPMessage **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

url

(IN) Points to the URL to use in the search operation.

attronly

(IN) Specifies whether attribute values are returned with the specified attributes.

- 0—return attributes and values
- 1—return only attributes

res

(OUT) Returns a pointer to an array of result messages if the search succeeds or NULL if no results are returned.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “LDAP Return Codes”.

Remarks

To check the results of the operation, use the `ldap_result` or the `ldap_result2error` function.

Server timeouts and size limits for this function are set using the LDAP_OPT_TIMELIMIT and LDAP_OPT_SIZELIMIT options on the LDAP handle. This function has no client time or size limits.

See Also

[ldap_free_urldesc](#) (page 158), [ldap_url_search](#) (page 286), [ldap_url_search_st](#) (page 290)

[LDAPMessage](#) (page 461)

ldap_url_search_st

Uses the specified URL to perform a synchronous search operation that includes a specified time limit.

LDAP Version: v3

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_url_search_st (
    LDAP          *ld,
    const char    *url,
    int            attrsonly,
    struct timeval *timeout,
    LDAPMessage   **res);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

url

(IN) Points to the URL to use in the search operation.

attrsonly

(IN) Specifies whether attribute values are returned with the specified attributes.

- 0—return attributes and values
- 1—return only attributes

timeout

(IN) Points to a timeval structure that specifies the maximum time to wait for the results of a search to complete. It specifies both the time the server waits for the operation to complete as well as the time the local function waits for the server to respond. If the timeout parameter is set to NULL, the client timeout is infinite and the server uses the timeout value stored in the session handle option, LDAP_OPT_TIMELIMIT (whose default value is no timeout). For more information about possible values, see [timeval \(page 476\)](#).

res

(OUT) Returns a pointer to an array of result messages if the search succeeds or NULL if no results are returned.

Return Values

Returns the message ID of the search operation.

Remarks

To check the results of the operation, use the `ldap_result` or the `ldap_result2error` function.

Server timeouts and size limits for this function are set using the `LDAP_OPT_TIMELIMIT` and `LDAP_OPT_SIZELIMIT` options on the LDAP handle. Client timeouts are set using the `timeout` parameter. This function has no client size limit.

See Also

[ldap_free_urldesc \(page 158\)](#), [ldap_url_search \(page 286\)](#), [ldap_url_search_s \(page 288\)](#)
[timeval \(page 476\)](#), [LDAPMessage \(page 461\)](#)

ldap_value_free

Frees the memory allocated for an array of string values.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_value_free (
    char    **vals);
```

Parameters

vals

(IN) Points to the array of values returned by the ldap_get_values function.

Remarks

The memory for each value is freed as well as the array.

If NULL is passed for the vals parameter, this function does nothing.

See Also

[ldap_get_values](#) (page 167), [ldap_count_values](#) (page 119), [ldap_value_free_len](#) (page 293)

ldap_value_free_len

Frees the memory allocated for an array of berval structures.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

void ldap_value_free_len (
    struct berval **vals);
```

Parameters

vals

(IN) Points to the array of values returned by the `ldap_get_values_len` function.

Remarks

The memory for each berval structure is freed as well as the array.

If NULL is passed for the vals parameter, this function does nothing.

See Also

[ldap_get_values_len](#) (page 169), [ldap_count_values_len](#) (page 120), [ldap_value_free](#) (page 292)

ldapssl_client_init

Initializes the SSL (Secure Socket Layer) library.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_client_init (
    const char    *certFile,
    void          *reserved);
```

Parameters

certFile

(IN) Points to the trusted root certificate file, a fully-qualified file path and the file must contain a DER encoded certificate.

reserved

(IN) Not currently used. Pass a NULL.

Return Values

0	Success
-1	Failure

Remarks

The LDAP SSL library provides SSL server authentication. In order to verify the server, the library needs to be configured with a trusted root certificate.

The certFile parameter is the fully qualified path of a file containing a trusted root certificate DER encoded.

It is also possible to pass NULL in the certFile parameter and use ldapssl_add_trusted_cert to add trusted root certificates to the LDAP SSL library. The API ldapssl_add_trusted_cert accepts DER and B64 (PEM) encoded certificates.

If the SSL handshake fails, the LDAP library returns an LDAP_SERVER_DOWN error. The handshake can fail because the server is down or because SSL has not been set up correctly on the client or LDAP server.

When you are finished with the SSL library, you should call the `ldapssl_client_deinit` function.

For sample code, see `sslbind.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

`ldapssl_client_deinit` (page 296), `ldapssl_init` (page 297), `ldapssl_install_routines` (page 161)

ldapssl_client_deinit

Deinitializes the SSL library.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_client_deinit (
    void);
```

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. See “ LDAP Return Codes ”.

Remarks

This function must be called after you are finished using the SSL library. Before calling this function, all SSL LDAP session handles must be closed using the ldap_unbind function.

For sample code, see [sslbind.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldap_unbind](#), [ldap_unbind_s](#) (page 278), [ldapssl_client_init](#) (page 294)

ldapssl_init

Creates an LDAP session handle that is SSL enabled.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

LDAP * ldapssl_init (
    const char    *host,
    int           port,
    int           secure);
```

Parameters

host

(IN) Contains the names of the available hosts, each separated by a space, or a list of IP addresses (in dot format) of the hosts, each separated by a space. If a port number is included with the name or the address, it is separated from them with a colon (:).

port

(IN) Contains the TCP port number to connect to, which for an SSL connection is the SSL port number of the LDAP server. If a port number is included with the host parameter, this parameter is ignored.

secure

(IN) Specifies whether the connection is established over SSL.

- Zero—do not establish the connection over SSL (which makes this function essentially the same as the ldap_init function)
- Non-zero—establish the connection over SSL

Return Values

>0	Success; session handle
NULL	Unsuccessful

Remarks

If you connect to an LDAP v2 server, you must call an LDAP bind operation before performing any operations. If you connect to an LDAP v3 server, some operations can be performed before calling a bind operation.

Before calling this function, you must first call the `ldapsl_client_init` function which initializes the SSL library.

Calling the `ldapsl_init` function is equivalent to calling the `ldap_init` function followed by the `ldapsl_install_routines` function.

The `ldapsl_init` function does not actually communicate with the LDAP server. Communication begins when the application binds or does some other operation.

The LDAP libraries first contact the first server listed in the host parameter. If they are unable to communicate with that server, they try the next server and then the next.

The session handle returned contains opaque data identifying the session. To get or set handle information, use `ldap_set_option` and `ldap_get_option`. For a list of the handle options, see [Section 6.10, “Session Preference Options,” on page 413](#).

For sample code, see `sslbind.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

IMPORTANT: The `ldap_init` function allocates memory for the LDAP structure. This memory must be freed by calling `ldap_unbind` or `ldap_unbind_s` even when an LDAP bind function is not called or the LDAP bind function fails.

See Also

[ldapsl_client_init](#) (page 294), [ldap_init](#) (page 174)

ldapssl_add_trusted_cert

Adds certificates to the list of trusted certificates.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_add_trusted_cert (
    void    *cert,
    int     type);
```

Parameters

cert

(IN) Points to the trusted root certificate to add.

type

(IN) Certificate type. This must be one of the following values:

- LDAPSSL_CERT_FILETYPE_B64
- LDAPSSL_CERT_FILETYPE_DER
- LDAPSSL_CERT_BUFFTYPE_B64
- LDAPSSL_CERT_BUFFTYPE_DER

Return Values

0	Success
-1	Failure

Remarks

This function can be called repeatedly to build a group of trusted certificates. It supports certificates encoded as DER and B64 (PEM) formats.

When one of the "FILETYPE" types is specified (see the type parameter), the cert parameter must be a pointer to a character array containing the fully qualified filename of the file containing the certificate. When one of the "BUFFTYPE" types are specified, the cert parameter must be a pointer an [LDAPSSL_Cert \(page 469\)](#) structure.

For sample code, see `sslbind.c`, `sslbind_interactive.c` (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

`ldapssl_client_init` (page 294), `ldapssl_client_deinit` (page 296), `ldapssl_init` (page 297), `ldapssl_install_routines` (page 161)

ldapssl_get_cert

Returns a certificate encoded in the requested format.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_get_cert (
    void          *certHandle,
    int           type,
    LDAPSSL_Cert  *cert);
```

Parameters

certHandle

(IN) certificate handle received by a verify callback function.

type

(IN) Desired certificate encoding. This must be one of the following values:

- LDAPSSL_CERT_BUFFTYPE_B64
- LDAPSSL_CERT_BUFFTYPE_DER

cert

(I/O) Pointer to an [LDAPSSL_Cert \(page 469\)](#) structure.

Return Values

0	Success
-1	Failure

Remarks

Applications use `ldapssl_get_cert` to retrieve the certificate from the certificate handle passed to the [ldapssl_set_verify_callback \(page 311\)](#) function.

The `certHandle` parameter is the certificate handle (void *) received by the verify callback routine.

An [LDAPSSL_Cert \(page 469\)](#) structure contains two elements, `length` and `data`. The `data` element is a pointer to a buffer allocated by the application and `length` is the size of the buffer. To determine

the correct size for the buffer, applications can pass in an LDAPSSL_Cert structure with the data element set to NULL and the length element will be updated with the appropriate size. The appropriate memory can then be allocated and ldapssl_get_cert can be called again with the LDAPSSL_Cert data element set to the allocated memory.

Applications can use ldapssl_get_cert to retrieve the certificate information as a buffer and use it as desired. One possibility is to add it to the list of trusted certificates using [ldapssl_add_trusted_cert \(page 299\)](#). After adding the certificate to the list of trusted certificates, the verify callback routine will no longer be called if the certificate is received when establishing future SSL connections.

For sample code, see [sslbind_interactive.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_set_verify_callback \(page 311\)](#), [ldapssl_add_trusted_cert \(page 299\)](#)

ldapssl_get_cert_attribute

Returns requested certificate information.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_get_cert_attribute (
    void          *certHandle,
    unsigned long  attrID,
    void          *value,
    int           *length);
```

Parameters

certHandle

(IN) Certificate handle received by a verify callback function.

attrID

(IN) Certificate information to return. See [Table 6-3 on page 408](#).

value

(OUT) Pointer to memory appropriate for the information requested.

length

(I/O) Pointer to length of value parameter memory.

Return Values

0	Success
-1	Failure

Remarks

This function is used to query information about a server certificate received by the verify callback routine.

The certHandle parameter is the certificate handle (void *) received by the verify callback routine.

The attrID parameter specifies the information to retrieve, and the value parameter points to memory appropriate for the information. For specific attrID(s) and data types see [Table 6-3 on page 408](#).

The length parameter is both an input and an output. On input, length is the size of the memory pointed to by the value parameter. On output, length is updated to reflect the actual size of the information copied.

In order to allocate memory, applications can pass in a NULL for the value parameter and the length parameter will be updated with the appropriate size, but no data will be copied. Applications can then allocate the appropriate memory and call ldapssl_get_cert_attribute again to retrieve the information.

For sample code, see [sslbind_interactive.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_set_verify_callback](#) (page 311)

ldapssl_set_verify_mode

Sets the server certificate verification mode used when establishing an SSL connection.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_set_verify_mode (
    int mode);
```

Parameters

mode

(IN) Server certificate verify mode. This must be set to the following value:

- LDAPSSL_VERIFY_SERVER

Return Values

0	Success
-1	Failure

Remarks

The default mode is server verification (LDAPSSL_VERIFY_SERVER).

See Also

[ldapssl_get_verify_mode \(page 310\)](#)

ldapssl_set_client_cert

Specifies the client certificate to be used with client-based certificate authentication (CBCA).

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_set_client_cert (
    void    *cert,
    int     type
    void    *password);
```

Parameters

cert

(IN) Points to the encoded client certificate file.

type

(IN) Certificate type. This must be one of the following values:

- LDAPSSL_CERT_FILETYPE_B64
- LDAPSSL_CERT_FILETYPE_DER
- LDAPSSL_CERT_BUFFTYPE_B64
- LDAPSSL_CERT_BUFFTYPE_DER

password

(IN) Points to the client certificate password.

Return Values

0	Success
-1	Failure

Remarks

For sample code, see [mutual.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm), (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_set_client_private_key](#) (page 308)

ldapssl_set_client_private_key

Specifies the private key to be used with client-based certificate authentication (CBCA).

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_set_client_private_key (
    void    *key,
    int     type
    void    *password);
```

Parameters

key

(IN) Points to the encoded client private key file.

type

(IN) Key type. This must be one of the following values:

- LDAPSSL_CERT_FILETYPE_B64
- LDAPSSL_CERT_FILETYPE_DER
- LDAPSSL_CERT_BUFCTYPE_B64
- LDAPSSL_CERT_BUFCTYPE_DER

password

(IN) Points to the client private key password.

Return Values

0	Success
-1	Failure

Remarks

For sample code, see [mutual.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm), (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_set_client_cert](#) (page 306)

ldapssl_get_verify_mode

Returns the current server certificate verification mode that is used when establishing an SSL connection.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_get_verify_mode (
    int *mode);
```

Parameters

mode

(OUT) Current server certificate verify mode. This will be the following value:

- LDAPSSL_VERIFY_SERVER

Return Values

0	Success
-1	Failure

Remarks

The default mode is server verification (LDAPSSL_VERIFY_SERVER).

See Also

[ldapssl_set_verify_mode \(page 305\)](#)

ldapssl_set_verify_callback

Sets the routine to be called during SSL connection establishment if the server certificate received is not trusted.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_set_verify_callback (
    int (LIBCALL *certVerifyFunc) (void*) );
```

Parameters

certVerifyFunc

(IN) Callback routine, called during SSL connection establishment if the server certificate received is not trusted.

Return Values

0	Success
-1	Failure

Remarks

The certVerifyFunc must be a pointer to a function that takes one parameter (a void *) and returns an int.

If an untrusted server certificate is received while establishing an SSL connection, the callback routine is called with a handle to the certificate (void*).

This handle can be passed into [ldapssl_get_cert_attribute \(page 303\)](#) to query specific certificate information.

In order to accept the server certificate and continue the SSL connection, the callback routine should return LDAPSSL_CERT_ACCEPT. To reject the server certificate and abort the connection the callback routine should return LDAPSSL_CERT_REJECT.

For sample code, see [sslbind_interactive.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_get_cert_attribute](#) (page 303), [ldapssl_get_cert](#) (page 301)

ldapssl_start_tls

Starts Transport Layer Security (TLS/SSL). Works with eDirectory 8.7 or higher.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_start_tls (
    LDAP *ld);
```

Parameters

ld

(IN) LDAP session handle.

Return Values

0	Success
-1	Failure

Remarks

For sample code, see [starttls.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_stop_tls](#) (page 314)

ldapssl_stop_tls

Stops Transport Layer Security (TLS/SSL). Works with eDirectory 8.7 or higher.

LDAP Version: v3

Library: *ldapssl.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_ssl.h>

int ldapssl_stop_tls (
    LDAP *ld);
```

Parameters

ld

(IN) LDAP session handle.

Return Values

0	Success
-1	Failure

Remarks

For sample code, see [starttls.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

See Also

[ldapssl_start_tls](#) (page 313)

LDAP Extension Functions

4

This chapter describes the Novell LDAP extensions for naming contexts and replicas. These extensions allow access to the following directory features:

- Naming contexts: split, join, get number of entries, abort operation
- Replicas: add, remove, change type, list on server, return information
- Replica synchronization: to a specified server, to all replicas, at a specified time
- Schema synchronization
- Get effective eDirectory rights for attributes
- Get DN of logged in caller
- Restart the LDAP server
- Event monitoring

All of the naming context and replica functions are synchronous functions. If the naming context or replica is in a state that makes the requested operation possible, eDirectory responds to the client with a successful completion code. eDirectory then completes the operation in the background since some operations on large trees can take hours to complete. Clients can check on the status of the operation by calling the `ldap_get_replica_info` function.

All of these functions require LDAP extensions on the LDAP server.

NOTE: LDAP distinguished names are UTF-8 encoded.

Renamed Functions The "naming context" terminology is now obsolete. The following functions have been renamed to replace "naming context" terminology with "partition":

- `ldap_create_naming_context`, renamed to `ldap_split_partition`.
- `ldap_merge_naming_contexts`, renamed to `ldap_merge_partitions`.
- `ldap_naming_context_entry_count`, renamed to `ldap_partition_entry_count`.
- `ldap_request_naming_context_sync`, renamed to `ldap_request_partition_sync`.
- `ldap_abort_naming_context_operation`, renamed to `ldap_abort_partition_operation`.
- `ldap_get_context_identity_name`, renamed to `ldap_get_bind_dn`.
- `ldap_create_orphan_naming_context`, renamed to `ldap_split_orphan_partition`.
- `ldap_remove_orphan_naming_context`, renamed to `ldap_remove_orphan_partition`.

ldap_abort_partition_operation

Aborts the last partition operation on the specified partition.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_abort_partition_operation (
    LDAP      *ld,
    char      *dn,
    int       flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of partition whose current operation should be aborted.

flags

(IN) Set to zero. Not currently used.

Return Values

0x00	LDAP_SUCCESS or no partition operation is pending.
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

In eDirectory, partition operations include

- Adding, changing, and removing replicas
- Joining and splitting partitions

At any given time, only one partition operation is pending. If a partition operation is not pending when this function is called, the function returns LDAP_SUCCESS.

For sample code, see [abortpo.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.29) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.30) and there is no responseValue.

```
ResponseBer
  NULL
```

See Also

[ldap_add_replica \(page 318\)](#)
[ldap_change_replica_type \(page 322\)](#)
[ldap_create_partition \(page 324\)](#)
[ldap_merge_partitions \(page 344\)](#)
[ldap_remove_replica \(page 368\)](#)

ldap_add_replica

Adds a replica to the specified directory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_add_replica (
    LDAP          *ld,
    char          *dn,
    char          *serverDN,
    LDAP_REPLICA_TYPE  replicaType,
    int           flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of the replica's partition root.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server on which a new replica is to be added.

replicaType

(IN) Specifies the type of replica to add (see [Section 6.7, “Replica Types,” on page 411](#)).

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.

0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This operation is performed on the server with the master replica of the replica that is being added.

The caller must have supervisor rights to the master replica.

For sample code, see [addrepl.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.7) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  replicaType INTEGER
  serverName LDAPDN
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.8) and there is no responseValue.

```
ResponseBer
  NULL
```

See Also

[ldap_change_replica_type](#) (page 322)

[ldap_remove_replica](#) (page 368)

[ldap_abort_partition_operation](#) (page 316)

ldap_backup_object

Backs up the attribute names and values for an object.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.8 or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

ldap_backup_object (
    LDAP *ld,
    const char *dn,
    const char *passwd,
    char **objectState,
    char **objectInfo,
    char **chunkSize,
    int *size);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the object for which information is to be returned.

passwd

(IN) Points to the password for encryption and decryption, when any one of the attributes in the user object has been encrypted. If the password is supplied then the connection to the servers will be over TLS.

objectState

(IN/OUT)

objectInfo

(OUT) Points to the requested attribute names and values.

chunkSize

(OUT) Specifies the length of each chunk.

size

(OUT) Specifies the length of the information to be returned.

Return Values

Points to the distinguished name of the entry that is authenticating.

LDAP_PARAM_ERROR
LDAP_NO_MEMORY
LDAP_ENCODING_ERROR
LDAP_DECODING_ERROR
LDAP_NOT_SUPPORTED
LDAP_OPERATIONS_ERROR
LDAP_SUCCESS

See Also

[ldap_restore_object](#) (page 374)

ldap_change_replica_type

Changes the type of the specified replica on the specified directory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_change_replica_type (
    LDAP      *ld,
    char      *dn,
    char      *serverDN,
    LDAP_REPLICA_TYPE  replicaType,
    int       sflags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of the replica's partition root.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server on which the replica resides.

replicaType

(IN) Specifies the new type for the replica (see [Section 6.7, “Replica Types,” on page 411](#)).

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR

0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This operation is performed on the server with the master replica of the replica that is being changed.

The replica type of the master replica cannot be changed by directly calling this function. The master's replica type can only be changed as a side effect of using this function to change another replica to the master replica. When this happens, the old master automatically becomes a secondary replica.

The caller must have supervisor rights to the master replica.

For sample code, see [chgrepl.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.15) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  replicaType INTEGER
  serverName LDAPDN
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.16) and there is no responseValue.

```
ResponseBer
  NULL
```

See Also

[ldap_add_replica](#) (page 318)

[ldap_remove_replica](#) (page 368)

[ldap_abort_partition_operation](#) (page 316)

ldap_create_partition

Creates a new LDAP partition.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_create_partition (
    LDAP    *ld,
    char    *dn,
    int      flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Specifies the distinguished name, in LDAP format, of the child container where the new partition is created.

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

In eDirectory terminology, creating a partition splits a partition into a parent partition and a child partition at the child container specified in the call.

This operation is performed on the server with the master replica of the parent replica. The server with the parent's master replica must be running or this operation fails.

The caller must have supervisor rights to the parent's master replica.

For sample code, see [splitpart.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.3) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.4) and there is no responseValue.

```
ResponseBer
  NULL
```

See Also

[ldap_merge_partitions](#) (page 344)

[ldap_abort_partition_operation](#) (page 316)

ldap_create_orphan_partition

Creates an orphan partition on the specified server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_create_orphan_partition (
    LDAP      *ld,
    char      *serverDN,
    char      *contextName);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server that will hold the orphan naming context.

contextName

(IN) Points to the distinguished name for the partition (naming context), for example, "dc=test, dc=germany, dc=provo, dc=novell, dc=com, t=dns".

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See "LDAP Return Codes" .

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.39) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverDN      LDAPDN
  contextName   LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.40) and there is no responseValue.

```
ResponseBer
  NULL
```

See Also

[ldap_create_partition](#) (page 324)

[ldap_abort_partition_operation](#) (page 316)

ldap_event_free

Frees data allocated by the ldap event functions.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_event_free (
    void *eventData);
```

Parameters

eventData

(IN) Pointer to event data allocated by [ldap_parse_monitor_events_response \(page 356\)](#), [ldap_parse_ds_event \(page 350\)](#), or a pointer to an array of NDSEventSpecifiers.

Return Values

LDAP_SUCCESS	Request was successfully sent
[Other value]	Non-zero codes indicate errors. See “ LDAP Return Codes ” for information.

See Also

[ldap_monitor_events \(page 346\)](#), [ldap_parse_ds_event \(page 350\)](#),
[ldap_parse_monitor_events_response \(page 356\)](#)

ldap_get_bind_dn

Returns the distinguished name of the client associated with the LDAP connection.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_get_bind_dn (
    LDAP      *ld,
    char      **identity);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

identity

(OUT) Points to the distinguished name, in LDAP format, of the client.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

If the connection is not authenticated and is using an anonymous bind, the function returns an empty string.

The function allocates the memory for the identity parameter, and the caller is responsible for freeing it with the ldapx_memfree function.

The first field in the structure contains the length of the name, and the second field contains the name.

For sample code, see [getidname.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.31) and the requestValue has no value.

```
RequestBer
    NULL
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.32) and the responseValue is a BER encoding of the following:

```
ResponseBer
    identity    OCTET STRING
```

ldap_get_effective_privileges

Returns the effective rights of the specified entry to the specified attribute.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_get_effective_privileges (
    LDAP      *ld,
    char      *dn,
    char      *trusteeDN,
    char      *attrName,
    int       *privileges);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of the entry that contains the attribute in question.

trusteeDN

(IN) Points to the distinguished name, in LDAP format, of the trustee whose rights are being returned, or you can specify [Public] or [Self].

attrName

(IN) Points to attribute whose rights are being returned or you can specify [Entry Rights] or [All Attribute Rights].

privileges

(OUT) Points to bitmask of the trustee's effective rights (see [Section 6.1, “Object Access Control Rights,”](#) on page 407 and [Section 6.2, “Attribute Access Control Rights,”](#) on page 407).

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.

0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

To understand the difference between the `dn` and the `trusteeDN` arguments, suppose that an entry named Kim has a telephone number attribute, and a client named Tom wants to know if he has rights to the attribute. In this case,

- `dn` points to the distinguished name of Kim
- `trusteeDN` points to the distinguished name of Tom
- `attrName` points to Telephone Number
- `privileges` points to the rights Tom has to Kim's Telephone Number attribute

For sample code, see [getpriv.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The `requestName` is set to the OID (2.16.840.1.113719.1.27.100.33) and the `requestValue` is a BER encoding of the following:

```
RequestBer
  dn          LDAPDN
  trusteeDN   LDAPDN
  attrName    OCTET STRING
```

The `responseName` is set to the OID (2.16.840.1.113719.1.27.100.34) and the `responseValue` is a BER encoding of the following:

```
ResponseBer
  privileges  INTEGER
```

ldap_get_replication_filter

Gets the replication filter defined for an eDirectory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5, SP1

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_get_replication_filter (
    LDAP      *ld,
    char      *serverDN,
    char      **filter);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name of the server from which to read the replication filter.

filter

(OUT) Points to the address of the filter. For the format, see [Section 6.6, “Replication Filters,” on page 411](#). The returned filter must be freed by calling the `ldapx_memfree` function.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “LDAP Return Codes” .

Remarks

An eDirectory server can only have one replication filter, and that filter is used for all filtered replicas on the server.

NDS eDirectory 8.5 and above supports filtered replicas. The extension to get and set the replication filter is not available until NDS 8.5 SP1.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.37) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverName      LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.38) and the responseValue is a BER encoding of the following:

```
ResponseBer
  SEQUENCE of SEQUENCE {
    classname      OCTET STRING
    SEQUENCE of ATTRIBUTES
  }
where
  ATTRIBUTES::: OCTET STRING
```

See Also

[ldap_set_replication_filter \(page 378\)](#)

ldap_get_replica_info

Returns information about the specified replica on the specified directory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_get_replica_info (
    LDAP          *ld,
    char          *dn,
    char          *serverDN,
    struct LDAPReplicaInfo *partitionInfo);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of the replica from which information is being requested.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server containing the replica.

partitionInfo

(OUT) Points to a [LDAPReplicaInfo \(page 464\)](#) structure that returns with the replica information.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See " LDAP Return Codes ".

Remarks

The specified server holding the replica must be running or an error is returned.

For sample code, see [getrinfo.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.17) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverName      LDAPDN
  dn              LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.18) and the responseValue is a BER encoding of the following:

```
ResponseBer
  partitionID      INTEGER
  replicaState     INTEGER
  modificationTime INTEGER
  purgeTime        INTEGER
  localPartitionID INTEGER
  partitionDN      LDAPDN
  replicaType      INTEGER
  flags            INTEGER
```

See Also

[ldap_list_replicas \(page 342\)](#)

ldap_lburp_end_request

Sends an LBURP end request extended operation to the server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>ldap_lburp_end_request (          LDAP *ld,          int
sequenceNumber,          int *msgID);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

sequenceNumber

(IN) Points to the sequence number used to specify the ordering of the LBURP operation. The value in sequence number is one greater than the last LBURP operation sequence number in the LBURP operation stream. It allows the server to know when it has received all outstanding asynchronous LBURP operations.

msgidp

(OUT) Points to the message ID of the request when the add request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see LDAP Return Codes.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

To obtain the results of the operation, call the ldap_result function with the returned message ID.

See Also

[ldap_parse_lburp_end_response \(page 352\)](#)

ldap_lburp_operation_request

Sends an LBURP operation request extended operation to the server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>int ldap_lburp_operation_request(                                LDAP
*ld,                                int sequenceNumber,
LBURPUpdateOperationList **updateList,                                int *msgID)
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

sequenceNumber

(IN) Points to the sequenceNumber used to specify the ordering of the LBURP operation.

updateList

(IN) Points to an array of LDAPMod structures along with package ID, that contain the attributes and values to add/delete/modify/modify rdn entries.

msgidp

(OUT) Points to the message ID of the request when the add request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see LDAP Return Codes.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The sequence number is used to specify the ordering of LBURP operation requests. This enables the client to know the order in which the LBURP operation requests must be processed even if it receives them in a sequence different from that in which they were sent from the client. The client must set the value of sequence number of the first LBURP operation to 1, and must increment the value of sequence number for each succeeding LBURP operation.

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

See Also

[ldap_lburp_parse_operation_response](#) (page 340)

ldap_lburp_parse_operation_response

Parses LBURP operation response data when the result code is LDAP_RES_EXTENDED.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>int ldap_parse_lburp_operation_response(      LDAP
*ld,      LDAPMessage *lburpMessage,      int *resultCode,
char      **errorMsg,      int *failedOperationCount,
LBURPUpdateResult **failedOperations,      int freeIt);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

lburpOperationMessage

(IN) Pointer to the LDAPMessage returned by ldap_result.

resultCode

(OUT) Returns the resultCode from the server.

errorMessage

(OUT) Returns the error message from the server, may be NULL if no error messages are requested. This memory must be freed using ldap_memfree.

failedOperationCount

(OUT) Returns the number of failed operations from the server, may be NULL if no data is requested. This memory must be freed using ldapx_memfree.

failedOperations

(OUT) a pointer to a pointer to a structure containing the data returned by this particular LBURP operation.

The structure is allocated by this function. If the LBURP operation is success, does not have associated data the pointer will be set to NULL. When the application no longer needs the data it should free the data by calling the ldapx_memfree function.

failedOperations

(IN) If non-zero, the function will free the memory referenced by the lburpMessage parameter.

ldap_lburp_start_request

Sends an LBURP start request extended operation to the server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>int ldap_lburp_start_request (LDAP*ld,
int *msgId);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

msgidp

(OUT) Points to the message ID of the request when the add request succeeds.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see LDAP Return Codes.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

To obtain the results of the operation, call the ldap_result function with the returned message ID.

See Also

[ldap_parse_lburp_start_response \(page 354\)](#)

ldap_list_replicas

Lists all the replicas on the specified directory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_list_replicas (
    LDAP      *ld,
    char      *serverDN,
    char      ***replicaList);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server whose replicas are being listed.

replicaList

(OUT) Points to a list of replicas.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “LDAP Return Codes” .

Remarks

The function allocates the memory for replicaList, but the caller is responsible for freeing the memory with the ldapx_memfree function.

This function returns all replicas including subordinate references. The replicaList argument points to a null terminated array of strings. Each string in the array contains the distinguished name of a replica's partition root.

For sample code, see [listrepl.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.19) and the requestValue is a BER encoding of the following:

```
RequestBer
    serverDN      LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.20) and the responseValue is a BER encoding of the following:

```
ResponseBer
    replicaList    SEQUENCE OF OCTET STRINGS
```

See Also

[ldap_get_replica_info \(page 335\)](#)

ldap_merge_partitions

Joins a parent and child partition.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_merge_partitions (
    LDAP      *ld,
    char      *dn,
    int       flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Specifies the distinguished name, in LDAP format, of the child partition's root that is to be joined to its parent.

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This operation is performed on the server containing the master replica of the parent partition. The caller must have supervisor rights to the child's master replica and the parent's master replica.

For sample code, see [joinpart.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.5) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.6) and the requestValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_create_partition](#) (page 324)

[ldap_abort_partition_operation](#) (page 316)

ldap_monitor_events

Sends an EventMonitorRequest extended operation to the server. Event monitoring works with eDirectory 8.7 or higher.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_monitor_events (
    LDAP          *ld,
    int           eventCount,
    EVT_EventSpecifier[] *events,
    int           *msgId);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

eventCount

(IN) The number of events you wish to monitor.

events

(IN) An array of **EVT_EventSpecifier** structures describing the events the application wishes to monitor. The number of events is specified by eventCount.

msgId

(OUT) Set to the message id of the request if the ldap_monitor_event call succeeds. The value is undefined if the function returns a value other than LDAP_SUCCESS.

Return Values

LDAP_SUCCESS	Request was successfully sent
[Other value]	Non-zero codes indicate errors. See “ LDAP Return Codes ” for information.

Remarks

The ldap_monitor_events function sends a MonitorEventRequest extended operation to the server. The function sends the request asynchronously; it does not wait for a response from the server.

To include a filter with your request to limit the events returned see [ldap_monitor_events_filtered](#) (page 348).

See Also

[ldap_parse_ds_event](#) (page 350), [ldap_parse_monitor_events_response](#) (page 356), [ldap_event_free](#) (page 328), [ldap_monitor_events_filtered](#) (page 348)

ldap_monitor_events_filtered

Sends a filtered EventMonitorRequest extended operation to the server. Event monitoring works with eDirectory 8.7 or higher.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_monitor_events_filtered (
    LDAP          *ld,
    int           eventCount,
    EVT_FilteredEventSpecifier[] *events,
    int           *msgId);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

eventCount

(IN) The number of events you wish to monitor.

events

(IN) An array of **EVT_FilteredEventSpecifier** structures describing the events the application wishes to monitor including an event filter. The number of events is specified by eventCount.

msgId

(OUT) Set to the message id of the request if the ldap_monitor_event call succeeds. The value is undefined if the function returns a value other than LDAP_SUCCESS.

Return Values

LDAP_SUCCESS	Request was successfully sent
[Other value]	Non-zero codes indicate errors. See “ LDAP Return Codes ” for information.

Remarks

The `ldap_monitor_events_filtered` function sends a `FilteredMonitorEventRequest` extended operation to the server. The function sends the request asynchronously; it does not wait for a response from the server.

See Also

[ldap_parse_ds_event](#) (page 350), [ldap_parse_monitor_events_response](#) (page 356), [ldap_event_free](#) (page 328), [ldap_monitor_events](#) (page 346)

ldap_parse_ds_event

Parses event data when the ldap result code is LDAP_RES_EXTENDED. This result code Indicates that an error or exceptional situation occurred and events will not be monitored. Event monitoring works with eDirectory 8.7 or higher.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_ds_event (
    LDAP          *ld,
    LDAPMessage    *eventMessage,
    int            *eventType,
    int            *eventResult,
    void           **eventData,
    int            freeIt);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

eventMessage

(IN) Pointer to the LDAPMessage returned by [ldap_result \(page 231\)](#).

eventType

(OUT) Receives the type of the event.

eventResult

(OUT) Receives the result associated with the event.

eventData

(OUT) a pointer to a pointer to a structure containing the data returned by this particular event. The structure is allocated by this function. The type of the structure is determined by the eventType. If the event does not have associated data the pointer will be set to NULL. When the application no longer needs the data it should free the data by calling the [ldap_event_free \(page 328\)](#) function.

freeIt

(IN) If non-zero, the function will free the memory referenced by the eventMessage parameter.

Return Values

LDAP_SUCCESS	Request was successfully sent
[Other value]	Non-zero codes indicate errors. See “ LDAP Return Codes ” for information.

See Also

[ldap_monitor_events](#) (page 346), [ldap_parse_monitor_events_response](#) (page 356), [ldap_event_free](#) (page 328)

ldap_parse_lburp_end_response

Parses LBURP end response data when the result code is LDAP_RES_EXTENDED.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>int ldap_parse_lburp_end_response (      LDAP *ld,  
LDAPMessage *lburpEndMessage,      int *resultCode,      char  
**errorMsg,      int freeIt)
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

lburpEndMessage

(IN) Pointer to the LDAPMessage returned by [ldap_result \(page 231\)](#).

resultCode

(OUT) Returns the resultCode from the server.

errorMsg

(OUT) Returns the error message from the server, may be NULL if no error messages are requested. This memory must be freed using [ldap_memfree \(page 178\)](#).

badEventsCount

(OUT) Returns the number of bad events from the server, may be NULL if no data is requested. This memory must be freed using [ldapx_memfree \(page 386\)](#).

freeIt

(IN) If non-zero, the function will free the memory referenced by the lburpEndMessage parameter.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

To obtain the results of the operation, call the `ldap_result` function with the returned message ID.

See Also

[ldap_lburp_end_request](#) (page 337)

ldap_parse_lburp_start_response

Parses LBURP start response data when the result code is LDAP_RES_EXTENDED.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>int ldap_parse_lburp_start_response(  
LDAP      *ld,                LDAPMessage *lburpStartMessage,  
int *resultCode,              char      **errorMsg,                int  
*transactionSize,            int freeIt                          );
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

lburpStartMessage

(IN) Pointer to the LDAPMessage returned by [ldap_result \(page 231\)](#).

resultCode

(OUT) Returns the resultCode from the server.

errorMsg

(OUT) Returns the error message from the server, may be NULL if no error messages are requested. This memory must be freed using [ldap_memfree \(page 178\)](#).

transactionSize

(OUT) Returns the LBURP transaction size.

freeIt

(IN) If non-zero, the function will free the memory referenced by the lburpStartMessage parameter.

Return Values

0x00	LDAP_SUCCESS
Non-zero	Failure. For a complete list, see “ LDAP Return Codes ”.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY

Remarks

The transactionSize is a hint sent by the server to tell the client the number of update operations per UpdateOperation that it would like the client to send. The client must not send more update operations in a single Update Operation than the value in transactionSize.

See Also

[ldap_lburp_start_request \(page 341\)](#)

ldap_parse_monitor_events_response

Parses event data when the result code is LDAP_RES_INTERMEDIATE. This result code indicates that an event has occurred. Event monitoring works with eDirectory 8.7 or higher.

LDAP Version: v2 or higher

Library: *ldapsdk.*

NDS Version: 8.7 or higher

Platform: NLM, Windows (NT, 95, 98, 2000), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap.h>

int ldap_parse_monitor_events_response (
    LDAP          *ld,
    LDAPMessage    *eventMessage,
    int            *resultCode,
    char           **errorMessage,
    int            *badEventsCount,
    EVT_EventSpecifier **badEvents,
    int            freeIt);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

eventMessage

(IN) Pointer to the LDAPMessage returned by [ldap_result \(page 231\)](#).

resultCode

(OUT) Returns the resultCode from the server.

errorMessage

(OUT) Returns the error message from the server, may be NULL if no error messages are requested. This memory must be freed using [ldap_memfree \(page 178\)](#).

badEventsCount

(OUT) Returns the number of bad events from the server, may be NULL if no data is requested. This memory must be freed using [ldapx_memfree \(page 386\)](#).

badEvents

(OUT) If the value of resultCode is LDAP_PROTOCOL_ERROR, this parameter receives an array of [EVT_EventSpecifier](#) structures identifying the unrecognized events (free with [ldap_event_free \(page 328\)](#)). Otherwise, the parameter is set to NULL.

freeIt

(IN) If non-zero, the function will free the memory referenced by the eventMessage parameter.

Return Values

LDAP_SUCCESS	Request was successfully sent
[Other value]	See LDAP Result Codes for information

See Also

[ldap_monitor_events](#) (page 346), [ldap_parse_ds_event](#) (page 350), [ldap_event_free](#) (page 328)

ldap_partition_entry_count

Returns the number of entries in the specified partition.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_partition_entry_count (
    LDAP          *ld,
    char          *dn,
    unsigned long  *count);
```

Parameters

- ld**
(IN) Points to the handle for the LDAP session.
- dn**
(IN) Points to the distinguished name, in LDAP format, of an entry in the partition whose entries are to be counted.
- count**
(OUT) Points to the address where the count is returned.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “LDAP Return Codes”.

Remarks

This function stops at the boundary of the partition. It does not cross the boundary and count the entries in child partitions.

If this function is called immediately after creating a new partition, the count will be inaccurate until the partition moves from the new state (LDAP_RS_NEW_REPLICA) to the on state (LDAP_RS_ON).

For sample code, see [getcount.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.13) and the requestValue is a BER encoding of the following:

```
RequestBer
  dn      LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.14) and the responsetValue is a BER encoding of the following:

```
ResponseBer
  count   INTEGER
```

See Also

[ldap_get_replica_info \(page 335\)](#)

ldap_nds_to_ldap

Converts a typeless, distinguished eDirectory name into LDAP format.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_nds_to_ldap (
    LDAP          *ld,
    unsigned short *ndsName,
    char          **ldapName);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

ndsName

(IN) Points to the eDirectory distinguished name in typeless, dotted format that includes the tree name (for example: ksmith.provo.novell.novell_inc). The string must be a Unicode string. If the object is in a DNS rooted tree, com.dns must be included as the tree name (for example: ksmith.provo.novell.com.dns).

ldapName

(OUT) Points to the entry's distinguished name in LDAP format, for example, "cn=ksmith, ou=provo, o=novell"

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See " LDAP Return Codes ".

Remarks

This function is provided for legacy eDirectory applications and utilities that expect the entry names to be entered in eDirectory typeless, dotted format.

It is provided for convenience, but should be used sparingly or the application's performance will be affected. This is not a local function. The function makes a call to the LDAP server to find the types.

The function allocates memory for the ldapName parameter, and the caller is responsible for freeing it with the ldapx_memfree function.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.1) and the requestValue is a BER encoding of the following:

```
RequestBer
  dn          LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.2) and the responsetValue is a BER encoding of the following:.

```
ResponseBer
  ldapName    OCTET STRING
```


ldap_receive_all_updates

Requests that a specified replica on a specified server receive all updates.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_receive_all_updates (
    LDAP      *ld,
    char      *partitionRoot,
    char      *toServerDN,
    char      *fromServerDN);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

partitionRoot

(IN) Points to the distinguished name, in LDAP format, of the replica that receives the updates.

toServerDN

(IN) Points to the distinguished name, in LDAP format, of the server holding the replica to be updated.

fromServerDN

(IN) Points to distinguished name, in LDAP format, of the server from which the updates are sent. Not currently used.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

A replica's distinguished name is the distinguished name of the replica's root container, called the partition root in eDirectory.

In NDS 7.x and above, updates can come from any server that holds a replica of the partition; therefore, eDirectory does not currently use the fromServerDN parameter to specify which server should send the updates.

Each ld is associated with a particular server. eDirectory uses the ld rather than the toServerDN parameter to specify the server with the replica that needs updating.

For sample code, see [recvupd.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.21) and the requestValue is a BER encoding of the following:

```
RequestBer
  partitionRoot    LDAPDN
  toServerDN       LDAPDN
  fromServerDN     LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.22) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_send_all_updates \(page 376\)](#)

ldap_refresh_server

Restarts the LDAP server associated with the specified session handle.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_refresh_server (
    LDAP      *ld);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x32	LDAP_INSUFFICIENT_ACCESS
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

Each ld is associated with a particular LDAP server. By specifying the ld, you specify the LDAP server that is restarted, and you use the authentication credentials established for that ld.

To restart the LDAP server, the client must have write permissions to the extensionInfo attribute of the LDAP server object.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.9) and the requestValue has no value:

```
RequestBer
    NULL
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.10) and the responseValue has no value:.

ResponseBer
NULL

ldap_remove_orphan_partition

Removes the specified orphan partition from the specified server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_remove_orphan_partition (
    LDAP      *ld,
    char      *serverDN,
    char      *contextName);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name of the server holding the orphan naming context to remove.

contextName

(IN) Points to the distinguished name of the orphan partition (naming context) to remove.

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This function fails if the server does not hold the specified partition.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.41) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverDN      LDAPDN
  contextName   LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.42) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_create_orphan_partition](#) (page 326)

[ldap_abort_partition_operation](#) (page 316)

ldap_remove_replica

Removes a replica from the specified directory server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_remove_replica (
    LDAP      *ld,
    char      *dn,
    char      *serverDN,
    int       flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name, in LDAP format, of the replica to be removed.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server holding the replica to be removed.

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED

Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.
----------	-----------------------------------------------------------------------------

Remarks

A replica's distinguished name is the distinguished name of the replica's root container, called the partition root in eDirectory.

The caller must have supervisor rights to the master replica.

This function can remove all the replicas of a partition except the master replica. If the master replica is the only replica left, it is removed by merging the child partition with its parent partition.

For sample code, see [remrepl.c](#) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.11) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  serverName LDAPDN
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.12) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_add_replica](#) (page 318)

ldap_request_partition_sync

Schedules synchronization of the specified partition after the specified delay.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_request_partition_sync (
    LDAP      *ld,
    char      *serverDN,
    char      *partitionRoot,
    int        delay);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to distinguished name, in LDAP format, of the server containing the partition.

partitionRoot

(IN) Points to the distinguished name, in LDAP format, of the partition root to synchronize.

delay

(IN) Specifies the time, in seconds, to delay before synchronization starts.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

In eDirectory, this function causes the server to examine its Transitive Vector attribute, and the server initiates synchronization with those servers whose replica update time is older than the local time stamp.

For sample code, see [parsync.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.25) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverName      LDAPDN
  partitionRoot   LDAPDN
  delay           INTEGER
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.26) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_receive_all_updates \(page 362\)](#)

[ldap_send_all_updates \(page 376\)](#)

ldap_request_schema_sync

Schedules synchronization of the schema after the specified delay.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_request_schema_sync (
    LDAP      *ld,
    char      *serverDN,
    int        delay);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name, in LDAP format, of the server.

delay

(IN) Specifies the time, in seconds, to delay before synchronization starts.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

For sample code, see [schsync.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.27) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverName      LDAPDN
  delay           INTEGER
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.28) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_request_partition_sync \(page 370\)](#)

ldap_restore_object

Restores an object's attribute name and values that were saved by calling [ldap_backup_object](#) (page 320).

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.8 or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

ldap_restore_object (
    LDAP *ld,
    const char *dn,
    const char *passwd,
    char *objectInfo,
    char *chunkSize,
    int size);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Points to the distinguished name of the object for which information is to be restored.

passwd

(IN) Points to the password for encryption and decryption, when any one of the attributes in the user object has been encrypted. If password is supplied then the connection to the servers will be over TLS.

objectInfo

(OUT) Points to the requested attribute names and values.

chunkSize

(OUT) Specifies the length of each chunk.

size

(OUT) Specifies the length of the information to be restored.

Return Values

LDAP_PARAM_ERROR

LDAP_NO_MEMORY
LDAP_ENCODING_ERROR
LDAP_NOT_SUPPORTED
LDAP_SUCCESS

See Also

[ldap_backup_object](#) (page 320)

ldap_send_all_updates

Requests that a specified server send all updates to the replica ring.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_send_all_updates (
    LDAP      *ld,
    char      *partitionRoot,
    char      *origServerDN);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

partitionRoot

(IN) Points to the distinguished name, in LDAP format, of the replica that contains the updates.

origServerDN

(IN) Points to the distinguished name, in LDAP format, of the server that sends the updates.
Not currently used.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “LDAP Return Codes” .

Remarks

A replica's distinguished name is the distinguished name of the replica's root container, called the partition root in eDirectory.

In NDS 7.x and higher, any server containing a replica can send updates. Since each ld has a server associated with it, NDS uses the ld to specify the originating server rather than the origServerDN parameter.

For sample code, see [sendupd.c \(http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm\)](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.23) and the requestValue is a BER encoding of the following:

```
RequestBer
  partitionRoot      LDAPDN
  origServerDN       LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.24) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_receive_all_updates \(page 362\)](#)

ldap_set_replication_filter

Sets the attribute and class filter for an eDirectory filtered replica.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5, SP1

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_set_replication_filter (
    LDAP      *ld,
    char      *serverDN,
    char      *filter);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name of the server that holds the filtered replicas.

filter

(IN) Points to the filter that identifies the classes and attributes that are allowed in filtered replicas on the server. For the format, see [Section 6.6, “Replication Filters,” on page 411](#).

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “LDAP Return Codes” .

Remarks

An eDirectory server can only have one replication filter, and that filter is used for all filtered replicas on the server.

NDS eDirectory 8.5 and above supports filtered replicas. The extension to get and set the replication filter is not available until NDS 8.5 SP1.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.35) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverName      LDAPDN
  SEQUENCE of SEQUENCE {
    classname      OCTET STRING
    SEQUENCE of ATTRIBUTES
  }
```

where

```
ATTRIBUTES:: OCTET STRING
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.36) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_get_replication_filter \(page 333\)](#)

ldap_split_orphan_partition

Splits the specified orphan partition from the specified server.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_split_orphan_partition (
    LDAP      *ld,
    char      *serverDN,
    char      *contextName);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

serverDN

(IN) Points to the distinguished name of the server holding the orphan partition.

contextName

(IN) Points to the distinguished name of the orphan partition (naming context) to split.

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This function fails if the server does not hold the specified partition.

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.41) and the requestValue is a BER encoding of the following:

```
RequestBer
  serverDN      LDAPDN
  contextName   LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.42) and the responseValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_create_orphan_partition](#) (page 326)

[ldap_abort_partition_operation](#) (page 316)

ldap_split_partition

Splits a partition.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

int ldap_split_partition (
    LDAP      *ld,
    char      *dn,
    int       flags);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

dn

(IN) Specifies the distinguished name, in LDAP format, of the new root partition.

flags

(IN) Specifies whether all the servers in the replica ring must be up before proceeding. When set to zero, the status of the servers is not checked. When set to LDAP_ENSURE_SERVERS_UP, all the servers must be up for the operation to proceed.

Return Values

0x00	LDAP_SUCCESS
0x01	LDAP_OPERATIONS_ERROR: A string is returned with this error code that indicates the source of the error.
0x53	LDAP_ENCODING_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

This operation is performed on the server containing the master replica of the parent partition. The caller must have supervisor rights to the child's master replica and the parent's master replica.

For sample code, see [splitpart.c](http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm) (http://developer.novell.com/ndk/doc/samplecode/cldap_sample/index.htm).

Packet Format

The requestName is set to the OID (2.16.840.1.113719.1.27.100.5) and the requestValue is a BER encoding of the following:

```
RequestBer
  flags      INTEGER
  dn         LDAPDN
```

The responseName is set to the OID (2.16.840.1.113719.1.27.100.6) and the requestValue has no value.

```
ResponseBer
  NULL
```

See Also

[ldap_abort_partition_operation](#) (page 316)

ldap_trigger_back_process

Triggers a background process.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5, SP1

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

LIBLDAP_F(int) ldap_trigger_back_process (
    LDAP      *ld,
    int       processID);
```

Parameters

ld

(IN) Points to the handle for the LDAP session.

processID

(IN) Flag determining the background process to trigger. This flag will be one of the following:

LDAP_BK_PROCESS_BKLINKER 1
LDAP_BK_PROCESS_JANITOR 2
LDAP_BK_PROCESS_LIMBER 3
LDAP_BK_PROCESS_SKULKER 4
LDAP_BK_PROCESS_SCHEMA_SYNC 5
LDAP_BK_PROCESS_PART_PURGE 6

Return Values

0x00	LDAP_SUCCESS
0x53	LDAP_ENCODING_ERROR
0x59	LDAP_PARAM_ERROR
0x5A	LDAP_NO_MEMORY
0x5C	LDAP_NOT_SUPPORTED
Non-zero	Non-zero values indicate errors. See “ LDAP Return Codes ”.

Remarks

eDirectory background processes run automatically at periodic intervals to keep different replicas in an eDirectory tree synchronized. On rare occasions, it may be desirable to initiate one or more of these processes manually rather than waiting for the next scheduled execution.

Back Linker (Reference Checker). Keeps the "backlink" attribute of objects synchronized between servers. This attribute keeps track of external references to the object and also maintains DRLs (Dynamic Reference Links, "Used By" attribute).

Janitor. Cleans up bindery information, refreshes server status, and updates inherited ACLs.

Limber. Maintains tree connectivity, making sure tree names are consistent among servers. Also caches information from the NCPServer object and index information. After creating a new index, an application may want to kick off the limber process to cause it to start creating the index immediately.

Skulker. Replicates and synchronizes directory data among replicas. A developer may want to initiate this process to start data synchronization immediately.

Schema Sync. Replicates and synchronizes the schema.

Partition Purge. Purges deleted entries and deleted values that have been synchronized with all replicas.

ldapx_memfree

Frees memory allocated by the LDAP extension library.

LDAP Version: v3

Library: *ldapx.*

NDS Version: 8.5

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldapx.h>

void ldapx_memfree (
    void *mem);
```

Parameters

mem

(IN) Points to the memory to free.

Remarks

This function should be used to free any memory allocated by the LDAP extension library.

The request and reply packets have the following formats.

See Also

[ldap_get_bind_dn](#) (page 329)

[ldap_list_replicas](#) (page 342)

[ldap_nds_to_ldap](#) (page 360)

Directory data in the LDAPv3 API is sent and received in UTF-8 format. For a discussion of the relationship between UTF-8, local, multi-byte, wide character, and unicode, see [Section 1.8, “Character Conversions,” on page 44](#).

To simplify the use of UTF-8 character sets, the LDAP SDK contains functions to provide developers a standard, cross-platform method to work with UTF-8 strings.

Functions to convert data between wide character and UTF-8 formats are grouped in the following categories:

- [Section 5.1, “UTF-8 / Wide Character Conversions,” on page 387](#).

In addition, the LDAP SDK contains a number of utility functions for working with UTF-8 strings. They are contained in the following category:

- [Section 5.2, “UTF-8 Utility Functions,” on page 393](#).

5.1 UTF-8 / Wide Character Conversions

The UTF-8 / wide conversion routines follow the pattern of the ANSI C conversion routines. These routines use the `wchar_t` type, which is two bytes on some systems and four bytes on others. The advantage to using the `wchar_t` type is that all the standard wide-character functions may be used on these strings, such as `wcslen`, `wscat`, etc.

ldap_x_utf8_to_wc

Convert a single UTF-8 encoded character to a wide character.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_utf8_to_wc (
    wchar_t      *wchar,
    const char   *utf8char);
```

Parameters

wchar

(OUT) Points to a wide character to receive the converted character code.

utf8char

(IN) Address of the UTF8 sequence of bytes.

Return Values

If successful, the function returns the length in bytes of the UTF-8 input character.

If utf8char is NULL or points to an empty string, the function returns 1 and a NULL is written to wchar.

If utf8char contains an invalid UTF-8 sequence -1 is returned.

Example

```
char utchr_in[] = { 0xE2U, 0x98U, 0xA0U };
wchar_t wc_out;
int n;

/* Convert a UTF-8 character to a wide character.
   Returns wc_out = 0x2620.
   Returns n = 3.   (Byte length of utchr_in)
*/
n = ldap_x_utf8_to_wc(&wc_out, utchr_in);
```

ldap_x_utf8s_to_wcs

Convert a UTF-8 string to a wide character string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_utf8s_to_wcs (
    wchar_t      *wcstr,
    const char    *utf8str,
    size_t        count);
```

Parameters

wcstr

(OUT) Points to a wide char buffer to receive the converted wide char string. The output string will be null-terminated if there is space for it in the buffer.

utf8char

(IN) Address of the null-terminated UTF-8 string to convert.

count

(IN) The number of UTF-8 characters to convert, or equivalently, the size of the output buffer in wide characters.

Return Values

If successful, the function returns the number of wide characters written to wcstr, excluding the null termination character, if any.

If wcstr is NULL, the function returns the number of wide characters required to contain the converted string, excluding the null-termination character.

If an invalid UTF-8 sequence is encountered, the function returns -1.

If the return value equals count, there was not enough space to fit the string and the null terminator in the buffer. As much of the string as will fit is written to the buffer, but it is not null-terminated.

Example

```
#define WCSTRLEN    10
char utstr_in[] = { 0xE2U, 0x98U, 0xA0U, 'a', 'b', 'c', 0 };
wchar_t wcstr_out[WCSTRLEN];
int n;
/* Convert a UTF-8 string to a wide char string.
   Returns wcstr_out = { 0x2620, 'a', 'b', 'c', 0 }
   Returns n = 4.  ( # wide chars written, excl null )
```

```
    If n==WCSTRLEN, the string could not completely fit in the given
    buffer.
    */
    n = ldap_x_utf8s_to_wcs(wcstr_out, utstr_in, WCSTRLEN);
```

ldap_x_wc_to_utf8

Convert a single wide character to a UTF-8 sequence.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_wc_to_utf8 (
    char      *utf8char,
    wchar_t   wchar,
    size_t     count);
```

Parameters

utf8char

(OUT) Points to a byte array to receive the converted UTF-8 string.

wchar

(IN) The wide character to be converted.

count

(IN) The maximum number of bytes to write to the output buffer. Normally set this to LDAP_MAX_UTF8_LEN, which is defined as 3 or 6 depending on the size of wchar_t. A partial character will not be written.

Return Values

If successful, the function returns the length in bytes of the converted UTF-8 output character.

If wchar is NULL, the function returns 1 and 0 is written to utf8char.

If wchar cannot be converted to a UTF-8 character, the function returns -1.

If the converted character will not fit in count bytes, 0 is returned.

Example

```
wchar_t wc_in = 0x2620;
char utchr_out[LDAP_MAX_UTF8_LEN];    /* Either 3 or 6 bytes */
int n;

/* Convert a wide character to a UTF-8 character.
   Returns utchr_out[] = { 0xE2, 0x98, 0xA0 }
   Returns n = 3.   (Byte length of utchr_out)
*/
n = ldap_x_wc_to_utf8(utchr_out, wc_in, LDAP_MAX_UTF8_LEN);
```

ldap_x_wcs_to_utf8s

Convert a wide character string to a UTF-8 string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_wcs_to_utf8s (
    char            *utf8str,
    const wchar_t   *wcstr,
    size_t          count);
```

Parameters

utf8str

(OUT) Points to a byte array to receive the converted UTF-8 string. The output string will be null terminated if there is space for it in the buffer.

wcstr

(IN) Address of the null-terminated wide char string to convert.

count

(IN) The size of the output buffer in bytes.

Return Values

If successful, the function returns the number of bytes written to utf8str, excluding the null termination character, if any.

If utf8str is NULL, the function returns the number of bytes required to contain the converted string, excluding the null-termination character. The count parameter is ignored.

If the function encounters a wide character that cannot be mapped to a UTF-8 sequence, the function returns -1.

If the return value equals count, there was not enough space to fit the string and the null terminator in the buffer. As much of the string as will fit is written to the buffer, but it is not null-terminated. A partial character will not be written.

Example

```
#define UTFSTRLEN    20
wchar_t wcstr_in[] = { 0x2620, 'a', 'b', 'c', 0 };
char utstr_out[UTFSTRLEN];
int n;

/* Convert a wide char string to a UTF-8 string.
```

```

Returns utstr = { 0xE2, 0x98, 0xA0, 'a', 'b', 'c', 0 }
Returns n = 6. ( # bytes written, excl null )
If n==UTFSTRLEN, the string could not completely fit in the given
buffer.
*/
n = ldap_x_wcs_to_utf8s(utstr_out, wcstr_in, UTFSTRLEN);

```

5.2 UTF-8 Utility Functions

The LDAP SDK contains a number of utility functions for working with UTF-8 strings. They are as follows:

- [ldap_x_utf8_chars \(page 394\)](#) Return the number of UTF-8 characters (not bytes) in a null-terminated UTF-8 string.
- [ldap_x_utf8_charlen \(page 395\)](#) Return the number of bytes in a UTF-8 character.
- [ldap_x_utf8_next \(page 397\)](#) Find the next character in a UTF-8 string.
- [ldap_x_utf8_prev \(page 398\)](#) Find the previous character in a UTF-8 string.
- [ldap_x_utf8_copy \(page 399\)](#) Copy one UTF-8 character.
- [ldap_x_utf8_strchr \(page 400\)](#) Find a character in a string.
- [ldap_x_utf8_strspn \(page 401\)](#) Find the first substring.
- [ldap_x_utf8_strcspn \(page 402\)](#) Find a substring in a string.
- [ldap_x_utf8_strpbrk \(page 403\)](#) Find first occurrence of a character from one string in another string.
- [ldap_x_utf8_strtok \(page 404\)](#) Find next token in string.

ldap_x_utf8_chars

Return the number of UTF-8 characters (not bytes) in a null-terminated UTF-8 string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

ber_len_t ldap_x_utf8_chars (
    const char *p);
```

Parameters

p

(IN) Contains the null-terminated UTF-8 string to count.

Return Values

Number of chars (not bytes) in p.

Example

```
/* String with 4 UTF-8 characters */
char utstr[] = { 0xe2U, 0x98U, 0xa0U, 'a', 'b', 'c', 0 };
int n = ldap_x_utf8_chars(utstr);    /* Returns 4 */
```

ldap_x_utf8_charlen

Return the number of bytes in a UTF-8 character.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_utf8_charlen (
    const char *p);
```

Parameters

p

(IN) Points to the UTF-8 character.

Return Values

Returns length in bytes of the UTF-8 character. (1-6). 0 is returned for an invalid character.

Remarks

The length of the character is determined by looking only at the first byte of the UTF-8 character.

Example

```
/* String starts with a 3-byte UTF-8 character. */
char utstr[] = { 0xe2U, 0x98U, 0xa0U, 'a', 'b', 'c', 0 };
int n = ldap_x_utf8_charlen(utstr);    /* Returns 3 */
n = ldap_x_utf8_charlen(utstr+4);    /* 'b' char. Returns 1 */
```

ldap_x_utf8_charlen2

Return the number of bytes in a UTF-8 character, while catching shortest possible illegal UTF-8 encoding.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_utf8_charlen2 (
    const char *p);
```

Parameters

p

(IN) Points to the UTF-8 character.

Return Values

Returns length in bytes of the UTF-8 character. (1-6). 0 is returned for an invalid character.

Remarks

The length of the character is determined by looking only at the first byte of the UTF-8 character.

ldap_x_utf8_next

Find the next character in a UTF-8 string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

char* ldap_x_utf8_next (
    const char *p);
```

Parameters

p

(IN) Points to a UTF-8 string.

Return Values

Returns the address of the next UTF-8 character in the string.

Remarks

The function will step over NULLs just like any other character. The application must take care not to step beyond the end of the string.

Example

```
/* String starts with a 3-byte UTF-8 character. */
char utstr[] = { 0xe2U, 0x98U, 0xa0U, 'a', 'b', 'c', 0 };
char *p = ldap_x_utf8_next(utstr); /* p now points to the 'a' char */
p = ldap_x_utf8_next(p); /* p now points to the 'b' char */
```

ldap_x_utf8_prev

Find the previous character in a UTF-8 string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

char* ldap_x_utf8_prev (
    const char *p);
```

Parameters

p

(IN) Points to a UTF-8 string.

Return Values

Returns a pointer to the previous character in the string.

Remarks

The application must take care not to step beyond the beginning of the string.

Example

```
char utstr[] = { 0xe2U, 0x98U, 0xa0U, 'a', 'b', 'c', 0 };
char *p = ldap_x_utf8_prev(utstr+4);    /* p now points to the 'a' char
*/
p = ldap_x_utf8_prev(p);    /* p now points to the beginning char */
```

ldap_x_utf8_copy

Copy one UTF-8 character.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

int ldap_x_utf8_copy (
    char      *dst,
    const char *src);
```

Parameters

dst

(IN) Points to the output buffer.

src

(IN) Points to the UTF-8 character to copy.

Return Value

Number of bytes copied.

Example

```
char utstr[] = { 0xe2U, 0x98U, 0xa0U, 'a', 'b', 'c', 0 };
char dest[3];
int n = ldap_x_utf8_copy(dest, utstr); /* Copies 1st char. Returns 3.
*/
```

ldap_x_utf8_strchr

Find a character in a string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

char * ldap_x_utf8_strchr (
    const char *str,
    const char *chr);
```

Parameters

str

(IN) Null-terminated UTF-8 string to search.

chr

(IN) Points to the UTF-8 character to be located.

Return Values

Returns the first occurrence of chr in str, or NULL if not found.

Example

```
char utstr[] = { 'a', 'b', 0xe2U, 0x98U, 0xa0U, 'x', 'y', 0 };
char chr[] = { 0xe2U, 0x98U, 0xa0U };
char *p = ldap_x_utf8_strchr(utstr, chr);    /* Returns utstr+2 */
```

ldap_x_utf8_strspn

Find the first substring.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

ber_len_t ldap_x_utf8_strspn (
    const char *str,
    const char *set);
```

Parameters

str

(IN) Null-terminated UTF-8 string to search.

set

(IN) Null-terminated character set.

Return Values

Returns the length of the substring in str that consists entirely of characters in set.

Remarks

This function returns the number of bytes, not characters.

Example

```
char utstr[] = { 'a', 'b', 0xe2U, 0x98U, 0xa0U, 'x', 'y', 0 };
char set[] = { 'b', 0xe2U, 0x98U, 0xa0U, 'a', 0 };
int n = ldap_x_utf8_strspn(utstr, set); /* Returns 5 */
```


ldap_x_utf8_strcspn

Find a substring in a string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

ber_len_t ldap_x_utf8_strcspn (
    const char *str,
    const char *set);
```

Parameters

str

(IN) Null-terminated UTF-8 string to search.

set

(IN) Null-terminated character set.

Return Values

Returns the length of the initial segment of str that consists entirely of characters not in set.

Remarks

This function returns the number of bytes, not characters.

Example

```
char utstr[] = { 'a', 'b', 0xe2U, 0x98U, 0xa0U, 'x', 'y', 0 };
char set[] = { 'x', 0xe2U, 0x98U, 0xa0U, 0 };
int n = ldap_x_utf8_strcspn(utstr, set);    /* Returns 2 */
```

ldap_x_utf8_strpbrk

Find first occurrence of a character from one string in another string.

Library: *ldapsdk.*

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

char * ldap_x_utf8_strpbrk (
    const char *str,
    const char *set);
```

Parameters

str

(IN) Null-terminated UTF-8 string to search.

set

(IN) Null-terminated character set.

Return Values

Returns a pointer to the first occurrence of any character from set in str.

Example

```
char utstr[] = { 'a', 'b', 0xe2U, 0x98U, 0xa0U, 'x', 'y', 0 };
char set[] = { 'x', 0xe2U, 0x98U, 0xa0U, 0 };
char *p = ldap_x_utf8_strpbrk(utstr, set);    /* Returns utstr+2 */
```

ldap_x_utf8_strtok

Find next token in string.

Library: *ldapsdk.*

NDS Version: 7.xx or higher

Platform: NLM, Windows (NT, 95, 98, 2000, XP), Solaris, Linux, AIX, HP-UX

Syntax

```
#include <ldap_utf8.h>

char * ldap_x_utf8_strtok (
    char      *str,
    const char *sep,
    char      **last);
```

Parameters

str

(IN) UTF-8 string to parse. On subsequent calls to this function this parameter should be NULL.

sep

(IN) Set of separator characters.

last

(IN/OUT) After each function call, returns a pointer to the token following the separator character. This value should be passed in to the next function call.

Return Values

Returns a pointer to the next token found in str.

Remarks

Parses a string into tokens based on a set of delimiters. When a delimiter is encountered, it is replaced by a NULL, allowing the token to be processed as a null-terminated string.

On the first call, a value is returned in last. On subsequent calls, NULL should be given for str. Last is updated after each call.

Example

```
char utstr[] = { 'a', 0xe2U, 0x98U, 0xa0U, 'b', 0xe2U, 0x98U, 0xa0U,
                 'c', 0 };
char delims[] = { 0xe2U, 0x98U, 0xa0U, 0 };
char *last;
char *p;
```

```
p = ldap_x_utf8_strtok(utstr, delims, &last);    /* p points to "a" */  
p = ldap_x_utf8_strtok(NULL, delims, &last);    /* p points to "b" */  
p = ldap_x_utf8_strtok(NULL, delims, &last);    /* p points to "c" */
```


This chapter defines the flags used by the LDAP functions.

6.1 Object Access Control Rights

Table 6-1 *Object Rights*

Flag Name	C Value	Description
DS_ENTRY_BROWSE	0x00000001L	Allows a trustee to discover objects in the eDirectory tree.
DS_ENTRY_ADD	0x00000002L	Allows a trustee to create child objects (new objects that are subordinate to the object in the tree).
DS_ENTRY_DELETE	0x00000004L	Allows a trustee to delete an object. This right does not allow a trustee to delete a container object that has subordinate objects.
DS_ENTRY_RENAME	0x00000008L	Allows a trustee to rename the object.
DS_ENTRY_SUPERVISOR	0x00000010L	Gives a trustee all rights to an object and its attributes.
DS_ENTRY_INHERIT_CTL	0x00000040L	Allows a trustee to inherit the rights granted in the ACL and exercise them on subordinate objects.

6.2 Attribute Access Control Rights

Table 6-2 *Attribute Rights*

Flag Name	C Value	Description
LDAP_DS_ATTR_COMPARE	0x00000001L	Allows a trustee to compare a value with an attribute's value. This allows the trustee to see if the attribute contains the value without having rights to see the value.
LDAP_DS_ATTR_READ	0x00000002L	Allows a trustee to read an attribute value. This right confers the Compare right.
LDAP_DS_ATTR_WRITE	0x00000004L	Allows a trustee to add, delete, or modify an attribute value. This right also gives the trustee the Self (Add or Delete Self) right.
LDAP_DS_ATTR_SELF	0x00000008L	Allows a trustee to add or delete its name as an attribute value on those attributes that take object names as their values.

Flag Name	C Value	Description
LDAP_DS_ATTR_SUPERVISOR	0x00000020L	Gives a trustee all rights to the object's attributes.
LDAP_DS_ATTR_INHERIT_CTL	0x00000040L	Allows a trustee to inherit the rights granted in the ACL and exercise these attribute rights on subordinate objects.

6.3 Certificate Attribute IDs

Table 6-3 *Certificate Attribute IDs*

Attribute ID	Data Type	Description
LDAPSSL_CERT_ATTR_ISSUER	char *	A pointer to a character array containing the certificate issuer name. The issuer is the distinguished name of the certificate authority that issued the certificate. The length returned is the length of the string not including the NULL termination character.
LDAPSSL_CERT_ATTR_SUBJECT	char *	A pointer to a character array containing the certificate subject name. The subject is the distinguished name of the entity that owns the certificate. The length returned is the length of the string not including the NULL termination character.
LDAPSSL_CERT_ATTR_VALIDITY_PERIOD	LDAPSSL_Cert_VValidity_Period *	A pointer to a LDAPSSL_Cert_VValidity_Period structure. The validity period structure contains a not valid after and a not valid before timestamp which defines the period during which the certificate should be considered valid. The timestamps can be a universal time string or a generalized time string (see LDAPSSL_Cert_VValidity_Period (page 470)).
LDAPSSL_CERT_GET_STATUS	int *	<p>The certificate status codes are described in Section 6.12, "SSL Certificate Status Codes," on page 424</p> <p>The cert status is one of sixteen certificate status codes indicating the status of an untrusted SSL certificate.</p>

6.4 Inheritance Control Rights

The bit settings for the Inheritance Control rights use values that ensure compatibility with NetWare 4.x.

Table 6-4 *Inheritance Control Settings*

NetWare Version	Object Right DS_ENTRY_INHERIT_CTL	[All Attributes Rights] DS_ATTR_INHERIT_CTL	Specific Attribute DS_ATTR_INHERIT_CTL
NetWare 4.x	NetWare 4.x does not support this functionality. Inheritance of object rights is always supported. NetWare 4.x requires this bit to be set to 0.	NetWare 4.x does not support this functionality. Inheritance of rights to [All Attributes Rights] is always supported. NetWare 4.x requires this bit to be set to 0.	NetWare 4.x does not support this functionality. Inheritance of ACLs to specific attributes is always blocked. NetWare 4.x requires this bit to be set to 0.
NetWare 5.x	NetWare 5.x supports this right. Set this bit to 0 (zero) to allow the inheritance of the rights in the ACL. Set this bit to 1 (one) to block the inheritance of the ACL rights.	NetWare 5.x supports this right. Set this bit to 0 (zero) to allow the inheritance of the rights granted to [All Attributes Rights]. Set this bit to 1 (one) to block the inheritance of the ACL rights.	NetWare 5.x supports this right. Set this bit to 1 (one) to allow the inheritance of the rights granted to the specific attribute. Set this bit to 0 to block the inheritance of the ACL rights.

6.5 Replica States

The replica states indicate the current state of the replica. For more information, see “[Replica Transition States](#)” (*Developer Kit*).

NOTE: These values are not continuous.

Table 6-5 *Replica States*

Flag Name	C Value	Meaning
LDAP_RS_ON	0	Indicates that the replica is fully functioning and capable of responding to NDS™ requests.
LDAP_RS_NEW_REPLICA	1	Indicates that a new replica has been added but has not received a full download of information from <ul style="list-style-type: none"> • The master replica if NDS 6.x and lower • Another replica if NDS 7.x and higher
LDAP_RS_DYING_REPLICA	2	Indicates that a replica of the partition is being deleted. In NDS 6.x and lower, the replica stays in this state until it synchronizes with another replica. In NDS 7.x and higher, indicates that the request has been received.

Flag Name	C Value	Meaning
LDAP_RS_LOCKED	3	Indicates that the replica is locked. The move partition operation uses this state to lock the parent partition of the child partition that is moving.
LDAP_RS_TRANSITION_ON	6	Indicates that a new replica has finished receiving its download from the master replica and is now receiving synchronization updates from the other replicas. Used only in NDS 6.x and lower.
LDAP_RS_DEAD_REPLICA	7	Indicates that the dying replica needs to synchronize with another replica before being converted to an external reference, if a root replica, or to a subordinate reference, if a nonroot replica. Used only in NDS 7.x and higher.
LDAP_RS_BEGIN_ADD	8	Indicates that subordinate references of the new replica are being added. Used only in NDS 7.x and higher.
LDAP_RS_MASTER_START	11	Indicates that a partition is receiving a new master replica. The replica that will be the new master replica is set to this state.
LDAP_RS_MASTER_DONE	12	Indicates that a partition has a new master replica. When the new master is set to this state, it knows it is now the master and changes its replica type to master and the old master to Read/Write.
LDAP_RS_SS_0	48	Indicates that a partition is going to split into two partitions. In this state, other replicas of the partition are informed of the pending split operation.
LDAP_RS_SS_1	49	Indicates that the split partition operation has started. When the split is finished, the state will change to RS_ON.
LDAP_RS_JS_0	64	Indicates that two partitions are in the process of joining into one partition. In this state, the replicas that are affected are informed of the join operation. The master replica of the parent and child partitions are first set to this state and then all the replicas of the parent and child. New replicas are added where needed.
LDAP_RS_JS_1	65	Indicates that two partitions are in the process of joining into one partition. This state indicates that the join operation is waiting for the new replicas to synchronize and move to the RS_ON state.
LDAP_RS_JS_2	66	Indicates that two partitions are in the process of joining into one partition. This state indicates that all the new replicas are in the RS_ON state and that the rest of the work can be completed.

6.6 Replication Filters

NDS eDirectory 8.5 and above support filtered replicas. Previous versions of eDirectory do not support filtered replicas.

A single replication filter is set for an eDirectory server, and all replicas that reside on that specified server conform to that particular filter. The filter parameter (for the `ldap_get_replication_filter` and `ldap_set_replication_filter` functions) is a UTF string that comprises a sequence of object class names and attribute names delimited by the dollar (\$) sign. The filter follows these rules:

1. Each class name and each attribute name is terminated by a \$ sign.
2. Each sequence of a class with its attribute names is terminated by a \$ sign.
3. The filter is terminated with a \$ sign.

The asterisk character (*) can be used in place of an attribute name to indicate all attributes from a particular class.

The following sample filter selects three attributes from the user class and one attribute from the `groupOfUniqueNames` class for the filter.

```
"user$cn$surname$mail$$groupOfUniqueNames$member$$$"
```

The following sample filter selects all attributes from the user class and one attribute from the `groupOfUniqueNames` class for the filter:

```
"user$*$groupOfUniqueNames$member$$$"
```

A single \$ sign in a filter is used for two special cases:

- It resets the filter.
- It represents the absence of a filter on the server.

6.7 Replica Types

The replica types identify the type of replica and are defined in the `REPLICA_TYPE` typedef enumeration in the `ldapx.h` file. Replica type determines the types of client operations that can be performed on the replica.

Table 6-6 *Replica Types*

Flag Name	C Value	Meaning
LDAP_RT_MASTER	0	Identifies this replica as the master replica of the partition. Entries can be modified; partition operations can be performed.
LDAP_RT_SECONDARY	1	Identifies this replica as a secondary replica of the partition. Secondary replicas are Read/Write replicas and entries can be modified.
LDAP_RT_READONLY	2	Identifies the replica as a Read-Only replica. Only the eDirectory synchronization processes can modify the information on this replica.

Flag Name	C Value	Meaning
LDAP_RT_SUBREF	3	Identifies the replica as a subordinate reference. eDirectory automatically adds these replicas to a server when the server does not contain replicas of all child partitions. Only eDirectory can modify information on this replica.
LDAP_RT_SPARSE_WRITE	4	Identifies the replica as a Read/Write replica with sparse data. It is configured to contain only specified object types and attributes.
LDAP_RT_SPARSE_READ	5	Identifies the replica as a Read-Only replica with sparse data. It is configured to contain only specified object types and attributes.
LDAP_RT_COUNT	6	Identifies the total number of replica types that have been defined.

6.8 Request Message Types

The following table details the types of the request messages that are supported by the LDAP libraries for C.

Table 6-7 *Request Messages Types*

Type	Description
LDAP_REQ_DELETE (0x4A)	Indicates a delete operation.
LDAP_REQ_UNBIND (0x42)	Indicates an unbind operation.
LDAP_REQ_ABANDON (0x50)	Indicates a request to abandon an operation.
LDAP_REQ_BIND (0x60)	Indicates a bind operation.
LDAP_REQ_SEARCH (0x63)	Indicates a search operation.
LDAP_REQ_MODIFY (0x66)	Indicates a modify operation.
LDAP_REQ_ADD (0x68)	Indicates an add operation.
LDAP_REQ_RENAME	See LDAP_REQ_MODRDN.
LDAP_REQ_MODDN	See LDAP_REQ_MODRDN.
LDAP_REQ_MODRDN (0x6C)	Indicates a modify RDN operation.
LDAP_REQ_COMPARE (0x6E)	Indicates a compare operation.
LDAP_REQ_EXTENDED (0x77)	Indicates an extended operation

6.9 Result Message Types

The following table details the types of the result messages that are supported by the LDAP libraries for C.

Table 6-8 *Result Message Type*

Type	Description
LDAP_RES_BIND (0x61)	Indicates that the LDAPMessage structure contains the results of a bind operation.
LDAP_RES_SEARCH_ENTRY (0x64)	Indicates that the LDAPMessage structure contains information about an entry which was found during a search operation.
LDAP_RES_SEARCH_RESULT (0x65)	Indicates that the LDAPMessage structure contains the results of a search operation
LDAP_RES_MODIFY (0x67)	Indicates that the LDAPMessage structure contains the results of a modify operation.
LDAP_RES_ADD ((0x69)	Indicates that the LDAPMessage structure contains the results of an add operation.
LDAP_RES_DELETE (0x6B)	Indicates that the LDAPMessage structure contains the results of a delete operation.
LDAP_RES_RENAME (0x6D)	Indicates that the LDAPMessage structure contains the results of a rename operation.
LDAP_RES_MODDN (0x6D)	Indicates that the LDAPMessage structure contains the results of a modify DN operation.
LDAP_RES_MODRDN (0x6D)	Indicates that the LDAPMessage structure contains the results of a modify RDN operation.
LDAP_RES_COMPARE (0x6F)	Indicates that the LDAPMessage structure contains the results of a compare operation.
LDAP_RES_SEARCH_REFERENCE (0x73)	Indicates that the LDAPMessage structure contains a referral to another LDAP server which was found during a search operation.
LDAP_RES_EXTENDED (0x78)	Indicates that the LDAPMessage structure contains the results of an extended operation

6.10 Session Preference Options

These flags are used by the [ldap_get_option](#) (page 166) and [ldap_set_option](#) (page 266) functions.

Table 6-9 *Session Preference Options*

Option	Value	Description
LDAP_OPT_API_FEATURE_INFO	0x0015	Specifies version information about an LDAP API extended feature. ldap_set_option data type: Not supported; ldap_get_option data type: LDAPAPIFeatureInfo *

Option	Value	Description
LDAP_OPT_API_INFO	0x0000	<p>Retrieves basic information about the API implementation. It cannot be used to set information.</p> <p>It includes the API version, minimum LDAP version, maximum LDAP version, vendor name, and vendor version. If the ldap_get_option function returns</p> <ul style="list-style-type: none"> • The vendor name, the application must free the memory by calling the ldap_memfree function. • Some ldap extensions, the application must free the memory by calling the ldap_value_free function. <p>ldap_set_option data type: Not supported; ldap_get_option data type: LDAPAPIInfo *;</p>
LDAP_OPT_CLIENT_CONTROLS	0x0013	<p>Specifies a default list of client controls that affect the LDAP session.</p> <p>ldap_set_option data type, LDAPControl **; ldap_get_option data type: LDAPControl ***</p> <p>The application should free memory with ldap_controls_free.</p>
LDAP_OPT_CURRENT_NAME	0x7003	<p>Returns the client address associated with the supplied session handle argument.</p> <p>ldap_get_option data type : struct sockaddr_in *</p> <p>This is read only.</p>
LDAP_OPT_DEBUG_LEVEL	0x5001	<p>Contains the debug level. Uses the following values:</p> <p>0x0001 LDAP_DEBUG_TRACE 0x0002 LDAP_DEBUG_PACKETS 0x0004 LDAP_DEBUG_ARGS 0x0008 LDAP_DEBUG_CONNS 0x0010 LDAP_DEBUG_BER 0x0020 LDAP_DEBUG_FILTER 0x0040 LDAP_DEBUG_CONFIG 0x0080 LDAP_DEBUG_ACL 0x0100 LDAP_DEBUG_STATS 0x0200 LDAP_DEBUG_STATS2 0x0400 LDAP_DEBUG_SHELL 0x0800 LDAP_DEBUG_PARSE 0x8000 LDAP_DEBUG_NONE -1 LDAP_DEBUG_ANY</p> <p>ldap_set_option and ldap_get_option data type: int*</p>

Option	Value	Description
LDAP_OPT_DEREF	0x0002	<p>Determines how aliases are handled during a search. Supports the following values:</p> <p>LDAP_DEREF_NEVER (0x00) LDAP_DEREF_SEARCHING (0x01) LDAP_DEREF_FINDING (0x02) LDAP_DEREF_ALWAYS (0x03)</p> <p>The LDAP_DEREF_SEARCHING flag indicates that aliases are dereferenced during the search but not when locating the base object of the search.</p> <p>The LDAP_DEREF_FINDING flag indicates that aliases are dereferenced when locating the base object but not during the search.</p> <p>The LDAP_DEREF_ALWAYS flag indicates that aliases are dereferenced when locating the base object and when finding entries.</p> <p>The LDAP_DEREF_NEVER flag indicates that aliases are not dereferenced.</p> <p>The default is LDAP_DEREF_NEVER.</p> <p>ldap_get_option and ldap_set_option data type: int *</p>
LDAP_OPT_ERROR_STRING	0x0032	<p>Contains the message that returned with the most recent LDAP error that occurred on this session.</p> <p>ldap_set_option data type: char *; ldap_get_option data type: char **</p> <p>The application should free memory with ldap_memfree.</p>
LDAP_OPT_HOST_NAME	0x0030	<p>Specifies the host name or a list of hosts for the primary LDAP server.</p> <p>ldap_set_option data type: char *; ldap_get_option data type: char **</p> <p>The application should free memory with ldap_memfree.</p>
LDAP_OPT_MATCHED_DN	0x0033	<p>Contains the matched DN value returned with the most recent LDAP error that occurred on this session.</p> <p>ldap_set_option data type: char *; ldap_get_option data type: char **</p> <p>The application should free memory with ldap_memfree.</p>

Option	Value	Description
LDAP_OPT_NETWORK_TIMEOUT	0x5005	<p>Enables a connection timeout to be set. This is the timeout of the initial connection to a server, which usually occurs when the bind command is executed, or, if no bind command is given, on the first LDAP operation. Initial connections may also occur during a referral or rebind operation.</p> <p>If no timeout is set, timeout depends upon the underlying socket timeout setting of the operating system.</p> <p>ldap_set_option data type: struct timeval *; ldap_get_option data type: struct timeval **</p>
LDAP_OPT_PEER_NAME	0x7002	<p>Returns the peer address associated with the supplied session handle argument.</p> <p>ldap_get_option data type : struct sockaddr_in *</p> <p>This is read only.</p>
LDAP_OPT_PROTOCOL_VERSION	0x0011	<p>Specifies the version of the LDAP protocol used when communication with the LDAP server. It can be set to one of the following values:</p> <p>LDAP_VERSION2 (2) LDAP_VERSION3 (3)</p> <p>If no version is set, the default is LDAP_VERSION2.</p> <p>ldap_get_option and ldap_set_option data type: int *</p>
LDAP_OPT_REFERRAL_LIST	0x5007	<p>If the server returns referrals and the client library is set to return them to the application (LDAP_OPT_REFERRALS=0), this option can be used to obtain the list of referrals after an error 10 (LDAP_REFERRAL). It returns a NULL-terminated list of string pointers containing the referrals.</p> <p>ldap_set_option data type: char**; ldap_get_option data type char***</p> <p>The memory returned should be freed by the application with ldap_value_free().</p>
LDAP_OPT_REFERRALS	0x0008	<p>Determines whether the LDAP libraries automatically follow referrals. It can be set to one of the following values:</p> <p>LDAP_OPT_ON (void*) 1 LDAP_OPT_OFF (void*) 0</p> <p>The default is ON.</p> <p>ldap_set_option data type: void*; ldap_get_option data type: int*</p>

Option	Value	Description
LDAP_OPT_RESULT_CODE	0x0031	Specifies the code of the most recently returned LDAP error that occurred on this session. ldap_get_option and ldap_set_option data type: int *
LDAP_OPT_RESTART	0x0009	Determines whether LDAP I/O operations automatically restart if they abort prematurely. It can be set to one of the following values: LDAP_OPT_ON (void*) 1 LDAP_OPT_OFF (void*) 0 The default is OFF ldap_set_option data type: void*; ldap_get_option data type: int*
LDAP_OPT_SERVER_CONTROLS	0x0012	Specifies a default list of LDAP server controls that are sent with each request. ldap_set_option data type, LDAPControl **; ldap_get_option data type: LDAPControl *** The application should free memory with ldap_controls_free.
LDAP_OPT_SESSION_REFCNT	0x8001	Returns the reference count associated with the supplied session handle argument. This is read only.
LDAP_OPT_SIZELIMIT	0x0003	LDAP server sizelimit, determines how many entries are returned from a search. A value of LDAP_NO_LIMIT (0) means no limit. This is a server limit used in all search operations except when overridden by a client timeout in the search_ext functions. The default is LDAP_NO_LIMIT. ldap_get_option and ldap_set_option data type: int *
LDAP_OPT_TIMELIMIT	0x0004	LDAP Server timelimit, determines the number of seconds an LDAP server will spend on a search. A value of LDAP_NO_LIMIT (0) means no limit. This value is passed to the LDAP server in the search request. This is a server limit used in all search operations except when overridden by a client timeout in the search_ext functions. The default is LDAP_NO_LIMIT. ldap_get_option and ldap_set_option data type: int *

Option	Value	Description
LDAP_OPT_TLS_CIPHER_LIMIT	0x9001	<p>Contains the cipher level and its values:</p> <ul style="list-style-type: none"> • LDAP_TLS_CIPHER_LOW: The key strength is 56 and algorithm is single DES. • LDAP_TLS_CIPHER_MEDIUM: The key strength is 128 and algorithm is single RSA. • LDAP_TLS_CIPHER_HIGH: The key strength is 168 and algorithm is triple DES. • LDAP_TLS_CIPHER_EXPORT: The key strength is 56 and algorithm is SHA. <p>The default is LDAP_TLS_CIPHER_HIGH.</p> <p>ldap_get_option and ldap_set_option data type:int</p>

6.11 Schema Element Types

This chapter contains values used with the ldap_schema functions. The following list contains the types of schema elements that can be used:

- “LDAP_SCHEMA_ATTRIBUTE_TYPE” on page 418
- “LDAP_SCHEMA_OBJECT_CLASS” on page 420
- “LDAP_SCHEMA_MATCHING_RULE” on page 421
- “LDAP_SCHEMA_MATCHING_RULE_USE” on page 422
- “LDAP_SCHEMA_NAME_FORM” on page 422
- “LDAP_SCHEMA_SYNTAX” on page 423
- “LDAP_SCHEMA_DIT_CONTENT_RULE” on page 423
- “LDAP_SCHEMA_DIT_STRUCTURE_RULE” on page 423

Each section contains a table listing the field names valid in a specific type of a schema element. Addition fields to those defined in these sections may be used.

6.11.1 LDAP_SCHEMA_ATTRIBUTE_TYPE

Table 6-10 Details of the LDAP_SCHEMA_ATTRIBUTE_TYPE Schema Elements

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.

Flag Name	C Value	Description
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_SUPERIOR	SUP	Defines the name of the attribute from which this attribute is derived.
LDAP_SCHEMA_EQUALITY	EQUALITY	Defines the Object identifier of the Matching rule used for an equality comparison of this attribute.
LDAP_SCHEMA_SUPERIOR	SUP	Defines the name of the attribute from which this attribute is derived.
LDAP_SCHEMA_ORDERING	ORDERING	Defines the Object identifier of the Matching rule used for an ordering-collating comparison of this attribute.
LDAP_SCHEMA_SUBSTRING	SUBSTR	Defines the Object identifier of the Matching rule used for a substring comparison of this attribute.
LDAP_SCHEMA_SYNTAX_OID	SYNTAX	Defines the Object identifier of the syntax that will be used for this attribute.
LDAP_SCHEMA_SINGLE_VALUE D	SINGLE-VALUE	Defines whether or not this attribute is multi-valued or not. This field has no value. If the field name is present the attribute is single valued, otherwise it is multi-valued.
LDAP_SCHEMA_COLLECTIVE	COLLECTIVE	Defines whether or not this attribute is collective, meaning all instances of an object with this attribute will have the same value for this attribute. This field has no value. If the field name is present the attribute is collective, otherwise it is not.
LDAP_SCHEMA_NO_USER_MOD D	NO-USER-MODIFICATION	Defines whether or not a user can modify this attribute. This field has no value. If the field name is present the attribute is not modifiable, otherwise it is modifiable.

Flag Name	C Value	Description
LDAP_SCHEMA_USAGE	USAGE	<p>Defines whether this attribute is used by a user application, a directory operation, a distributed operation or a per-DSA (Directory Service Agent) operation. The following define strings for the value of this field:</p> <p>LDAP_SCHEMA_USER_APP userApplications If the LDAP_SCHEMA_USAGE field name has this value then the attribute is used by an application independent of the directory server.</p> <p>LDAP_SCHEMA_DIRECTORY_OP directoryOperation If the LDAP_SCHEMA_USAGE field name has this value then the directory uses the defined attribute.</p> <p>LDAP_SCHEMA_DISTRIBUTED_OP distributedOperation If the LDAP_SCHEMA_USAGE field name has this value then the attribute is share between DSAs, Directory Server Agents.</p>
LDAP_SCHEMA_DSA_OP	dSAOperation	If the LDAP_SCHEMA_USAGE field name has this value then the attribute can be unique for each DSA, Directory Server Agent.

6.11.2 LDAP_SCHEMA_OBJECT_CLASS

Table 6-11 Details of the LDAP_SCHEMA_OBJECT_CLASS Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_SUPERIOR	SUP	Defines all attributes that must be defined in an instance of this object.
LDAP_SCHEMA_MUST_ATTRIBUTES	MUST	Defines the Object identifier of the Matching rule used for an equality comparison of this attribute.
LDAP_SCHEMA_MAY_ATTRIBUTES	MAY	Defines all attributes that may be defined in an instance of this object.

Flag Name	C Value	Description
LDAP_SCHEMA_TYPE_ABSTRACT	ABSTRACT	Defines that this object is abstract. An abstract object can be derived from but not instantiated. This field name does not have a value. This field name cannot be present if LDAP_SCHEMA_TYPE_STRUCTURAL or LDAP_SCHEMA_TYPE_AUXILIARY is present.
LDAP_SCHEMA_TYPE_STRUCTURAL	STRUCTURAL	Defines that this object is structural. A structural object can be derived from and instantiated. This field name does not have a value. This field name cannot be present if LDAP_SCHEMA_TYPE_ABSTRACT or LDAP_SCHEMA_TYPE_AUXILIARY is present.
LDAP_SCHEMA_TYPE_AUXILIARY	AUXILIARY	Defines that this object is auxiliary. An auxiliary object can be associated with any instantiated object. This field name does not have a value. This field name cannot be present if LDAP_SCHEMA_TYPE_ABSTRACT or LDAP_SCHEMA_TYPE_STRUCTURAL is present.

6.11.3 LDAP_SCHEMA_MATCHING_RULE

Table 6-12 Details of the LDAP_SCHEMA_MATCHING_RULE Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_SYNTAX_OID	SYNTAX	Defines the syntax of the Matching Rule. Only one value can exist for this field name.

6.11.4 LDAP_SCHEMA_MATCHING_RULE_USE

Table 6-13 Details of the LDAP_SCHEMA_MATCHING_RULE Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_APPLIES	APPLIES	Defines the attributes that the Matching Rule applies to. This field is required for Matching Rule Use definitions.

6.11.5 LDAP_SCHEMA_NAME_FORM

Table 6-14 Details of the LDAP_SCHEMA_NAME_FORM Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_NAME_FORM_OBJECTS	OC	Defines the Object Classes to which this Name Form applies. This field is required for name forms..
LDAP_SCHEMA_MUST_ATTRIBUTES	MUST	Defines the mandatory attributes to which this name form applies. This field is required for name forms.
LDAP_SCHEMA_MAY_ATTRIBUTES	MAY	Defines the optional attributes to which this name form applies.

6.11.6 LDAP_SCHEMA_SYNTAX

Table 6-15 Details of the LDAP_SCHEMA_SYNTAX Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.

6.11.7 LDAP_SCHEMA_DIT_CONTENT_RULE

Table 6-16 Details of the LDAP_SCHEMA_DIT_CONTENT_RULE Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_OID	OID	Object identifier of the schema element. This field has only one value.
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_AUX_CLASSES	AUX	Defines the auxiliary classes that can be applied to a structural object Class.
LDAP_SCHEMA_MUST_ATTRIBUTES	MUST	Defines the mandatory attributes to which this name form applies. This field is required for name forms.
LDAP_SCHEMA_MAY_ATTRIBUTES	MAY	Defines the optional attributes to which this name form applies.
LDAP_SCHEMA_NOT_ATTRIBUTES	NOT	Defines the attributes that a structural object class cannot obtain from an auxiliary class.

6.11.8 LDAP_SCHEMA_DIT_STRUCTURE_RULE

Table 6-17 Details of the LDAP_SCHEMA_DIT_STRUCTURE_RULE Schema Element

Flag Name	C Value	Description
LDAP_SCHEMA_RULE_ID	RULEID	Defines the integer identifier for this rule.

Flag Name	C Value	Description
LDAP_SCHEMA_DESCRIPTION	DESC	This field is a string definition of the schema element. This field has only one value.
LDAP_SCHEMA_NAMES	NAME	Defines all names used to identify the schema element.
LDAP_SCHEMA_OBSOLETE	OBSOLETE	Defines whether this schema definition is still in use. This field has no value. If the field name is present, the definition is obsolete; otherwise the definition is still valid.
LDAP_SCHEMA_NAME_FORM_OID	FORM	Defines the Name Form that applies to this structure rule.
LDAP_SCHEMA_SUPERIOR	SUP	Defines all structure rules that this rule derives from.

6.12 SSL Certificate Status Codes

These status codes are used by interactive ssl and the [ldapssl_get_cert_attribute \(page 303\)](#) function.

Table 6-18 *SSL Certificate Status Codes*

Status
2 ERR_UNABLE_TO_GET_ISSUER_CERT Unable to get issuer certificate
6 ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY Unable to decode issuer public key
7 ERR_CERT_SIGNATURE_FAILURE Certificate signature failure
9 ERR_CERT_NOT_YET_VALID Certificate is not yet valid
10 ERR_CERT_HAS_EXPIRED CRL is not yet valid
13 ERROR_IN_CERT_NOT_BEFORE_FIELD Format error in certificate's notBefore field
14 ERROR_IN_CERT_NOT_AFTER_FIELD Format error in certificate's notAfter field
18 DEPTH_ZERO_SELF_SIGNED_CERT Self-signed certificate

Status	
19	SELF_SIGNED_CERT_IN_CHAIN Self-signed certificate in certificate chain
20	UNABLE_TO_GET_ISSUER_CERT_LOCALLY Unable to get local issuer certificate
21	UNABLE_TO_VERIFY_LEAF_SIGNATURE Unable to verify the first certificate
24	INVALID_CA Invalid CA certificate
25	PATH_LENGTH_EXCEEDED Path length constraint exceeded
26	INVALID_PURPOSE Unsupported certificate purpose
27	CERT_UNTRUSTED Certificate not trusted
28	CERT_REJECTED Certificate rejected

Structures

7

This chapter describes the structures used by the LDAP functions.

BerElement

Contains an opaque data structure for data encoded with BER (Basic Encoding Rules).

Remarks

The LDAP libraries provide functions for creating and manipulating the data within the BerElement structure, but clients do not need access to the fields in the structure.

For example, the `ldap_first_attribute` function creates a BerElement structure that tracks the current position in the entry.

The `ber_init` function converts a BerElement structure to a berval structure, and the `ber_flatten` function converts a berval structure to a BerElement structure.

berval

Contains binary data that is encoded with simplified BER (Basic Encoding Rules).

Structure

```
typedef struct berval {  
    unsigned long    bv_len;  
    char             *bv_val;  
};
```

Fields

bv_len

Specifies the length of the data.

bv_val

Points to the encoded data.

Remarks

The `ber_init` function converts a `BerElement` structure to a `berval` structure, and the `ber_flatten` function converts a `berval` structure to a `BerElement` structure.

DB_binary

Contains a binary debug event parameter value.

Structure

```
typedef struct DB_binary {  
    unsigned int  size;  
    void*         data;  
};
```

Fields

size

The number of bytes in binary value.

data

An array of size number of bytes containing the binary data.

DB_netAddress

Contains a net address debug event parameter value.

Structure

```
typedef struct DB_netAddress {  
    unsigned int  type;  
    unsigned int  length;  
    char*         data;  
};
```

Fields

type

An integer value indicating the address type.

length

The length, in bytes of the address value.

data

The actual address value.

DB_Parameter

Contains debug parameters associated with debug events.

Structure

```
typedef struct DB_Parameter {  
    int      type;  
    DB_value value;  
};
```

Fields

type

An integer that indicates the type of the parameter. It will be one of the following values:

Value	Type
1	DB_PARAM_TYPE_ENTRYID
2	DB_PARAM_TYPE_STRING
3	DB_PARAM_TYPE_BINARY
4	DB_PARAM_TYPE_INTEGER
5	DB_PARAM_TYPE_ADDRESS
6	DB_PARAM_TYPE_TIMESTAMP
7	DB_PARAM_TYPE_TIMEVECTOR

value

The DB_Value structure containing the actual parameter value.

DB_timeStampVector

Contains a time stamp vector debug event parameter value.

Structure

```
typedef struct DB_timeStampVector {  
    unsigned int    count;  
    EVT_TimeStamp  *timeStamps;  
};
```

Fields

count

The number of time stamps contained in the vector.

timeStamps

A pointer to an array containing count EVT_TimeStamp structures.

DB_value

Contains a value associated with debug events.

Structure

```
typedef union DB_value {  
    int                integer;  
    char               *utf8Str;  
    EVT_TimeStamp     timeStamp;  
    DB_netAddress      netAddress;  
    DB_binary          binary;  
    DB_timeStampVector timeStampVector;  
};
```

Fields

integer

Contains the integer value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_INTEGER.

utf8Str

Contains a pointer to the UTF-8 encoded string value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_STRING.

timeStamp

Contains the EVT_TimeStamp value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_TIMESTAMP.

netAddress

Contains the DB_netAddress value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_ADDRESS.

binary

Contains the DB_binary value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_BINARY.

timeStampVector

Contains the DB_timeStampVector value of the parameter if the type field of the DB_Parameter structure is DB_PARAM_TYPE_TIMEVECTOR.

EVT_BinderyObjectInfo

Contains information about a bindery object associated with an event.

Structure

```
typedef struct EVT_BinderyObjectInfo {  
    char          *entryDN;  
    unsigned int   type;  
    unsigned int   emuObjFlags;  
    unsigned int   security;  
    char          *name;  
};
```

Fields

entryDN

Specifies the DN of the Directory entry that is being created to represent the bindery object.

type

Specifies the bindery object type.

emuObjFlags

Specifies the bindery object flags.

security

Specifies the bindery object security.

name

Specifies the name of the bindery object.

EVT_ChangeConfigParm

Structure

```
typedef struct EVT_ChangeConfigParm {
    int     type;
    char    *name;
    union {
        int     integer;
        int     boolean;
        char    *utf8Str;
        struct {
            int     size;
            unsigned char* data;
        } binary;
    } value;
} EVT_ChangeConfigParm;
```

Fields

type

indicates the type of the configuration parameters data.

Type	Value
EVT_CFG_TYPE_NULL	0
EVT_CFG_TYPE_BINARY	1
EVT_CFG_TYPE_INT	2
EVT_CFG_TYPE_STRING	3
EVT_CFG_TYPE_BOOLEAN	4

name

name of the configuration parameter.

integer

If the value of type is EVT_CFG_TYPE_INT, this contains the integer value of the configuration parameter. This value is accessed using a pointer to the integer, such as data -> value.integer.

boolean

If the value of type is EVT_CFG_TYPE_BOOLEAN, this contains the boolean value of the configuration parameter (0 = false, 1 = true). This value is accessed using a pointer to the boolean, such as data -> value.boolean.

utf8str

If the value of type is `EVT_CFG_TYPE_STRING`, this contains a pointer to the utf-8 string value of the configuration parameter. This value is accessed using a pointer to the string, such as `data -> value.utf8str`.

size

If the value of type is `EVT_CFG_TYPE_BINARY`, this contains the number of bytes in the value of the configuration parameter. This value is accessed using a pointer to the size, such as `data -> value.binary.size`.

data

If the value of type is `EVT_CFG_TYPE_BINARY`, this contains a pointer to an array of the bytes in the value of the configuration parameter. This value is accessed using a pointer to the array, such as `data -> value.binary.data`.

EVT_ChangeConnState

Contains information about a connection whose state is being changed.

Structure

```
typedef struct EVT_ChangeConnState {  
    char            *connectionDN;  
    unsigned int    oldFlags;  
    unsigned int    newFlags;  
    char*           sourceModule;  
};
```

Fields

connectionDN

Specifies the DN of the entry associated with the connection.

oldFlags

Specifies the flag associated with the previous connection state, and is one of the following values:

C Value	Value Name
0x00000001	DSE_CONN_VALID
0x00000002	DSE_CONN_AUTHENTIC
0x00000004	DSE_CONN_SUPERVISOR
0x00000008	DSE_CONN_OPERATOR
0x00000010	DSE_CONN_LICENSED
0x00000020	DSE_CONN_SEV_IS_STALE
0x000000FF	DSE_CONN_OPERATIONAL_FLAGS
0x00010000	DSE_CONN_CLEAR_ON_UNLOCK
0x00020000	DSE_CONN_LOCKED
0x00040000	DSE_CONN_CLEAR_ON_EVENT
0x000F0000	DSE_CONN_SECURITY_FLAGS

newFlags

Specifies the flag that indicates the new connection state. Uses the same flags as oldFlags.

sourceModule

Specifies the module that caused the connection state to change.

EVT_ChangeServerAddr

Structure

```
typedef struct EVT_ChangeServerAddr {  
    unsigned    flags;  
    int         proto;  
    int         addrFamily;  
    int         addrSize;  
    unsigned char *addr;  
    char        *pstkname;  
    char        *sourceModule;  
};
```

Fields

flags

proto

addrFamily

addrSize

addr

pstkname

sourceModule

Remarks

EVT_DebugInfo

Contains data associated with debug events.

Structure

```
typedef struct EVT_DebugInfo {
    unsigned int    dsTime;
    unsigned int    milliseconds;
    char            *perpetratorDN;
    char            *formatString;
    int             verb;
    int             paramCount;
    DB_Parameter    *parameters;
};
```

Fields

dsTime

Specifies the time the event occurred as the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time, according to the system clock.

milliseconds

The millisecond portion of the time the event occurred.

perpetratorDN

The DN of the object that caused this event.

formatString

The format string used to create the string printed in the DS Trace utility. The format string describes the string that is displayed by the DS Trace utility. It contains literal characters as well as format characters that serve as place holder for parameter values. See the remarks for a list of valid format characters.

verb

The ID of the ds verb that was executing when the event occurred.

paramCount

The number of parameters specified in the format string.

parameters

A pointer to an array containing paramCount DB_Parameter structures. The parameters are in the same order as the parameter characters in the format string.

Remarks

The formatString parameter is formatted according to the following:

```
%[flags][width][.precision][L,l,h,! ]type
```

Element	Description
flags	-, +, #, 0
width	An optional integer indicating the width of the displayed value
precision	An optional integer indicating the precision of the displayed value
L, l, h, !	a character indication the size of the parameter, one of the following values: <ul style="list-style-type: none"> • L: DOUBLE_FLAG • l: LONG_FLAG • h: SHORT_FLAG • !: l64_FLAG
type	A character indicating the data type of the parameter, one of the following values: <p>C: color (no associated parameter)</p> <p>t: current time (no associated parameter)</p> <p>s: string, EVT_TAG_DB_STRING</p> <p>a: network address</p> <p>U: string, EVT_TAG_DB_STRING</p> <p>T: time stamp</p> <p>V: time stamp vector</p> <p>S: string, EVT_TAG_DB_STRING</p> <p>D: binary data</p> <p>x: hex integer, EVT_TAG_DB_INTEGER</p> <p>v: verb number, EVT_TAG_DB_INTEGER</p> <p>u: unsigned decimal integer, EVT_TAG_DB_INTEGER</p> <p>o: octal integer, EVT_TAG_DB_INTEGER</p> <p>e: error code value, EVT_TAG_DB_INTEGER</p> <p>d: normal decimal integer, EVT_TAG_DB_INTEGER</p> <p>c: single character, EVT_TAG_DB_INTEGER</p> <p>p: raw memory pointer, EVT_TAG_DB_INTEGER</p> <p>X: HEX integer, EVT_TAG_DB_INTEGER</p> <p>E: error code value, EVT_TAG_DB_INTEGER</p>

EVT_EntryInfo

Contains data associated with state changes on individual entries in the directory.

Structure

```
typedef struct EVT_EntryInfo {  
    char            *perpetratorDN;  
    char            *entryDN;  
    char            *className;  
    unsigned int     verb;  
    unsigned int     flags;  
    EVT_TimeStamp    creationTime;  
    char            *newDN;  
};
```

Fields

perpetratorDN

Specifies the DN of the entry that caused the event.

entryDN

Specifies the DN of the entry that was acted upon.

parentDN

Specifies the parent DN of the acted upon entry.

className

Specifies the DN of the object that was acted upon.

verb

Specifies the action that caused the event to occur.

flags

creationTime

newDN

Specifies the new DN of the entry that was acted upon.

EVT_EventData

Contains data associated with general DS events. The meaning of this structure's content is dependent on the type of event.

Structure

```
typedef struct EVT_EventData {  
    unsigned int    dstime;  
    unsigned int    milliseconds;  
    unsigned int    curProcess;  
    unsigned int    verb;  
    char            *perpetratorDN;  
    unsigned int    intValues[4];  
    char            strValues[4];  
};
```

Fields

dstime

Specifies the time in milliseconds when the event occurred.

milliseconds

curProcess

Specifies the process that was running when the event occurred.

verb

Specifies the action that caused the event to occur.

perpetratorDN

Specifies the DN of the entry that caused the event.

intValues

Contains event data determined by the event type

strValues

Contains event data determined by the event type.

EVT_EventSpecifier

Contains information about a single event to monitor.

Structure

```
typedef struct EVT_EventSpecifier {  
    int     eventType;  
    int     eventStatus;  
};
```

Fields

eventType

Specifies an event type to monitor. For a complete listing of events, see “[LDAP Event Services](#)” in the LDAP and eDirectory Integration Guide.

eventStatus

Specifies the event status for which you would like to be notified. This can be one of the following values:

Status	Value
EVT_STATUS_ALL	0
EVT_STATUS_SUCCESS	1
EVT_STATUS_FAILURE	2

EVT_STATUS_ALL causes all events to be reported regardless of status.

EVT_STATUS_SUCCESS causes only events with a successful result to be reported.

EVT_STATUS_FAILURE causes only events with a failure result to be reported.

EVT_FilteredEventSpecifier

Contains information about a single event to monitor, including a filter used by the server to limit returned events.

Structure

```
typedef struct EVT_FilteredEventSpecifier {  
    int     eventType;  
    int     eventStatus;  
    char*   filter;  
};
```

Fields

eventType

Specifies an event type to monitor. For a complete listing of events, see “[LDAP Event Services](#)” in the LDAP and eDirectory Integration Guide.

eventStatus

Specifies the event status for which you would like to be notified. This can be one of the following values:

Status	Value
EVT_STATUS_ALL	0
EVT_STATUS_SUCCESS	1
EVT_STATUS_FAILURE	2

EVT_STATUS_ALL causes all events to be reported regardless of status.

EVT_STATUS_SUCCESS causes only events with a successful result to be reported.

EVT_STATUS_FAILURE causes only events with a failure result to be reported.

filter

Specifies a filter to limit events returned by the server. This event filter is patterned after the string representation of an LDAP search filter, and can filter based on any of the parameters returned in an event structure. See the remarks for additional information.

Remarks

An event filter is patterned after the string representation of an LDAP search filter. An event filter is contained in parenthesis "()", and can filter events based on one or more values returned by an event.

For example, a value event (a change to an attribute value) returns the following nine parameters in an EVT_ValueInfo structure:

```
verb  
perpetratorDN  
entryDN
```

```
attributeName
syntaxOID
className
timeStamp
size
value
```

When monitoring a value event, you can specify a filter based on one or more of the the nine values returned by this event.

For example, the following event filter causes the server to return only value events where the acted upon attribute is a title:

```
(attributeName=title)
```

More complex event filters can be created using the same syntax as LDAP search filters.

```
(&(entryDN=cn=user1,o=system)(perpetratorDN=cn=admin,o=system)(attributeName=fullName))
```

```
(|(attributeName=modifiersName)(&(entryDN=cn=user1,o=system)(perpetratorDN=cn=admin,o=system)(attributeName=fullName)))
```

For additional information on LDAP search filters see [“Using Search Filters” on page 37](#).

EVT_ModuleState

Contains information about an eDirectory module state that is being changed.

Structure

```
typedef struct EVT_ModuleState {
    char        *connectionDN;
    unsigned    flags;
    int         module;
    int         source;
    char        name[EVT_MAX_MODULE_NAME];
    char        description[EVT_MAX_MODULE_DESCR];
};
```

Fields

connectionDN

Specifies the DN of the entry associated with the connection.

flags

The least significant byte of the flags field contains module attribute flags. The next byte contains event subtype flags. They indicate the type of module event in progress. The values for flags field are contained in the following table:

0x0001 DSE_MOD_HIDDEN
0x0002 DSE_MOD_SYSTEM
0x0004 DSE_MOD_ENGINE
0x0008 DSE_MOD_AUTOMATIC
0x00FF DSE_MOD_FILE_MASK
0x0100 DSE_MOD_POSTEVENT
0x0200 DSE_MOD_AVAILABLE
0x0400 DSE_MOD_LOADING
0x0800 DSE_MOD_MODIFY
0x8000 DSE_MOD_NEGATE_BIT
0xFF00 DSE_MOD_EVENT_MASK

The NEGATE_BIT negates the meaning of the other event type flags. For example, the DSE_MOD_LOADING flag is set along with the DSE_MOD_NEGATE_BIT to indicate the module is unloading.

module

Target module for this event.

source

Specifies the affecting module

name

Module name

description

Specifies the name and description of the target module.

EVT_NetAddress

Contains a network address associated with a DSEvent.

Structure

```
typedef struct EVT_NetAddress {  
    unsigned int    type;  
    unsigned int    length;  
    char            data[1];  
  
};
```

Fields

type

Specifies the type of the address. Can be one of the following values:

- NT_IPX
- NT_IP
- NT_SDLC
- NT_TOKENRING_ETHERNET
- NT_OSI
- NT_APPLETALK
- NT_COUNT

length

Specifies the number of bytes in which the address is stored.

data

A char array of bytes [length] long, containing the network address.

Remarks

The address is stored as a binary string. This string is the literal value of the address. To display it as a hexadecimal value, you must convert each 4-bit nibble to the correct character (0,1,2,3,...F).

For two net addresses to match, the type, length, and value of the addresses must match.

EVT_ReferralAddress

Structure

```
typedef struct EVT_ReferralAddress {  
    int      type;  
    int      length;  
    char     *address;  
};
```

Fields

type

indicates the address type.

length

length of the referral address.

address

Pointer to the address.

EVT_SEVInfo

Contains a Security Equivalence Vector associated with a DSEvent.

Structure

```
typedef struct EVT_SEVInfo {  
    char          *entryDN;  
    unsigned int   retryCount;  
    char          *valueDN;  
    int            referralCount;  
    EVT_ReferralAddress *referrals;  
};
```

Fields

entryDN

Specifies the DN of the Directory object whose Security Equivalence Vector (SEV) is being checked.

retryCount

Reserved

valueDN

Specifies the DN of an object or group being checked.

referralCount

Specifies the number of referrals in the referrals parameter.

referrals

Pointer to an array of [EVT_ReferralAddress \(page 450\)](#) structures.

EVT_TimeStamp

Contains a time stamp associated with an event.

Structure

```
typedef struct EVT_TimeStamp {  
    unsigned int    seconds;  
    unsigned short  replicaNumber;  
    unsigned short  event;  
  
};
```

Fields

seconds

Specifies in seconds when the event occurred. Zero equals 12:00 midnight, January 1, 1970, UTC.

replicaNumber

Specifies the number of the replica on which the change or event occurred.

event

Specifies an integer that further orders events occurring within the same whole-second interval.

Remarks

Two time stamp values are compared by comparing the seconds fields first and the event fields second. If the seconds fields are unequal, order is determined by the seconds field alone. If the seconds fields are equal, and the eventID fields are unequal, order is determined by the eventID fields. If the seconds and the event fields are equal, the time stamps are equal.

EVT_ValueInfo

Contains data associated with changes to individual attributes.

Structure

```
typedef struct EVT_ValueInfo {
    unsigned int    verb;
    char            *perpetratorDN;
    char            *entryDN;
    char            *attributeName;
    char            *syntaxOID;
    char            *className;
    EVT_TimeStamp   timeStamp;
    unsigned        size;
    char            *value;
};
```

Fields

verb

Specifies the action that caused the event to occur.

perpetratorDN

Specifies the DN of the entry that caused the event.

entryDN

Specifies the DN of the entry that was acted upon.

attributeName

Specifies the DN of the attribute that was acted upon.

syntaxOID

Specifies the Syntax OID of the entry that was acted upon.

className

Specifies the DN of the object that was acted upon.

timeStamp

size

Specifies the size (in bytes) of the information stored in the location identified by value.

value

Specifies the information that further identifies the changes that were made.

LBURPUpdateResult

Contains the result set of an LBURP operation.

Structure

```
typedef struct lburpupdateresult {    int sequenceNumber;    int  
resultCode;    char *errorMsg;} LBURPUpdateResult;
```

Fields

sequenceNumber

Points to the sequence number used to specify the ordering of the LBURP operation.

resultCode

Points to the response code from the server.

errorMessage

Points to the error message from the server, may be NULL if no error messages are requested.

LBURPUpdateOperationList

Contains the modifications to make to an entry.

Structure

```
typedef struct lburpoperationlist {    int operation;    char *dn;  
union {        LDAPMod **attrs;        char *newRDN;        int  
deleteOldRDN;        char *newSuperior;    }value;    LDAPControl  
**Servercontrols;    LDAPControl **Clientcontrols;}  
LBURPUpdateOperationList;
```

Fields

operation

Specifies the type of modification operation.

LDAP_REQ_ADD	Indicates an add operation.
LDAP_REQ_DELETE	Indicates a delete operation.
LDAP_REQ_MODIFY	Indicates a modify operation.
LDAP_REQ_MODRDN	Indicates a modify RDN operation.

dn

Points to the distinguished name of the entry.

attrs

Points to a NULL terminated array of LDAPMod structures that contain the attributes and value of the entry. All mandatory attributes must have values or the operation fails.

newRDN

Points to the new relative distinguished name for the entry. The entry's parent must remain the same. Applies to the MOD RDN operation only.

deleteOldRDN

Points to whether to delete the old RDN or not. Applicable to MOD RDN operation only

newSuperior

New superior DN.

Servercontrols

Points to an array of LDAPControl structures that list the server controls to use with the operation. Use NULL to specify no server controls.

Clientcontrols

Points to an array of LDAPControl structures that list the client controls to use with the operation. Use NULL to specify no client controls.

LDAP

Contains an opaque data structure for LDAP session handle information.

Remarks

All LDAP operation functions require the client to use an LDAP structure with the request. The LDAP structure contains session specific data about the connection to the LDAP server.

The LDAP library does not allow the client to directly manipulate the data in this session handle. Instead, it provides the following functions for various tasks.

Task	Function
Create	ldap_init or ldap_open
View settings	ldap_get_option
Modify settings	ldap_set_option
Delete	ldap_unbind, ldap_unbind_s, ldap_unbind_ext

For a list of the options that can be viewed or set, see [Section 6.10, “Session Preference Options,” on page 413](#).

LDAP_DIGEST_MD5_CONTEXT

Contains an opaque data structure for Digest-md5 data.

Remarks

This structure is used by [ldap_bind_digest_md5_start](#) (page 91), [ldapssl_install_routines](#) (page 161), and [ldap_bind_digest_md5_finish](#) (page 93) to contain Digest-MD5 data.

LDAPAPIFeatureInfo

Contains version information about the LDAP API extended features.

Structure

```
typedef struct ldap_apifeature_info {  
    int      ldapaif_info_version;  
    char     *ldapaif_name;  
    int      ldapaif_version;  
} LDAPAPIFeatureInfo;
```

Fields

ldapaif_info_version

Specifies the version of the LDAPAPIFeatureInfo structure.

ldapaif_name

Points to the name of the supported feature.

ldapaif_info_version

Specifies the revision of the supported feature.

Remarks

LDAPAPIInfo

Contains information about the vendor's implementation of the LDAP API.

Structure

```
typedef struct ldapapiinfo {  
    int      ldapai_info_version;  
    int      ldapai_api_version;  
    int      ldapai_protocol_version;  
    char     **ldapai_extensions;  
    char     *ldapai_vendor_name;  
    int      ldapai_vendor_version;  
} LDAPAPIInfo;
```

Fields

ldapai_info_version

Specifies the version of the LDAPAPIInfo structure.

ldapai_api_version

Specifies the revision of the API supported.

ldapai_protocol_version

Specifies the highest LDAP version supported by the LDAP library.

ldapai_extensions

Points to a NULL-terminated array of character strings that names the vendor's LDAP extensions. If no API extensions are supported, this field is set to NULL. The application is responsible for freeing this memory by calling the `ldap_value_free` function.

ldapai_vendor_name

Points to the vendor's name. The application is responsible for freeing this memory by calling the `ldap_memfree` function.

ldapai_vendor_version

Specifies the vendor's version of the LDAP libraries.

Remarks

To retrieve more information about an extension (the `ldapai_extensions` field), call the `ldap_get_option` function with the option parameter set to `LDAP_OPT_API_FEATURE_INFO`.

LDAPControl

Contains data about an LDAP control.

Structure

```
typedef struct ldapcontrol {
    char          *ldctl_oid;
    struct berval  ldctl_value;
    char          ldctl_iscritical;
} LDAPControl;
```

Fields

ldctl_oid

Points to the string object identifier (OID) assigned to the control.

ldctl_value

Specifies a [berval \(page 429\)](#) structure that contains the data, if any, associated with the control. The ldctl_value field can contain no data.

- To indicate a zero-length value, set ldctl_value.bv_len to zero and ldctl_value.bv_val to a zero-length string.
- To indicate that no data is associated with the control, set ldctl_value.bv_val to NULL.

ldctl_iscritical

Specifies whether the control is critical to the operation.

- If this field is non-zero, the operation fails if the LDAP server doesn't recognize the control.
- If this field is set to zero, the LDAP can continue the operation when it doesn't recognize the control.

Remarks

LDAPMessage

Contains an opaque data structure for the results of an asynchronous LDAP operation or a search operation.

Remarks

The following functions create either an LDAPMessage structure or an array of LDAPMessage structure.

- Search functions: `ldap_search_ext_s`, `ldap_search_ext`, `ldap_search`, `ldap_search_s`, and `ldap_search_st`
- Asynchronous operations that require `ldap_result` to read the results such as `ldap_add` and `ldap_add_ext`, `ldap_compare` and `ldap_compare_ext`, `ldap_delete` and `ldap_delete_ext`, `ldap_modify` and `ldap_modify_ext`

Use the `ldap_msgfree` function to free the LDAPMessage structure.

LDAPMod

Contains the modifications to make to one attribute of an entry.

Structure

```
typedef union mod_vals_u {
    char          **modv_strvals;
    struct berval  *mod_bvals;
} mod_vals_u_t;

typedef struct ldapmod {
    int            mod_op;
    char           *mod_type;
    mod_vals_u_t   mod_vals;
#define mod_values  mod_vals.modv_strvals
#define mod_bvalues mod_vals.modv_bvals
} LDAPMod;
```

Fields

mod_op

Specifies the type of modification operation.

- LDAP_MOD_ADD (0x0000)—adds the value, adding the attribute if no values currently exist.
- LDAP_MOD_DELETE (0x0001)—deletes the specified values, removing the attribute if no values remain.
- LDAP_MOD_REPLACE (0x0002)—replaces the current values with the specified values, adding the attribute if no values currently exist and removing the attribute if the specified value's field is NULL.
- LDAP_MOD_BVALUES (0x0080)—specifies binary values. If the mod_vals structure contains binary values, this flag should be ORed to one of the other flags to specify a binary modification. If this flag is not ORed, the default is to assume string value modifications.

mod_type

Points to the name of attribute to modify.

modv_strvals

Points to a NULL-terminated array of string values for the attribute. This field cannot contain values if the modv_bvals field contains values.

modv_bvals

Points to a NULL-terminated array of berval structures which are used to modify an attribute's binary values. This field cannot contain values if the modv_strvals field contains values.

Remarks

If `mod_op` is set to an operation flag with `LDAP_MOD_BVALUES` ORed to it, the `modv_strvals` should be empty. If `LDAP_MOD_BVALUES` is not ORed to the operation flag, the `modv_strvals` should contain the values for the modification operation.

Either the attribute contains string or binary values. Select the one that matches the attribute's syntax.

LDAPReplicaInfo

Contains information about a replica.

Service: LDAP

Defined In: ldapx.h

Structure

```
typedef struct ldapreplicainfo {
    unsigned long    rootID;
    unsigned long    state;
    unsigned long    modificationTime;
    unsigned long    purgeTime;
    unsigned long    localReplicaID;
    char             namingContextDN[2*256];
    LDAP_REPLICA_TYPE replicaType;
    unsigned long    flags;
} LDAPReplicaInfo;
```

Fields

rootID

Contains the entry ID of the naming context (replica) root object on the local server.

state

Contains the current state of the replica (see [Section 6.5, “Replica States,”](#) on page 409).

modificationTime

Contains the time for the most recent modification to the replica.

purgeTime

Contains the time at which all data has been synchronized. Data scheduled for deletion, that predates this time, can now be deleted.

localReplicaID

Contains the local server's identifier for the replica.

namingContextDN

Contains the distinguished name of the naming context (replica).

replicaType

Contains the replica's type (see [Section 6.7, “Replica Types,”](#) on page 411).

flags

Indicates whether the replica is busy performing an operation or not. Uses the following flags:

LDAP_DS_FLAG_BUSY (0x01)

1 indicates busy, 0 indicates not busy.

LDAPSchema

Contains an opaque data structure for schema information.

Remarks

LDAPSchema represents a local copy of an LDAP Directory schema. This structure is needed to locate, modify, and delete schema definitions.

LDAPSchemaElement

Contains an opaque data structure for a single schema definition.

Remarks

An LDAPSchemaElement represents one of eight possible schema definition types described in [Section 6.11, “Schema Element Types,” on page 418](#).

LDAPSchemaMod

Contains the definition of one field in a schema definition.

Structure

```
typedef struct ldap_schema_mod {  
    int      op;  
    char     *fieldName;  
    char     **values;  
};
```

Fields

op

Indicates whether the values are to add to, replace, or delete from the existing values of a field, and is one of the following values:

- LDAP_MOD_ADD
- LDAP_MOD_DELETE
- LDAP_MOD_REPLACE

fieldName

Identifies the name of the field. Macros for standard field names are defined in [Section 6.11, “Schema Element Types,”](#) on page 418.

values

A NULL-terminated array of strings, containing the values that correspond to the field name.

Remarks

A NULL-terminated array of LDAPSchemaMod structures represent all fields to be included in a new definition, or all fields to be modified in an existing definition. A field value can be added, replaced or deleted.

LDAPSortKey

Contains information about a sort key.

Structure

```
typedef struct ldapsortkey {  
    char    attributeType;  
    char    *orderingRule;  
    int     reverseOrder;  
} LDAPSortKey;
```

Fields

attributeType

Points to the name of the attribute to use for sorting.

orderingRule

Points to the OID of the ordering rule to use for the sorting. eDirectory does not support ordering rules.

reverseOrder

Specifies whether to sort results in reverse order:

- Non-zero indicates that the results are sorted in reverse order (large to small). eDirectory does not support this type of sort.
- Zero indicates that the results are sorted in forward order (small to large)

LDAPSSL_Cert

Contains SSL certificate information.

Structure

```
typedef struct_LDAPSSL_Cert {  
    unsigned long    length;  
    void             *data;  
}
```

Fields

length

The length of the memory pointed to by data.

data

Points to memory allocated by the application for the certificate information.

Remarks

The LDAPSSL_Cert structure is used by [ldapssl_get_cert \(page 301\)](#). After retrieving the certificate, the structure can be passed into [ldapssl_add_trusted_cert \(page 299\)](#) to add the certificate to the list of trusted certificates.

LDAPSSL_Cert_Validity_Period

Contains the earliest and latest times that a certificate is valid.

Structure

```
typedef struct LDAPSSL_Cert_Validity_Period {  
    char    notBeforeTime[40];  
    int     notBeforeType;  
    char    notAfterTime[40];  
    int     notAfterType;  
}
```

Fields

notBeforeTime

A string representation of the first time that the certificate should be considered valid.

notBeforeType

The type of the notBeforeTime parameter. The time can be represented as universal time string or a generalized time string. (LDAPSSL_CERT_UTC_TIME or LDAPSSL_CERT_GEN_TIME).

notAfterTime

A string representation of the expiration time of the certificate.

notAfterType

The type of the notAfterTime parameter. The time can be represented as universal time string or generalized time string. (LDAPSSL_CERT_UTC_TIME or LDAPSSL_CERT_GEN_TIME)

Remarks

Generalized Time Format. generalized time represents the values of year, month, day, hour, minutes, seconds and fractions of a second in any of three forms:

- Local time "YYYYMMDDHHMMSS.fff", where fff is optional and is fractions of a second
- Greenwich Mean Time (UTC) "YYYYMMDDHHMMSS.fffZ", Z indicates Greenwich Mean Time
- Difference between local and UTC time, "YYYYMMDDHHMMSS.fff+-HHMM", the +HHMM or -HHMM represents the time differential between the local and Greenwich Mean Times.

UTC Time Format. UTC format represents the values of year (2 digit), month, day, hour, minutes and optionally seconds.

- Local time "YYMMDDHHMMSS", where seconds (SS) is optional
- Greenwich Mean Time (UTC), "YYMMDDHHMMSSZ", seconds (SS) is optional and Z represents Greenwich Mean Time

- Difference between local and UTC time, "YYMMDDHHMMSS+-HHMM", seconds (SS) is optional and +HHMM or -HHMM represents the time differential between local and Greenwich Mean Times.

LDAPURLDesc

Contains URL information and the parameters for the search operation.

Structure

```
typedef struct ldap_url_desc {
    struct ldap_url_desc *lud_next;
    char *lud_scheme;
    char *lud_host;
    int lud_port;
    char *lud_dn;
    char **lud_attrs;
    int lud_scope;
    char *lud_filter;
    char **lud_exts;
    char lud_crit_exts;
} LDAPURLDesc;
```

Fields

lud_next

Points to the next URL.

lud_scheme

Specifies the URL scheme (either ldap or ldaps).

lud_host

Points to the name of the host as a dotted IP address or DNS format.

lud_port

Specifies the port from the URL.

lud_dn

Points to the distinguished name of the base entry from the URL.

lud_attrs

Points to a NULL-terminated list of attributes specified in the URL.

lud_scope

Specifies the scope in the URL and uses one of the following flags.

- LDAP_SCOPE_BASE (0x00)—searches the entry specified by the base parameter.
- LDAP_SCOPE_ONELEVEL (0x01)—searches the entry specified by the base parameter and one level beneath that entry.
- LDAP_SCOPE_SUBTREE (0x02)—searches the entire subtree starting with the entry specified by the base parameter.

lud_filter

Points to the search filter specified in the URL.

If NULL is passed, a default filter ("objectclass=*) is used.

lud_exts

Points to a NULL-terminated list of the extensions specified in the URL.

lud_crit_exts

Specifies whether or not any critical extensions are included.

LDAPVLVInfo

Contains state information associated with a series of virtual list view interactions between a client and an LDAP server.

Structure

```
typedef struct ldapvlvinfo {
    int                ldvlv_version;
    unsigned long      ldvlv_before_count;
    unsigned long      ldvlv_after_count;
    unsigned long      ldvlv_offset;
    unsigned long      ldvlv_count;
    struct berval       *ldvlv_attrvalue;
    struct berval       *ldvlv_context;
    void               *ldvlv_extradata;
} LDAPVLVInfo;
```

Fields

ldvlv_version

Specifies the version of this structure, which is currently 1.

ldvlv_before_count

Specifies the number of entries before the target entry that the client wants the server to return.

ldvlv_after_count

Specifies the number of entries after the target entry that the client wants the server to return.

ldvlv_offset

Specifies the target entry's position in the list. This parameter is used in connection with the ldvlv_count field, but is used only if the ldvlv_attrvalue field is NULL.

ldvlv_count

Specifies the total number of entries in the list. This parameter is used in connection with the ldvlv_offset field, but is used only if the ldvlv_attrvalue field is NULL. The following values have special consequences:

- If the ldvlv_count field is set to 0, the Novell LDAP server returns the entry specified by the ldvlv_offset parameter and sets this field to the number of entries currently in the list.
- If the value of the ldvlv_count field does not match the current number of entries in the list, the Novell LDAP server assumes that the ldvlv_offset parameter is relative to ldvlv_count. For example, if the list contains 10,000 entries and you specify the count as 500 and the offset as 250, the middle entry of the list is returned which, in this case, is entry number 5,000.

ldvlv_attrvalue

Points to the attribute value that the target entry's attribute is equal to or greater than. This can be used as a typedown value. For example, if the value specified is abc, the target entry will be the first entry in the list with abc, or if no abc entries exist, the first entry with abd. If this field is NULL, the ldvlv_offset and ldvlv_count fields are used to select the target entry.

ldvly_context

Points to server-specific data. On the first call, set this field to NULL. The server returns data that helps the server track who you are and where you are in the list. The context obtained from calling the `ldap_parse_vlv_control` function should be used as the context in the next `ldap_create_vlv_control` call.

ldvly_extradata

Reserved for application specific data. The virtual list view control does not use this field.

timeval

Contains timeout values for search requests.

Structure

```
typedef struct timeval {
    long    tv_sec;
    long    tv_usec;
};
```

Fields

tv_sec

Specifies the number of seconds for the time interval component.

tv_usec

Specifies the number of microseconds for the time interval component.

Remarks

These fields are used to determine the timeout value for both the LDAP server and the LDAP client libraries:

- If the server timeout expires before the server finishes the search operation, the server returns LDAP_TIMELIMIT_EXCEEDED to the application.
- If the client timeout expires before the server returns, the client returns LDAP_TIMEOUT to the application and sends an ldap_abandon to the server.

These fields have the following meanings for the timeout value for the LDAP server.

Field Values	Description
tv_sec=0; tv_usec>0	Sends a timeout value of one second to the server.
tv_sec>0; tv_usec>=0	Sends the tv_sec value to the server. The server ignores the tv_usec field.

The fields have the following meaning for a client timeout value.

Field Values	Description
tv_sec>=0; tv_usec>=0	Waits the time specified by the combination of the tv_sec and tv_usec fields.

The following table shows potential values for the fields and the timeout value that is computed for the server and the client.

Field Values	Server Timeout Value	Client Timeout Value
tv_sec=0; tv_usec=1	1 second	1 microsecond
tv_sec=1; tv_usec=500000	1 second	1.5 seconds
tv_sec=2; tv_usec=0	2 seconds	2 seconds

Only one of the fields can be set to zero. When both the tv_sec and tv_usec fields are set to zero, LDAP returns LDAP_PARAM_ERROR.

Source Code Contributors

A

Novell would like to acknowledge the following for contributing source code to the ldapsdk.* library:

- Copyright 1998, 1999 The OpenLDAP Foundation, Redwood City, CA All rights reserved.
- Copyright © 1990, 1995 Regents of the University of Michigan. All rights reserved.
- Copyright © 1987 Regents of the University of California
- Copyright © 1991 by the Massachusetts Institute of Technology
- Copyright © 1994 Enrique Silvestre Mora, Universitat Jaume I, Spain.
- Copyright © 1992, 1993, 1994 Henry Spencer.
- Copyright © 1992, 1993, 1994 The Regents of the University of California.
- Copyright © 1997, 1998, 1999 Computing Research Labs, New Mexico State University
- Copyright © 1999 PADL Software Pty Ltd.

For the complete text of these copyright notices, view the source code from [OpenLDAP \(http://www.openldap.org\)](http://www.openldap.org).

Revision History

B

The following table outlines all changes made to the LDAP Libraries for C documentation (in reverse chronological order):

June 2006	Added information to the subsection “GSSAPI” on page 31, in the Section 1.3.2, “Authentication,” on page 27
March 2006	Fixed formatting issues.
October 2005	<p>Added the following:</p> <ul style="list-style-type: none">• Information on Section 1.2.6, “Setting and Getting the Cipher Level,” on page 23.• 3 new session preference options, “LDAP_OPT_CURRENT_NAME” on page 414, “LDAP_OPT_PEER_NAME” on page 416, and “LDAP_OPT_TLS_CIPHER_LIMIT” on page 418.• Maximizing the security over the LDAP servers Section 1.3.5, “Recommendations,” on page 33. <p>Removed all instances of LDAPSSL_VERIFY_NONE option.</p>
June 2005	<p>Added the following:</p> <ul style="list-style-type: none">• A new search scope, LDAP_SCOPE_SUBORDINATESUBTREE (page 26).• ldap_create_geteffective_control (page 121).• ldap_create_reference_control (page 125).• ldap_create_sstatus_control (page 130).• ldap_parse_reference_control (page 214).• ldap_parse_sstatus_control (page 223).
March 2005	<ul style="list-style-type: none">• Added 2 new standard LDAP functions, ldap_cancel_ext (page 99) and ldap_cancel_ext_s (page 101).• Changed the syntax of two LDAP Extension Function, ldap_backup_object (page 320) and ldap_restore_object (page 374).
October 2004	<p>Added the following:</p> <ul style="list-style-type: none">• Information on Section 1.5, “LDAP Based Backup,” on page 39 and “GSSAPI” on page 31.• 2 new standard LDAP functions, ldap_gssbind (page 171) and ldap_gss_error (page 173).• 2 new LDAP extension functions, ldap_backup_object (page 320) and ldap_restore_object (page 374).
June 2004	<p>Added the following:</p> <ul style="list-style-type: none">• 2 new LDAP extension function, ldap_destroy (page 139), ldap_dup (page 141).• A new session preference option, “LDAP_OPT_SESSION_REFCNT” on page 417.

February 2004	Renamed the product name from “NDS” to “Novell eDirectory” at relevant instances.
October 2003	<p>Added the following:</p> <ul style="list-style-type: none"> • Support for the HP-UX platform • Information about Section 1.7.2, “LBURP,” on page 43. • 6 new LDAP extension functions, ldap_lburp_end_request (page 337), ldap_lburp_operation_request (page 338), ldap_lburp_parse_operation_response (page 340), ldap_lburp_start_request (page 341), ldap_parse_lburp_end_response (page 352), and ldap_parse_lburp_start_response (page 354). • 2 new structures, LBURPUpdateResult (page 454), and LBURPUpdateOperationList (page 455).
June 2003	<p>Added the following:</p> <ul style="list-style-type: none"> • Changed LDAP event system to eDirectory event system
September 2002	<p>Added the following:</p> <ul style="list-style-type: none"> • Information on referral handling, outlined in Section 1.6, “Referral Handling in LDAP v3,” on page 40 • 2 new session options, LDAP_OPT_REFERRAL_LIST and LDAP_OPT_NETWORK_TIMEOUT. See Section 1.6, “Referral Handling in LDAP v3,” on page 40 and “Setting Initial Connection Timeout” on page 22 for conceptual information on these new options. • Fixed errors in the <code>ldap_event</code> function descriptions <p>Added the following functions:</p> <ul style="list-style-type: none"> • ldap_multisort_entries (page 197)
May 2002	<p>Added the following:</p> <ul style="list-style-type: none"> • Information on the new SASL authentication mechanisms, outlined in “Authentication” on page 27. • LDAP event system • startTLS and stopTLS <p>Added the following functions:</p> <ul style="list-style-type: none"> • ldap_bind_digest_md5_start (page 91) • ldapssl_install_routines (page 161) • ldap_bind_digest_md5_finish (page 93) • ldap_bind_nmas_s (page 95) • ldapssl_start_tls (page 313) • ldapssl_stop_tls (page 314)
February 2002	<p>Added the following:</p> <ul style="list-style-type: none"> • Information on SSL Certificates, outlined in “SSL Certificates” on page 31. • Schema Parsing Functions including: • New DirLoad driver for the Novell Import Convert Export utility.

September 2001	<p>Added the following:</p> <ul style="list-style-type: none"> • More information on search filters. • More information on time formats. • Interactive SSL APIs
June 2001	<p>Added the following:</p> <ul style="list-style-type: none"> • Information on LDAP URLs. • Updated the LDAPURLDesc (page 472) struct to maintain IETF conformance. • Update the ldap_get_effective_privileges (page 331) function with new rights flags. • Added documentation for the DELIM handler in the Novell Import Convert Export Utility. • Updated the LDAP Utilities documentation. • Removed Multi-byte functions. • Updated the documentation for the LDAP unbind functions.
February 2001	<p>Added the following:</p> <ul style="list-style-type: none"> • New functions—<code>ldap_set_replication_filter</code>, <code>ldap_get_replication_filter</code>, <code>ldap_create_orphan_naming_context</code>, and <code>ldap_remove_orphan_naming_context</code>. • New UTF-8 conversion routines • Updated the LDAPMod structure • Replaced the <code>LDAP_OPT_ERROR_NUMBER</code> constant with <code>LDAP_OPT_RESULT_CODE</code> • Updated the <code>fbuf</code> parameter description for the <code>ber_free</code> function.
September 2000	<p>Added the following:</p> <ul style="list-style-type: none"> • Information to the <code>ldap_url_parse</code> function • Runtime information • XML rule information to the Novell Import Convert Export utility
July 2000	<p>Added the following:</p> <ul style="list-style-type: none"> • Support for the Solaris and Linux platforms • A task for changing a user's password • An <code>ldapx_memfree</code> function to free memory allocated by the LDAP extension library • <code>ldapadd</code> and Novell Import/Export utility information • LDIF examples
May 2000	<p>Added information about functions that allocate and free memory.</p> <p>Added the following new functions: <code>ldap_set_rebind_proc</code>, <code>ldap_is_ldap_url</code>, <code>ldap_is_ldaps_url</code>, <code>ldap_free_urldesc</code>, <code>ldap_url_search</code>, <code>ldap_url_search_s</code>, <code>ldap_url_search_st</code>, and <code>ldap_refresh_server</code>.</p> <p>Added the following structure: <code>LDAPURLDesc</code>.</p>
